



A free, open-source alternative to Mathematica

Mathics Core Version 6.0.0

The Mathics3 Team

February 25, 2023

Contents

I. Manual	5
1. Introduction	6
2. Language Tutorial	8
3. Further Tutorial Examples	30
4. Django-based Web Interface	35
II. Reference of Built-in Symbols	39
1. Applying Functions to Lists	40
2. Arithmetic Functions	45
3. Assignments	52
4. Atomic Elements of Expressions	65
5. Binary Data	84
6. Code Compilation	88
7. Colors	90
8. Date and Time	114
9. Definition Attributes	121
10. Descriptive Statistics	130
11. Distance and Similarity Measures	135
12. Drawing Graphics	142
13. Drawing Options and Option Values	167
14. Expression Structure	175
15. File Formats	182
16. File and Stream Operations	186
17. Filesystem Operations	197
18. Forms of Input and Output	211
19. Function Application	220

20. Functional Composition and Operator Forms	223
21. Functional Programming	225
22. Functions used in Quantum Mechanics	236
23. Global System Information	239
24. Graphics and Drawing	245
25. Image Manipulation	281
26. Importing and Exporting	311
27. Input and Output	318
28. Input/Output, Files, and Filesystem	319
29. Integer Functions	348
30. Integer and Number-Theoretical Functions	357
31. Interactive Manipulation	419
32. Layout	420
33. List Functions	427
34. Low level Format definitions	458
35. Mathematical Functions	460
36. Mathematical Optimization	471
37. Matrices and Linear Algebra	472
38. Message-related functions.	475
39. Numerical Functions	479
40. Operations on Vectors	484
41. Options Management	490
42. Physical and Chemical data	496
43. Procedural Programming	498
44. Rules and Patterns	504
45. Scoping Constructs	516
46. Solving Recurrence Equations	520
47. Sparse Array Functions	521
48. Special Functions	522
49. Strings and Characters	547

50. Tensors	563
51. Testing Expressions	568
52. The Main Loop	590
53. Tracing Built-in Functions	594
54. Units and Quantities	599
III. Mathics3 Modules	602
1. Graphs - Vertices and Edges	603
2. Natural Language Processing	637
IV. License	645
A. GNU General Public License	646
B. Included software and data	780
Index	784
Colophon	796

Part I.

Manual

1. Introduction

Mathics3—to be pronounced like “Mathematics” without the “emat”—is a general-purpose computer algebra system (CAS). It is meant to be a free, open-source alternative to *Mathematica®*. It is free both as in “free beer” and as in “freedom”. *Mathics* can be run *Mathics3* locally, and to facilitate installation of the vast amount of software need to run this, there is a docker image available on dockerhub.

The programming language of *Mathics3* is meant to resemble the *Wolfram Language* as much as possible. However, *Mathics3* is in no way affiliated or supported by *Wolfram*. *Mathics3* will probably never have the power to compete with *Mathematica®* in industrial applications; it is an alternative though. It also invites community development at all levels.

See the installation instructions for the most recent instructions for installing from PyPI, or the source. For implementation details see <https://mathics-development-guide.readthedocs.io/en/latest/>.

Contents

History	6	What is missing?	7	Why yet another CAS, one based on <i>Mathematica</i> ?	7
What does <i>Mathics3</i> offer?	7				

History

The first alpha versions of *Mathics3* were done in 2011 by Jan Pöschko. He worked on it for a couple of years to about the v0.5 release in 2012. By then, it had 386 built-in symbols. Currently there are over a 1,000 and even more when *Mathics3* modules are included.

After that, Angus Griffith took over primary leadership and rewrote the parser to pretty much the stage it is in now. He and later Ben Jones worked on it from 2013 to about 2017 to the v1.0 release. Towards the end of this period, Bernhard Liebl worked on this, mostly focused on graphics.

A docker image of the v.9 release can be found on dockerhub.

Around 2017, the project was largely abandoned in its largely Python 2.7 state, with support for Python 3.2-3.5 via six.

Subsequently, around mid 2020, it was picked up by the current developers. A list of authors and contributors can be found in the AUTHORS.txt file.

What does *Mathics3* offer?

Some of the features of *Mathics3* tries to be compatible with Wolfram-Language kernel within the confines of the Python ecosystem.

Given this, it is a powerful functional programming language, driven by pattern matching and rule application.

Primitive types include rationals, complex numbers, and arbitrary-precision numbers. Other primitive types such as images or graphs, or NLP come from the various Python libraries that *Mathics3* uses.

Outside of the “core” *Mathics3* kernel (which has a only primitive command-line interface), in separate github projects, as add-ons, there is:

- a Django-based web server
- a command-line interface using either prompt-toolkit, or GNU Readline

- a Mathics3 module for Graphs (via NetworkX),
- a Mathics3 module for NLP (via nltk, spacy, and others)
- a A docker container which bundles all of the above

What is missing?

There are lots of ways in which *Mathics3* could still be improved. *FUTURE.rst* has the current roadmap. While we always could use help, such as in Python programming, improving Documentation. But there are other ways to help. For example:

- Ensure this document is complete and accurate. We could use help to ensure all of the Built-in functions described properly and fully, and that they have link to corresponding Wiki, Sympy, WMA and/or mpath links. Make sure the builtin summaries and examples clear and useful.
- We could use help in LaTeX styling, and going over this document to remove overful boxes and things of that nature. We could also use help and our use of Asymptote. There are some graphics primitives such as for polyhedra that haven't been implemented. Similar graphics options are sometimes missing in Asymptote that we have available in other graphics backends.
- add another graphics backend: it could be a javascript library like jsfiddle
- Consider donating via Github Sponsors or some other mechanism.

Why yet another CAS, one based on Mathematica?

Mathematica® is great, but it has a couple of disadvantages.

- It is not open source.
- Its development is tightly controlled and centralized, and as such
- it can't hook into different kinds open-source packages that have independently developed algorithms and methods

The second point some may find an advantage.

However, even if you are willing to pay hundreds of dollars for the software, you would not be able to see what's going on "inside" the program if that is your interest. That's what free, open-source, and community-supported software is for!

Mathics3 aims at combining the best of both worlds: the beauty of *Mathematica*® backed by a free, extensible Python core which includes a rich set of Python tools including:

- mpmath for floating-point arithmetic with arbitrary precision,
- numpy for numeric computation,
- SymPy for symbolic mathematics, and
- optionally SciPy for Scientific calculations.

Performance of *Mathics3* is not, right now, practical in large-scale projects and calculations. However it can be used as a tool for exploration and education.

2. Language Tutorial

The following sections are introductions to the basic principles of the language of *Mathics3*. A few examples and functions are presented. Only their most common usages are listed; for a full description of a Symbols possible arguments, options, etc., see its entry in the Reference of Built-in Symbols.

However if you google for “Mathematica Tutorials” you will find easily dozens of other tutorials which are applicable. Be warned though that *Mathics3* does not yet offer the full range and features and capabilities of *Mathematica*®.

Contents

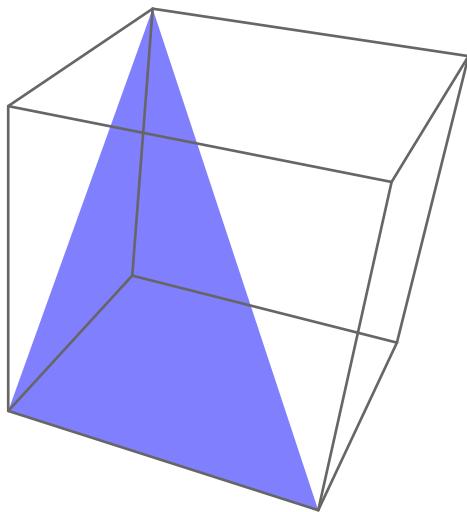
3D Graphics	9	Graphics Introduction		Strings	25
Basic calculations . . .	11	Examples	20	Symbols and	
Comparisons and		Plotting Introduction		Assignments . . .	26
Boolean Logic . .	12	Examples	21	The Structure of	
Formatting Output . .	15	Precision and Accuracy	22	<i>Mathics3 Objects</i>	27
Functions and Patterns	17	Program-Flow Control		Working with Lists . .	29
		Statements	23		
		Scoping	24		

3D Graphics

Three-dimensional graphics are created using the function `Graphics3D` and a list of 3D primitives. The following primitives are supported so far:

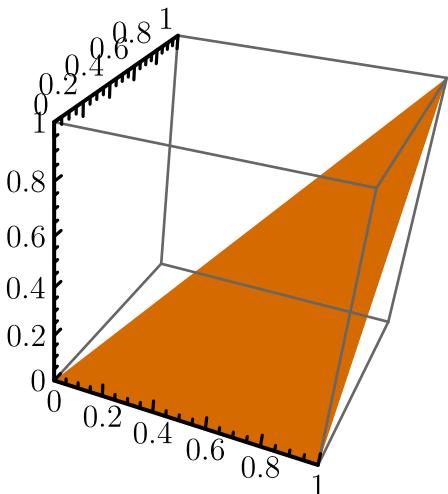
```
Polygon[{{x1, y1, z1}, {x2, y2, z3}, ...}]
    draws a filled polygon.
Line[{{x1, y1, z1}, {x2, y2, z3}, ...}]
    draws a line.
Point[{x1, y1, z1}]
    draws a point.
```

```
>> Graphics3D[Polygon[{{0,0,0}, {0,1,1}, {1,0,0}}]]
```



Colors can also be added to three-dimensional primitives.

```
>> Graphics3D[{Orange, Polygon[{{0,0,0}, {1,1,1}, {1,0,0}}]}, Axes->True]
```



`Graphics3D` produces a `Graphics3DBox`:

```
>> Head[ToBoxes[Graphics3D[{Polygon[]}]]]
```

Graphics3DBox

Basic calculations

Mathics3 can be used to calculate basic stuff:

```
>> 1 + 2
```

3

To submit a command to *Mathics3*, press Shift+Return in the Web interface or Return in the console interface. The result will be printed in a new line below your query.

Mathics3 understands all basic arithmetic operators and applies the usual operator precedence. Use parentheses when needed:

```
>> 1 - 2 * (3 + 5) / 4  
-3
```

The multiplication can be omitted:

```
>> 1 - 2 (3 + 5) / 4  
-3  
  
>> 2 4  
8
```

Powers can be entered using \wedge :

```
>> 3 ^ 4  
81
```

Integer divisions yield rational numbers:

```
>> 6 / 4  
3  
—  
2
```

To convert the result to a floating point number, apply the function N :

```
>> N[6 / 4]  
1.5
```

As you can see, functions are applied using square braces [and], in contrast to the common notation of (and). At first hand, this might seem strange, but this distinction between function application and precedence change is necessary to allow some general syntax structures, as you will see later.

Mathics3 provides many common mathematical functions and constants, e.g.:

```
>> Log[E]  
1  
  
>> Sin[Pi]  
0  
  
>> Cos[0.5]  
0.877583
```

When entering floating point numbers in your query, *Mathics3* will perform a numerical evaluation and present a numerical result, pretty much like if you had applied N .

Of course, *Mathics3* has complex numbers:

```
>> Sqrt[-4]  
2I  
  
>> I ^ 2  
-1  
  
>> (3 + 2 I)^ 4  
- 119 + 120I  
  
>> (3 + 2 I)^ (2.5 - I)  
43.663 + 8.28556I  
  
>> Tan[I + 0.5]  
0.195577 + 0.842966I
```

Abs calculates absolute values:

```
>> Abs[-3]  
3
```

```
>> Abs[3 + 4 I]  
5
```

Mathics3 can operate with pretty huge numbers:

```
>> 100!  
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286
```

(! denotes the factorial function.) The precision of numerical evaluation can be set:

```
>> N[Pi, 30]  
3.14159265358979323846264338328
```

Division by zero is forbidden:

```
>> 1 / 0  
Infinite expression 1 / 0 encountered.  
ComplexInfinity
```

Other expressions involving Infinity are evaluated:

```
>> Infinity + 2 Infinity  
∞
```

In contrast to combinatorial belief, 0^0 is undefined:

```
>> 0 ^ 0  
Indeterminate expression 0 ^ 0 encountered.  
Indeterminate
```

The result of the previous query to *Mathics3* can be accessed by %:

```
>> 3 + 4  
7  
>> % ^ 2  
49
```

Comparisons and Boolean Logic

Values can be compared for equality using the operator ==:

```
>> 3 == 3  
True  
>> 3 == 4  
False
```

The special symbols True and False are used to denote truth values. Naturally, there are inequality comparisons as well:

```
>> 3 > 4  
False
```

Inequalities can be chained:

```
>> 3 < 4 >= 2 != 1  
True
```

Truth values can be negated using ! (logical *not*) and combined using && (logical *and*) and || (logical *or*):

```
>> !True  
False
```

```

>> !False
True

>> 3 < 4 && 6 > 5
True

&& has higher precedence than ||, i.e. it binds stronger:

>> True && True || False && False
True

>> True && (True || False)&& False
False

```

Formatting Output

The way results are formatted for output in *Mathics3* is rather sophisticated, as compatibility to the way *Mathematica*® does things is one of the design goals. It can be summed up in the following procedure:

1. The result of the query is calculated.
2. The result is stored in `Out` (which `%` is a shortcut for).
3. Any `Format` rules for the desired output form are applied to the result. In the console version of *Mathics3*, the result is formatted as `OutputForm`; `MathMLForm` for the `StandardForm` is used in the interactive Web version; and `TeXForm` for the `StandardForm` is used to generate the `LATEX` version of this documentation.
4. `MakeBoxes` is applied to the formatted result, again given either `OutputForm`, `MathMLForm`, or `TeXForm` depending on the execution context of *Mathics3*. This yields a new expression consisting of “box constructs”.
5. The boxes are turned into an ordinary string and displayed in the console, sent to the browser, or written to the documentation `LATEX` file.

As a consequence, there are various ways to implement your own formatting strategy for custom objects.

You can specify how a symbol shall be formatted by assigning values to `Format`:

```

>> Format[x] = "y";

>> x
y

```

This will apply to `MathMLForm`, `OutputForm`, `StandardForm`, `TeXForm`, and `TraditionalForm`.

```

>> x // InputForm
x

```

You can specify a specific form in the assignment to `Format`:

```

>> Format[x, TeXForm] = "z";

>> x // TeXForm
\text{z}

```

Special formats might not be very relevant for individual symbols, but rather for custom functions (objects):

```

>> Format[r[args___]] = "<an r object>";

>> r[1, 2, 3]
<an r object>

```

You can use several helper functions to format expressions:

```

Infix[expr, op]
    formats the arguments of expr with infix operator op.
Prefix[expr, op]
    formats the argument of expr with prefix operator op.
Postfix[expr, op]
    formats the argument of expr with postfix operator op.
StringForm[form, arg1, arg2, ...]
    formats arguments using a format string.

```

```

>> Format[r[args___]] = Infix[{args}, "~"];
>> r[1, 2, 3]
1 ~ 2 ~ 3
>> StringForm["`1` and `2`", n, m]
n and m

```

There are several methods to display expressions in 2-D:

```

Row[{...}]
    displays expressions in a row.
Grid[{{...}}]
    displays a matrix in two-dimensional form.
Subscript[expr, i1, i2, ...]
    displays expr with subscript indices i1, i2, ...
Superscript[expr, exp]
    displays expr with superscript (exponent) exp.

```

```

>> Grid[{{a, b}, {c, d}}]
a   b
c   d
>> Subscript[a, 1, 2] // TeXForm
a_{1,2}

```

If you want even more low-level control over expression display, override **MakeBoxes**:

```

>> MakeBoxes[b, StandardForm] = "c";
>> b
c

```

This will even apply to **TeXForm**, because **TeXForm** implies **StandardForm**:

```

>> b // TeXForm
c

```

Except some other form is applied first:

```

>> b // OutputForm // TeXForm
b

```

MakeBoxes for another form:

```

>> MakeBoxes[b, TeXForm] = "d";
>> b // TeXForm
d

```

You can cause a much bigger mess by overriding `MakeBoxes` than by sticking to `Format`, e.g. generate invalid XML:

```
>> MakeBoxes[c, MathMLForm] = "<not closed";  
  
>> c // MathMLForm  
<not closed
```

However, this will not affect formatting of expressions involving `c`:

```
>> c + 1 // MathMLForm  
<math display="block"><mrow><mn>1</mn>  
<mo>+</mo> <mi>c</mi></mrow></math>
```

That's because `MathMLForm` will, when not overridden for a special case, call `StandardForm` first. `Format` will produce escaped output:

```
>> Format[d, MathMLForm] = "<not closed";  
  
>> d // MathMLForm  
<math display="block"><mtext>&lt;not&nbsp;closed</mtext></math>  
  
>> d + 1 // MathMLForm  
<math display="block"><mrow><mn>1</mn> <mo>+</mo>  
<mtext>&lt;not&nbsp;closed</mtext></mrow></math>
```

For instance, you can override `MakeBoxes` to format lists in a different way:

```
>> MakeBoxes[{items___}, StandardForm] := RowBox[{"[", Sequence @@  
Riffle[MakeBoxes /@ {items}, " "], "]"}]  
  
>> {1, 2, 3}  
[123]
```

However, this will not be accepted as input to *Mathics3* anymore:

```
>> [1 2 3]  
  
>> Clear[MakeBoxes]
```

By the way, `MakeBoxes` is the only built-in symbol that is not protected by default:

```
>> Attributes[MakeBoxes]  
{HoldAllComplete}
```

`MakeBoxes` must return a valid box construct:

```
>> MakeBoxes[squared[args___], StandardForm] := squared[args]^2  
  
>> squared[1, 2]  
  
>> squared[1, 2] // TeXForm
```

=

The desired effect can be achieved in the following way:

```
>> MakeBoxes[squared[args___], StandardForm] := SuperscriptBox[RowBox[{  
MakeBoxes[squared], "[", RowBox[Riffle[MakeBoxes[#]& /@ {args},  
, "]], "]"]}], 2]  
  
>> squared[1, 2]  
squared[1, 2]^2
```

You can view the box structure of a formatted expression using `ToBoxes`:

```
>> ToBoxes[m + n]
RowBox[{m, +, n}]
```

The list elements in this `RowBox` are strings, though string delimiters are not shown in the default output form:

```
>> InputForm[%]
RowBox[{"m", "+", "n"}]
```

Functions and Patterns

Functions can be defined in the following way:

```
>> f[x_] := x ^ 2
```

This tells *Mathics3* to replace every occurrence of `f` with one (arbitrary) parameter `x` with `x ^ 2`.

```
>> f[3]
9
>> f[a]
a^2
```

The definition of `f` does not specify anything for two parameters, so any such call will stay unevaluated:

```
>> f[1, 2]
f[1, 2]
```

In fact, *functions* in *Mathics3* are just one aspect of *patterns*: `f[x_]` is a pattern that *matches* expressions like `f[3]` and `f[a]`. The following patterns are available:

<code>_ or Blank[]</code>	matches one expression.
<code>Pattern[x, p]</code>	matches the pattern <code>p</code> and stores the value in <code>x</code> .
<code>x_ or Pattern[x, Blank[]]</code>	matches one expression and stores it in <code>x</code> .
<code>-- or BlankSequence[]</code>	matches a sequence of one or more expressions.
<code>--- or BlankNullSequence[]</code>	matches a sequence of zero or more expressions.
<code>_h or Blank[h]</code>	matches one expression with head <code>h</code> .
<code>x_h or Pattern[x, Blank[h]]</code>	matches one expression with head <code>h</code> and stores it in <code>x</code> .
<code>p q or Alternatives[p, q]</code>	matches either pattern <code>p</code> or <code>q</code> .
<code>p ? t or PatternTest[p, t]</code>	matches <code>p</code> if the test <code>t[p]</code> yields True.
<code>p /; c or Condition[p, c]</code>	matches <code>p</code> if condition <code>c</code> holds.
<code>Verbatim[p]</code>	matches an expression that equals <code>p</code> , without regarding patterns inside <code>p</code> .

As before, patterns can be used to define functions:

```

>> g[s___] := Plus[s] ^ 2
>> g[1, 2, 3]
36

```

MatchQ[e, p] tests whether e matches p:

```

>> MatchQ[a + b, x_ + y_]
True
>> MatchQ[6, _Integer]
True

```

ReplaceAll (/.) replaces all occurrences of a pattern in an expression using a Rule given by ->:

```

>> {2, "a", 3, 2.5, "b", c} /. x_Integer -> x ^ 2
{4, a, 9, 2.5, b, c}

```

You can also specify a list of rules:

```

>> {2, "a", 3, 2.5, "b", c} /. {x_Integer -> x ^ 2.0, y_String -> 10}
{4., 10, 9., 2.5, 10, c}

```

ReplaceRepeated (//.) applies a set of rules repeatedly, until the expression doesn't change anymore:

```

>> {2, "a", 3, 2.5, "b", c} //.{x_Integer -> x ^ 2.0, y_String -> 10}
{4., 100., 9., 2.5, 100., c}

```

There is a “delayed” version of Rule which can be specified by :> (similar to the relation of := to =):

```

>> a :> 1 + 2
a:>1 + 2
>> a -> 1 + 2
a -> 3

```

This is useful when the right side of a rule should not be evaluated immediately (before matching):

```

>> {1, 2} /. x_Integer -> N[x]
{1, 2}

```

Here, N is applied to x before the actual matching, simply yielding x. With a delayed rule this can be avoided:

```

>> {1, 2} /. x_Integer :> N[x]
{1., 2.}

```

ReplaceAll and ReplaceRepeated take the first possible match. However ReplaceList returns a list of all possible matches. This can be used to get all subsequences of a list, for instance:

```

>> ReplaceList[{a, b, c}, {___, x___, ___} -> {x}]
{{a}, {a, b}, {a, b, c}, {b}, {b, c}, {c}}

```

ReplaceAll would just return the first expression:

```

>> ReplaceAll[{a, b, c}, {___, x___, ___} -> {x}]
{a}

```

In addition to defining functions as rules for certain patterns, there are *pure* functions that can be defined using the & postfix operator, where everything before it is treated as the function body and # can be used as argument placeholder:

```

>> h = # ^ 2 &;

```

```
>> h[3]
9
```

Multiple arguments can simply be indexed:

```
>> sum = #1 + #2 &;
>> sum[4, 6]
10
```

It is also possible to name arguments using Function:

```
>> prod = Function[{x, y}, x * y];
>> prod[4, 6]
24
```

Pure functions are very handy when functions are used only locally, e.g., when combined with operators like Map:

```
>> # ^ 2 & /@ Range[5]
{1, 4, 9, 16, 25}
```

Sort according to the second part of a list:

```
>> Sort[{{x, 10}, {y, 2}, {z, 5}}, #1[[2]] < #2[[2]] &]
{{y, 2}, {z, 5}, {x, 10}}
```

Functions can be applied using prefix or postfix notation, in addition to using [] :

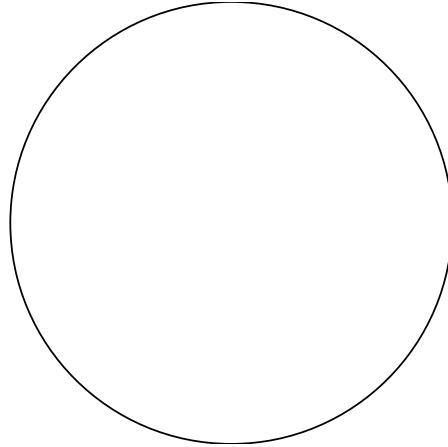
```
>> h @ 3
9
>> 3 // h
9
```

Graphics Introduction Examples

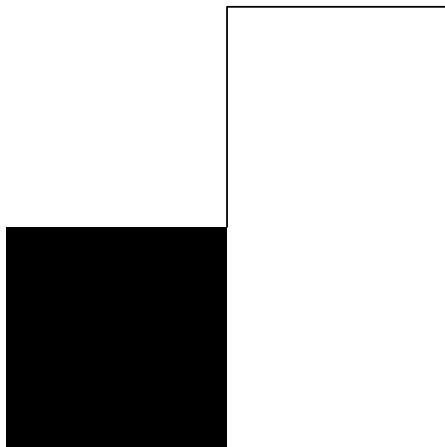
Two-dimensional graphics can be created using the function `Graphics` and a list of graphics primitives. For three-dimensional graphics see the following section. The following primitives are available:

```
Circle[{x, y}, r]
  draws a circle.
Disk[{x, y}, r]
  draws a filled disk.
Rectangle[{x1, y1}, {x2, y2}]
  draws a filled rectangle.
Polygon[{{x1, y1}, {x2, y2}, ...}]
  draws a filled polygon.
Line[{{x1, y1}, {x2, y2}, ...}]
  draws a line.
Text[text, {x, y}]
  draws text in a graphics.
```

```
>> Graphics[{Circle[{0, 0}, 1]}]
```



```
>> Graphics[{Line[{{0, 0}, {0, 1}, {1, 1}, {1, -1}}], Rectangle[{0, 0}, {-1, -1}]}]
```



Colors can be added in the list of graphics primitives to change the drawing color. The following ways to specify colors are supported:

`RGBColor[r, g, b]`

specifies a color using red, green, and blue.

`CMYKColor[c, m, y, k]`

specifies a color using cyan, magenta, yellow, and black.

`Hue[h, s, b]`

specifies a color using hue, saturation, and brightness.

All

`GrayLevel[l]`

specifies a color using a gray level.

components range from 0 to 1. Each color function can be supplied with an additional argument specifying the desired opacity ("alpha") of the color. There are many predefined colors, such as `Black`, `White`, `Red`, `Green`, `Blue`, etc.

```
>> Graphics[{Red, Disk[]}]
```

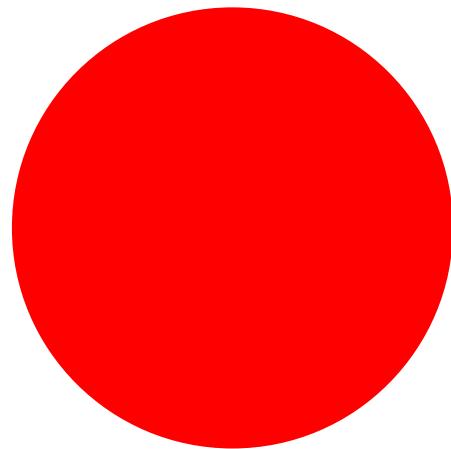
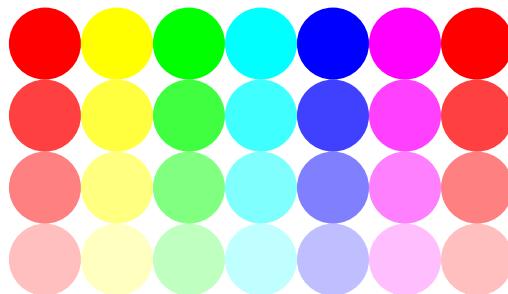


Table of hues:

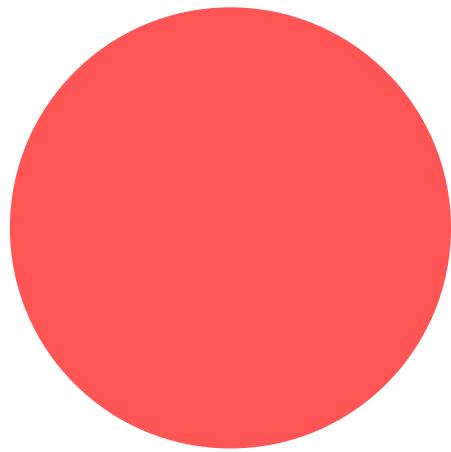
```
>> Graphics[Table[{Hue[h, s], Disk[{12h, 8s}]}}, {h, 0, 1, 1/6}, {s, 0, 1, 1/4}]]
```



Colors can be mixed and altered using the following functions:

```
Blend[{color1, color2}, ratio]
mixes color1 and color2 with ratio, where a ratio of 0 returns color1 and a ratio of 1 returns color2.
Lighter[color]
makes color lighter (mixes it with White).
Darker[color]
makes color darker (mixes it with Black).
```

```
>> Graphics[{Lighter[Red], Disk[]}]
```



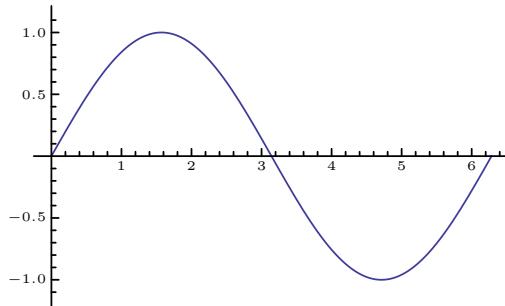
Graphics produces a GraphicsBox:

```
>> Head[ToBoxes[Graphics[{Circle[]}]]]  
GraphicsBox
```

Plotting Introduction Examples

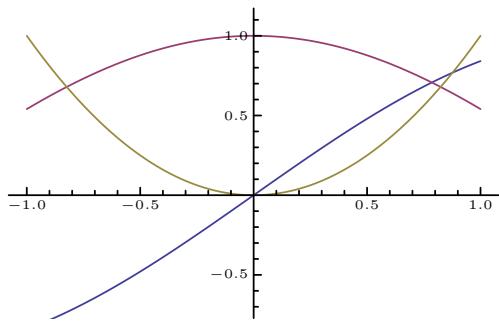
Mathics3 can plot functions:

```
>> Plot[Sin[x], {x, 0, 2 Pi}]
```



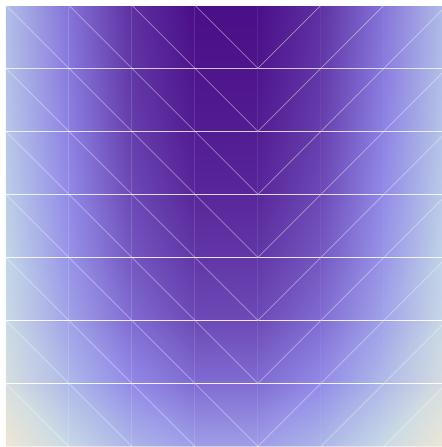
You can also plot multiple functions at once:

```
>> Plot[{Sin[x], Cos[x], x^2}, {x, -1, 1}]
```



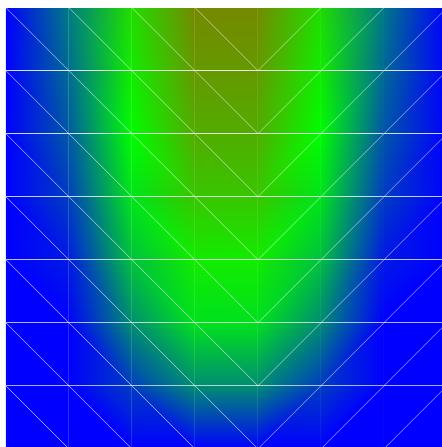
Two-dimensional functions can be plotted using DensityPlot:

```
>> DensityPlot[x^2 + 1/y, {x, -1, 1}, {y, 1, 4}]
```



You can use a custom coloring function:

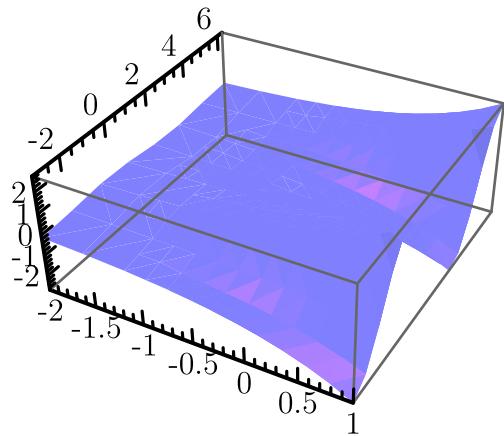
```
>> DensityPlot[x^2 + 1/y, {x, -1, 1}, {y, 1, 4}, ColorFunction -> (Blend[{Red, Green, Blue}, #]&)]
```



One problem with `DensityPlot` is that it's still very slow, basically due to function evaluation being pretty slow in general—and `DensityPlot` has to evaluate a lot of functions.

Three-dimensional plots are supported as well:

```
>> Plot3D[Exp[x] Cos[y], {x, -2, 1}, {y, -Pi, 2 Pi}]
```



Precision and Accuracy

Mathics3 handles relative and absolute uncertainties in numerical quantities. The *precision* or relative accuracy, is set by adding a RawBackquote character (`) and the number of digits of precision in the mantissa. For example:

```
>> 3.1416`3  
3.14
```

Above, two decimal places are shown in output after the decimal point, but three places of precision are stored.

The relative uncertainty of $3.1416`3$ is 10^{-3} . It is numerically equivalent, in three places after the decimal point, to $3.1413`4$:

```
>> 3.1416`3 == 3.1413`4  
True
```

We can get the precision of the number by using the *Mathics3* Built-in function *Precision* of section 4:

```
>> Precision[3.1413`4]  
4.
```

While 3.1419 not the closest approximation to Pi in 4 digits after the decimal point (or with precision 4), for 3 digits of precision it is:

```
>> Pi == 3.141987654321`3  
True
```

<url>The absolute accuracy of a number, is set by adding a two RawBackquotes `` and the number digits.

For example:

```
>> 13.1416``4  
13.142
```

is a number having a absolute uncertainty of 10^{-4} . This number is numerically equivalent to $13.1413``4$:

```
>> 13.1416``4 == 13.1413``4  
True
```

The absolute accuracy for the value 0 is a fixed-precision Real number:

```
>> 0``4  
0.0000
```

See also Accuracy and precision.

Program-Flow Control Statements

Like most programming languages, *Mathics3* has common program-flow control statements for conditions, loops, etc.:

```

If [cond, pos, neg]
    returns pos if cond evaluates to True, and neg if it evaluates to False.
Which [cond1, expr1, cond2, expr2, ...]
    yields expr1 if cond1 evaluates to True, expr2 if cond2 evaluates to True, etc.
Do [expr, {i, max}]
    evaluates expr max times, substituting i in expr with values from 1 to max.
For [start, test, incr, body]
    evaluates start, and then iteratively body and incr as long as test evaluates to True.
While [test, body]
    evaluates body as long as test evaluates to True.
Nest [f, expr, n]
    returns an expression with f applied n times to expr.
NestWhile [f, expr, test]
    applies a function f repeatedly on an expression expr, until applying test on the result no longer yields True.
FixedPoint [f, expr]
    starting with expr, repeatedly applies f until the result no longer changes.

```

```

>> If[2 < 3, a, b]
a
>> x = 3; Which[x < 2, a, x > 4, b, x < 5, c]
c

```

Compound statements can be entered with ; . The result of a compound expression is its last part or Null if it ends with a ; .

```

>> 1; 2; 3
3
>> 1; 2; 3;

```

Inside For, While, and Do loops, Break[] exits the loop and Continue[] continues to the next iteration.

```

>> For[i = 1, i <= 5, i++, If[i == 4, Break[]]; Print[i]]
1
2
3

```

Scoping

By default, all symbols are “global” in *Mathics3*, i.e. they can be read and written in any part of your program. However, sometimes “local” variables are needed in order not to disturb the global namespace. *Mathics3* provides two ways to support this:

- *lexicalscoping* by Module, and
- *dynamicscoping* by Block.

```

Module[{vars}, expr]
    localizes variables by giving them a temporary name of the form name$number, where number is the current value of $ModuleNumber . Each time a module is evaluated, $ModuleNumber is incremented.
Block[{vars}, expr]
    temporarily stores the definitions of certain variables, evaluates expr with reset values and restores the original definitions afterwards.

```

Both scoping constructs shield inner variables from affecting outer ones:

```

>> t = 3;

```

```

>> Module[{t}, t = 2]
2

>> Block[{t}, t = 2]
2

>> t
3

Module creates new variables:
>> y = x ^ 3;

>> Module[{x = 2}, x * y]
2x3

Block does not:
>> Block[{x = 2}, x * y]
16

Thus, Block can be used to temporarily assign a value to a variable:
>> expr = x ^ 2 + x;

>> Block[{x = 3}, expr]
12

>> x
x

Block can also be used to temporarily change the value of system parameters:
>> Block[{$RecursionLimit = 30}, x = 2 x]
Recursion depth of 30 exceeded.
$Aborted

>> f[x_] := f[x + 1]; Block[{$IterationLimit = 30}, f[1]]
Iteration limit of 30 exceeded.
$Aborted

```

It is common to use scoping constructs for function definitions with local variables:

```

>> fac[n_] := Module[{k, p}, p = 1; For[k = 1, k <= n, ++k, p *= k]; p]

>> fac[10]
3628800

>> 10!
3628800

```

Strings

Strings can be entered with " as delimiters:

```

>> "Hello world!"
Hello world!

```

As you can see, quotation marks are not printed in the output by default. This can be changed by using `InputForm`:

```

>> InputForm["Hello world!"]
"Hello world!"

```

Strings can be joined using `<>`:

```
>> "Hello" <> " " <> "world!"  
Hello world!
```

Numbers cannot be joined to strings:

```
>> "Debian" <> 6  
String expected.  
Debian<>6
```

They have to be converted to strings using `ToString` first:

```
>> "Debian" <> ToString[6]  
Debian6
```

Symbols and Assignments

Symbols need not be declared in *Mathics3*, they can just be entered and remain variable:

```
>> x  
x
```

Basic simplifications are performed:

```
>> x + 2 x  
3x
```

Symbols can have any name that consists of characters and digits:

```
>> iAm1Symbol ^ 2  
iAm1Symbol2
```

You can assign values to symbols:

```
>> a = 2  
2  
>> a ^ 3  
8  
>> a = 4  
4  
>> a ^ 3  
64
```

Assigning a value returns that value. If you want to suppress the output of any result, add a ; to the end of your query:

```
>> a = 4;
```

Values can be copied from one variable to another:

```
>> b = a;
```

Now changing a does not affect b:

```
>> a = 3;  
  
>> b  
4
```

Such a dependency can be achieved by using “delayed assignment” with the `:=` operator (which does not return anything, as the right side is not even evaluated):

```

>> b := a ^ 2
>> b
9
>> a = 5;
>> b
25

```

The Structure of *Mathics3* Objects

Every expression in *Mathics3* is built upon the same principle: it consists of a *head* and an arbitrary number of *children*, unless it is an *atom*, i.e. it can not be subdivided any further. To put it another way: everything is a function call. This can be best seen when displaying expressions in their “full form”:

```

>> FullForm[a + b + c]
Plus[a, b, c]

```

Nested calculations are nested function calls:

```

>> FullForm[a + b * (c + d)]
Plus[a, Times[b, Plus[c, d]]]

```

Even lists are function calls of the function *List*:

```

>> Head[{1, 2, 3}]
List

```

However, its full form is presented with \${ldots}\$

```

>> FullForm[{1, 2, 3}]
{1, 2, 3}

```

The head of an expression can be determined with *Head*:

```

>> Head[a + b + c]
Plus

```

The children of an expression can be accessed like list elements:

```

>> (a + b + c)[[2]]
b

```

The head is the 0th element:

```

>> (a + b + c)[[0]]
Plus

```

The head of an expression can be exchanged using the function *Apply*:

```

>> Apply[g, f[x, y]]
g[x, y]

```

```

>> Apply[Plus, a * b * c]
a + b + c

```

Apply can be written using the operator @@:

```

>> Times @@ {1, 2, 3, 4}
24

```

(This exchanges the head `List` of `{1, 2, 3, 4}` with `Times`, and then the expression `Times[1, 2, 3, 4]` is evaluated, yielding 24.) `Apply` can also be applied on a certain *level* of an expression:

```
>> Apply[f, {{1, 2}, {3, 4}}, {1}]  
{f[1,2],f[3,4]}
```

Or even on a range of levels:

```
>> Apply[f, {{1, 2}, {3, 4}}, {0, 2}]  
f[f[1,2],f[3,4]]
```

`Apply` is similar to `Map` (`/@`):

```
>> Map[f, {1, 2, 3, 4}]  
{f[1],f[2],f[3],f[4]}  
  
>> f /@ {{1, 2}, {3, 4}}  
{f[{1,2}],f[{3,4}]}
```

The atoms of *Mathics3* are numbers, symbols, and strings. `AtomQ` tests whether an expression is an atom:

```
>> AtomQ[5]  
True  
  
>> AtomQ[a + b]  
False
```

The full form of rational and complex numbers looks like they were compound expressions:

```
>> FullForm[3 / 5]  
Rational[3,5]  
  
>> FullForm[3 + 4 I]  
Complex[3,4]
```

However, they are still atoms, thus unaffected by applying functions, for instance:

```
>> f @@ Complex[3, 4]  
3 + 4I
```

Nevertheless, every atom has a head:

```
>> Head /@ {1, 1/2, 2.0, I, "a string", x}  
{Integer,Rational,Real,Complex,String,Symbol}
```

The operator `==` tests whether two expressions are the same on a structural level:

```
>> 3 == 3  
True  
  
>> 3 == 3.0  
True
```

But:

```
>> 3 === 3.0  
False
```

because 3 (an `Integer`) and 3.0 (a `Real`) are structurally different.

Working with Lists

Lists can be entered in *Mathics3* with curly braces `{` and `}`:

```
>> mylist = {a, b, c, d}
{a, b, c, d}
```

There are various functions for constructing lists:

```
>> Range[5]
{1, 2, 3, 4, 5}

>> Array[f, 4]
{f[1], f[2], f[3], f[4]}

>> ConstantArray[x, 4]
{x, x, x, x}

>> Table[n ^ 2, {n, 2, 5}]
{4, 9, 16, 25}
```

The number of elements of a list can be determined with Length:

```
>> Length[mylist]
4
```

Elements can be extracted using double square braces:

```
>> mylist[[3]]
c
```

Negative indices count from the end:

```
>> mylist[[-3]]
b
```

Lists can be nested:

```
>> mymatrix = {{1, 2}, {3, 4}, {5, 6}};
```

There are alternate forms to display lists:

```
>> TableForm[mymatrix]
```

```
1 2
3 4
5 6
```

```
>> MatrixForm[mymatrix]

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

```

There are various ways of extracting elements from a list:

```
>> mymatrix[[2, 1]]
```

```
3
```

```
>> mymatrix[[;;, 2]]
```

```
{2, 4, 6}
```

```
>> Take[mylist, 3]
```

```
{a, b, c}
```

```

>> Take[mylist, -2]
{c,d}

>> Drop[mylist, 2]
{c,d}

>> First[mymatrix]
{1,2}

>> Last[mylist]
d

>> Most[mylist]
{a,b,c}

>> Rest[mylist]
{b,c,d}

```

Lists can be used to assign values to multiple variables at once:

```

>> {a, b} = {1, 2};

>> a
1

>> b
2

```

Operations like addition and multiplication, “thread” over lists; lists are combined element-wise:

```

>> {1, 2, 3} + {4, 5, 6}
{5,7,9}

>> {1, 2, 3} * {4, 5, 6}
{4,10,18}

```

It is an error to combine lists with unequal lengths:

```

>> {1, 2} + {4, 5, 6}
Objects of unequal length cannot be combined.

{1,2} + {4,5,6}

```

3. Further Tutorial Examples

Contents

Curve sketching	31	Dice	33	Linear algebra	34
---------------------------	----	----------------	----	--------------------------	----

Curve sketching

Let's sketch the function

```
>> f[x_] := 4 x / (x ^ 2 + 3 x + 5)
```

The derivatives are:

```
>> {f'[x], f''[x], f'''[x]} // Together
{(-4 (-5 + x^2))/(5 + 3 x + x^2)^2, 8 (-15 - 15 x + x^3)/((5 + 3 x + x^2)^3), -24 (-20 - 60 x - 30 x^2 + x^4)/((5 + 3 x + x^2)^4)}
```

To get the extreme values of f , compute the zeroes of the first derivatives:

```
>> extremes = Solve[f'[x] == 0, x]
{{x -> -Sqrt[5]}, {x -> Sqrt[5]}}
```

And test the second derivative:

```
>> f''[x] /. extremes // N
{1.65086, -0.064079}
```

Thus, there is a local maximum at $x = \text{Sqrt}[5]$ and a local minimum at $x = -\text{Sqrt}[5]$. Compute the inflection points numerically, chopping imaginary parts close to 0:

```
>> inflections = Solve[f''[x] == 0, x] // N // Chop
{{x -> -1.0852}, {x -> -3.21463}, {x -> 4.29983}}
```

Insert into the third derivative:

```
>> f'''[x] /. inflections
{-3.67683, 0.694905, 0.00671894}
```

Being different from 0, all three points are actual inflection points. f is not defined where its denominator is 0:

```
>> Solve[Denominator[f[x]] == 0, x]
{{x -> -3/2 - I/2 Sqrt[11]}, {x -> -3/2 + I/2 Sqrt[11]}}
```

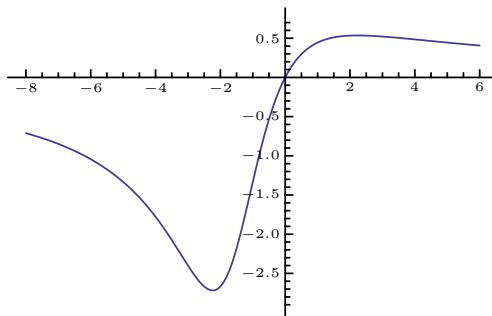
These are non-real numbers, consequently f is defined on all real numbers. The behaviour of f at the boundaries of its definition:

```
>> Limit[f[x], x -> Infinity]
0
```

```
>> Limit[f[x], x -> -Infinity]
0
```

Finally, let's plot f:

```
>> Plot[f[x], {x, -8, 6}]
```



Dice

Let's play with dice in this example. A Dice object shall represent the outcome of a series of rolling a dice with six faces, e.g.:

```
>> Dice[1, 6, 4, 4]
Dice[1, 6, 4, 4]
```

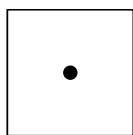
Like in most games, the ordering of the individual throws does not matter. We can express this by making Dice Orderless:

```
>> SetAttributes[Dice, Orderless]
>> Dice[1, 6, 4, 4]
Dice[1, 4, 4, 6]
```

A dice object shall be displayed as a rectangle with the given number of points in it, positioned like on a traditional dice:

```
>> Format[Dice[n_Integer?(1 <= # <= 6 &)] := Block[{p = 0.2, r = 0.05},
  Graphics[{EdgeForm[Black], White, Rectangle[], Black, EdgeForm[], If
  [OddQ[n], Disk[{0.5, 0.5}, r]], If[MemberQ[{2, 3, 4, 5, 6}, n], Disk
  [{p, p}, r]], If[MemberQ[{2, 3, 4, 5, 6}, n], Disk[{1 - p, 1 - p}, r
  ]], If[MemberQ[{4, 5, 6}, n], Disk[{p, 1 - p}, r]], If[MemberQ[{4, 5,
  6}, n], Disk[{1 - p, p}, r]], If[n === 6, {Disk[{p, 0.5}, r], Disk
  [{1 - p, 0.5}, r]}]}, ImageSize -> Tiny]]
```

```
>> Dice[1]
```



The empty series of dice shall be displayed as an empty dice:

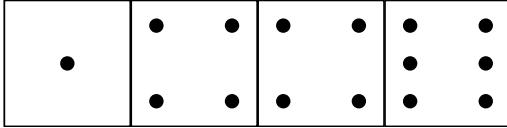
```
>> Format[Dice[]] := Graphics[{EdgeForm[Black], White, Rectangle[]}],
ImageSize -> Tiny]
```

```
>> Dice[]
```

Any non-empty series of dice shall be displayed as a row of individual dice:

```
>> Format[Dice[d__Integer?(1 <= # <= 6 &)]]:=Row[Dice /@ {d}]
```

```
>> Dice[1, 6, 4, 4]
```



Note that *Mathics3* will automatically sort the given format rules according to their “generality”, so the rule for the empty dice does not get overridden by the rule for a series of dice. We can still see the original form by using *InputForm*:

```
>> Dice[1, 6, 4, 4] // InputForm
```

```
Dice[1, 4, 4, 6]
```

We want to combine *Dice* objects using the *+* operator:

```
>> Dice[a__] + Dice[b__] ^:= Dice[Sequence @@ {a, b}]
```

The *^:=* (*UpSetDelayed*) tells *Mathics3* to associate this rule with *Dice* instead of *Plus*. *Plus* is protected—we would have to unprotect it first:

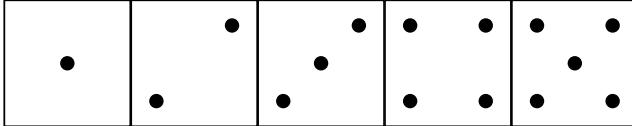
```
>> Dice[a__] + Dice[b__] := Dice[Sequence @@ {a, b}]
```

Tag Plus in Dice[a__] + Dice[b__] is Protected.

\$Failed

We can now combine dice:

```
>> Dice[1, 5] + Dice[3, 2] + Dice[4]
```



Let's write a function that returns the sum of the rolled dice:

```
>> DiceSum[Dice[d__]] := Plus @@ {d}
```

```
>> DiceSum @ Dice[1, 2, 5]
```

```
8
```

And now let's put some dice into a table:

```
>> Table[{Dice[Sequence @@ d], DiceSum @ Dice[Sequence @@ d]}, {d, {{1, 2}, {2, 2}, {2, 2}, {2, 6}}}] // TableForm
```

	3
	4
	8

It is not very sophisticated from a mathematical point of view, but it's beautiful.

Linear algebra

Let's consider the matrix

```
>> A = {{1, 1, 0}, {1, 0, 1}, {0, 1, 1}};  
  
>> MatrixForm[A]  

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

```

We can compute its eigenvalues and eigenvectors:

```
>> Eigenvalues[A]  
{2, -1, 1}  
  
>> Eigenvectors[A]  
{\{1, 1, 1\}, \{1, -2, 1\}, \{-1, 0, 1\}}
```

This yields the diagonalization of A:

```
>> T = Transpose[Eigenvectors[A]]; MatrixForm[T]  

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & 0 \\ 1 & 1 & 1 \end{pmatrix}  
  
>> Inverse[T] . A . T // MatrixForm  

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}  
  
>> % == DiagonalMatrix[Eigenvalues[A]]  
True$$$$

```

We can solve linear systems:

```
>> LinearSolve[A, {1, 2, 3}]  
{0, 1, 2}  
  
>> A . %  
{1, 2, 3}
```

In this case, the solution is unique:

```
>> NullSpace[A]  
{}
```

Let's consider a singular matrix:

```
>> B = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  
  
>> MatrixRank[B]  
2  
  
>> s = LinearSolve[B, {1, 2, 3}]  

$$\left\{-\frac{1}{3}, \frac{2}{3}, 0\right\}$$

```

```
>> NullSpace[B]
{{1, -2, 1}}
>> B . (RandomInteger[100] * %[[1]] + s)
{1, 2, 3}
```

4. Django-based Web Interface

In the future, we plan on providing an interface to Jupyter as a separate package.

However currently as part *Mathics3*, we distribute a browser-based interface using long-term-release (LTS) Django 4.

Since a Jupyter-based interface seems preferable to the home-grown interface described here, it is doubtful whether there will be future improvements to the this interface.

When you enter Mathics in the top after the Mathics logo and the word “Mathics” you’ll see a *menubar*. It looks like this:



These save and load worksheets, share sessions, run a gallery of examples, go to the GitHub organization page, and provide information about the particular Mathics3 installation.

These are explained in the sections below.

Contents

Gallery Examples	35	Saving, Loading, and Deleting Worksheets	37	Loading and Deleting Worksheets	38
Keyboard Commands	36	Saving Worksheets	37	URIs	38
Persistence of Mathics Definitions in a Session	36				

Gallery Examples

We have a number of examples showing off briefly some of the capabilities of the system. These are run when you hit the button that looks like this:



It is also shown in the pop-up text that appears when Mathics3 is first run.

Keyboard Commands

There are some keyboard commands you can use in the Django-based Web interface of *Mathics3*.

Shift+Return

This evaluates the current cell (the most important one, for sure). On the right-hand side you may also see an "=" button which can be clicked to do the same thing.

Ctrl+D

This moves the cursor over to the documentation pane on the right-hand side. From here you can perform a search for a pre-defined *Mathics3* function, or symbol. Clicking on the "?" symbol on the right-hand side does the same thing.

Ctrl+C

This moves the cursor back to document code pane area where you type *Mathics3* expressions

Ctrl+S

Save worksheet

Ctrl+O

Open worksheet

Right Click on MathML output

Opens MathJax Menu

Of special note is the last item on the list: right-click to open the MathJax menu. Under "Math Setting"/"Zoom Trigger", if the zoom trigger is set to a value other than "No Zoom", then when that trigger is applied on MathML formatted output, the MathML formula pop up a window for the formula. The window can show the formula larger. Also, this is a way to see output that is too large to fit on the display since the window allows for scrolling.

Keyboard commands behavior depends the browser used, the operating system, desktop settings, and customization. We hook into the desktop "Open the current document" and "Save the current document" functions that many desktops provide. For example see: Finding keyboard shortcuts

Often, these shortcut keyboard command are only recognized when a text field has focus; otherwise, the browser might do some browser-specific actions, like setting a bookmark etc.

Persistence of *Mathics* Definitions in a Session

When you use the Django-based Web interface of *Mathics3*, a browser session is created. Cookies have to be enabled to allow this. Your session holds a key which is used to access your definitions that are stored in a database on the server. As long as you don't clear the cookies in your browser, your definitions will remain even when you close and re-open the browser.

This implies that you should not store sensitive, private information in *Mathics3* variables when using the online Web interface. In addition to their values being stored in a database on the server, your queries might be saved for debugging purposes. However, the fact that they are transmitted over plain HTTP should make you aware that you should not transmit any sensitive information. When you want to do calculations with that kind of stuff, simply install *Mathics3* locally!

If you are using a public terminal, to erase all your definitions and close the browser window. When you use *Mathics3* in a browser, use the command `Quit[]` or its alias, `Exit[]`.

When you reload the current page in a browser using the default URL, e.g `http://localhost:8000`, all of the previous input and output disappears.

On the other hand, Definitions as described above do not, unless `Quit[]` or `Exit[]` is entered as described above.

If you want a URL that will record the input entered the `GenerateInputHash` button does this. The button looks like this:



For example, assuming you have a *Mathics3* server running at port 8000 on `localhost`, and you enter the URL `http://localhost:8000/#cXV1cmllcz14`, you should see a single line of input containing `x` entered.

Of course, what the value of this is when evaluated depends on whether `x` has been previously defined.

Saving, Loading, and Deleting Worksheets

<subsection title="Saving Worksheets">

Worksheets exist in the browser window only and are not stored on the server, by default. To save all your queries and results, use the *Save* button which is the middle graphic of the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+S" key combination may do the same thing.

<subsection title="Loading and Deleting Worksheets">

Saved worksheets can be loaded or deleted using the *FileOpen* button which is the left-most button in the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+O" key combination may do the same thing.

A pop-up menu should appear with the list of saved worksheets with an option to either load or delete the worksheet.

Saving Worksheets

<subsection title="Saving Worksheets">

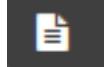
Worksheets exist in the browser window only and are not stored on the server, by default. To save all your queries and results, use the *Save* button which is the middle graphic of the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+S" key combination may do the same thing.

<subsection title="Loading and Deleting Worksheets">

Saved worksheets can be loaded or deleted using the *FileOpen* button which is the left-most button in the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+O" key combination may do the same thing.

A pop-up menu should appear with the list of saved worksheets with an option to either load or delete the worksheet.

Loading and Deleting Worksheets

<subsection title="Saving Worksheets">

Worksheets exist in the browser window only and are not stored on the server, by default. To save all your queries and results, use the *Save* button which is the middle graphic of the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+S" key combination may do the same thing.

<subsection title="Loading and Deleting Worksheets">

Saved worksheets can be loaded or deleted using the *FileOpen* button which is the left-most button in the menu bar. It looks like this:



Depending on browser, desktop, and OS-settings, the "Ctrl+O" key combination may do the same thing. A pop-up menu should appear with the list of saved worksheets with an option to either load or delete the worksheet.

URIs

For the most part, the application is a single-page application. Assuming you are running locally or on a host called `localhost` using the default port, 8000, here are some URLs and what they do:

`http://localhost:8000`

The single-page application; the main page.

`http://localhost:8000/about`

A page giving:

- the software versions of this package and version information of important software this uses.
- directory path information for the current setup
- machine information
- system information
- customizable system settings

`http://localhost:8000/doc`

An on-line formatted version of the documentation, which include this text. You can see this as a right side frame of the main page, when clicking "?" on the right-hand upper corner.

Part II.

Reference of Built-in Symbols

1. Applying Functions to Lists

Many computations can be conveniently specified in terms of applying functions in parallel to many elements in a list.

Many mathematical functions are automatically taken to be “listable”, so that they are always applied to every element in a list.

Contents

Apply (@@)	41	MapAt	42	Scan	43
List	41	MapIndexed	43	Thread	44
Map (/@)	41	MapThread	43		

Apply (@@)

```
Apply[f, expr]
f @@ expr
  replaces the head of expr with f.
Apply[f, expr, levelspec]
  applies f on the parts specified by levelspec.
```

```
>> f @@ {1, 2, 3}
f[1,2,3]

>> Plus @@ {1, 2, 3}
6
```

The head of *expr* need not be List:

```
>> f @@ (a + b + c)
f[a,b,c]
```

Apply on level 1:

```
>> Apply[f, {a + b, g[c, d, e * f], 3}, {1}]
{f[a,b],f[c,d,ef],3}
```

The default level is 0:

```
>> Apply[f, {a, b, c}, {0}]
f[a,b,c]
```

Range of levels, including negative level (counting from bottom):

```
>> Apply[f, {{{{a}}}}, {2, -3}]
{{f[f[{a}]]}}
```

Convert all operations to lists:

```
>> Apply[List, a + b * c ^ e * f[g], {0, Infinity}]
{a, {b, {g}}, {c, e}}
```

List

WMA link

```
List[e1, e2, ..., ei]
{e1, e2, ..., ei}
represents a list containing the elements e1...ei.
```

List is the head of lists:

```
>> Head[{1, 2, 3}]
List
```

Lists can be nested:

```
>> {{a, b, {c, d}}}
{{a, b, {c, d}}}
```

Map (/@)

```
Map[f, expr] or f /@ expr
applies f to each part on the first level of expr.
Map[f, expr, levelspec]
applies f to each level specified by levelspec of expr.
```

```
>> f /@ {1, 2, 3}
{f[1], f[2], f[3]}

>> #^2 & /@ {1, 2, 3, 4}
{1, 4, 9, 16}
```

Map f on the second level:

```
>> Map[f, {{a, b}, {c, d, e}}, {2}]
{{f[a], f[b]}, {f[c], f[d], f[e]}}
```

Include heads:

```
>> Map[f, a + b + c, Heads -> True]
f[Plus][f[a], f[b], f[c]]
```

MapAt

```
MapAt[f, expr, n]
    applies f to the element at position n in expr. If n is negative, the position is counted from the end.
MapAt[f, expr, {i, j ...}]
    applies f to the part of expr at position {i, j, ...}.
MapAt[f, pos]
    represents an operator form of MapAt that can be applied to an expression.
```

Map f onto the part at position 2:

```
>> MapAt[f, {a, b, c, d}, 2]
{a, f[b], c, d}
```

Map f onto multiple parts:

```
>> MapAt[f, {a, b, c, d}, {{1}, {4}}]
{f[a], b, c, f[d]}
```

Map f onto the at the end:

```
>> MapAt[f, {a, b, c, d}, -1]
{a, b, c, f[d]}
```

Map f onto an association:

```
>> MapAt[f, <"a" -> 1, "b" -> 2, "c" -> 3, "d" -> 4, "e" -> 5|>, 3]
{a -> 1, b -> 2, c -> f[3], d -> 4, e -> 5}
```

Use negative position in an association:

```
>> MapAt[f, <"a" -> 1, "b" -> 2, "c" -> 3, "d" -> 4|>, -3]
{a -> 1, b -> f[2], c -> 3, d -> 4}
```

Use the operator form of MapAt:

```
>> MapAt[f, 1][{a, b, c, d}]
{f[a], b, c, d}
```

MapIndexed

```
MapIndexed[f, expr]
    applies f to each part on the first level of expr, including the part positions in the call to f.
MapIndexed[f, expr, levelspec]
    applies f to each level specified by levelspec of expr.
```

```
>> MapIndexed[f, {a, b, c}]
{f[a, {1}], f[b, {2}], f[c, {3}]}
```

Include heads (index 0):

```
>> MapIndexed[f, {a, b, c}, Heads -> True]
f[List, {0}] [f[a, {1}], f[b, {2}], f[c, {3}]]
```

Map on levels 0 through 1 (outer expression gets index {}):

```
>> MapIndexed[f, a + b + c * d, {0, 1}]
f [f [a, {1}] + f [b, {2}] + f [c d, {3}], {}]
```

Get the positions of atoms in an expression (convert operations to List first to disable Listable functions):

```
>> expr = a + b * f[g] * c ^ e;
>> listified = Apply[List, expr, {0, Infinity}];
>> MapIndexed[#, listified, {-1}]
{{1}, {{2, 1}, {{2, 2, 1}}, {{2, 3, 1}, {2, 3, 2}}}}
```

Replace the heads with their positions, too:

```
>> MapIndexed[#, listified, {-1}, Heads -> True]
{0} [{1}, {2, 0} [{2, 1}, {2, 2, 0} [{2, 2, 1}], {2, 3, 0} [{2, 3, 1}, {2, 3, 2}]]]
```

The positions are given in the same format as used by Extract. Thus, mapping Extract on the indices given by MapIndexed re-constructs the original expression:

```
>> MapIndexed[Extract[expr, #2] &, listified, {-1}, Heads -> True]
a + b f [g] c ^ e
```

MapThread

```
'MapThread[f, {{a1, a2, ...}, {b1, b2, ...}, ...}]
  returns {f[a1, b1, ...], f[a2, b2, ...], ...}.
MapThread[f, {expr1, expr2, ...}, n]
  applies f at level n.
```

```
>> MapThread[f, {{a, b, c}, {1, 2, 3}}]
{f [a, 1], f [b, 2], f [c, 3]}

>> MapThread[f, {{{a, b}, {c, d}}, {{e, f}, {g, h}}}, 2]
{{f [a, e], f [b, f]}, {f [c, g], f [d, h]}}
```

Scan

```
Scan[f, expr]
  applies f to each element of expr and returns Null.
'Scan[f, expr, levelspec]
  applies f to each level specified by levelspec of expr.
```

```
>> Scan[Print, {1, 2, 3}]
1
2
3
```

Thread

```
Thread[f[args]]  
  threads f over any lists that appear in args.  
Thread[f[args], h]  
  threads over any parts with head h.
```

```
>> Thread[f[{a, b, c}]]  
{f[a], f[b], f[c]}  
  
>> Thread[f[{a, b, c}, t]]  
{f[a, t], f[b, t], f[c, t]}  
  
>> Thread[f[a + b + c], Plus]  
f[a] + f[b] + f[c]
```

Functions with attribute `Listable` are automatically threaded over lists:

```
>> {a, b, c} + {d, e, f} + g  
{a + d + g, b + e + g, c + f + g}
```

2. Arithmetic Functions

Arithmetic Functions are functions that work on individual numbers, lists, and arrays: in either symbolic or algebraic forms.

Contents

Basic Arithmetic	45	Plus (+)	47	Times (*)	50
CubeRoot	45	Power (^)	48	Sums, Simple Statistics	50
Divide (/)	46	Sqrt	49	Accumulate	50
Minus (-)	46	Subtract (-)	49	Total	51

Basic Arithmetic

Basic Arithmetic

The functions here are the basic arithmetic operations that you might find on a calculator.

CubeRoot

Cube root (WMA)

`CubeRoot[n]`

finds the real-valued cube root of the given n .

```
>> CubeRoot[16]
221/3
```

Divide (/)

Division (WMA link)

`Divide[a, b]`

a / b

represents the division of a by b .

```
>> 30 / 5
6
>> 1 / 8
>> Pi / 4
π
4
```

Use `N` or a decimal point to force numeric evaluation:

```

>> Pi / 4.0
0.785398

>> 1 / 8
0.125

Nested divisions:
>> a / b / c

$$\frac{a}{bc}$$


>> a / (b / c)

$$\frac{ac}{b}$$


>> a / b / (c / (d / e))

$$\frac{ad}{bce}$$


>> a / (b ^ 2 * c ^ 3 / e)

$$\frac{ae}{b^2c^3}$$


```

Minus (-)

Additive inverse (WMA)

Minus [*expr*]
is the negation of *expr*.

```

>> -a //FullForm
Times [-1, a]

```

Minus automatically distributes:

```

>> -(x - 2/3)

$$\frac{2}{3} - x$$


```

Minus threads over lists:

```

>> -Range[10]
{-1, -2, -3, -4, -5, -6, -7, -8, -9, -10}

```

Plus (+)

Addition (SymPy, WMA)

Plus [*a*, *b*, ...]
a + *b* + ...
represents the sum of the terms *a*, *b*, ...

```

>> 1 + 2
3

```

Plus performs basic simplification of terms:

```
>> a + b + a  
2a + b  
  
>> a + a + 3 * a  
5a  
  
>> a + b + 4.5 + a + b + a + 2 + 1.5 b  
6.5 + 3a + 3.5b
```

Apply Plus on a list to sum up its elements:

```
>> Plus @@ {2, 4, 6}  
12
```

The sum of the first 1000 integers:

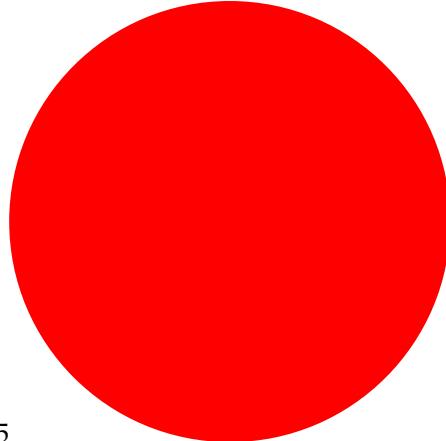
```
>> Plus @@ Range[1000]  
500500
```

Plus has default value 0:

```
>> DefaultValues[Plus]  
{HoldPattern[Default[Plus]] :> 0}  
  
>> a /. n_. + x_ :> {n, x}  
{0, a}
```

The sum of 2 red circles and 3 red circles is...

```
>> 2 Graphics[{Red,Disk[]}] + 3 Graphics[{Red,Disk[]}]
```



5

Power (^)

Exponentiation (SymPy, WMA)

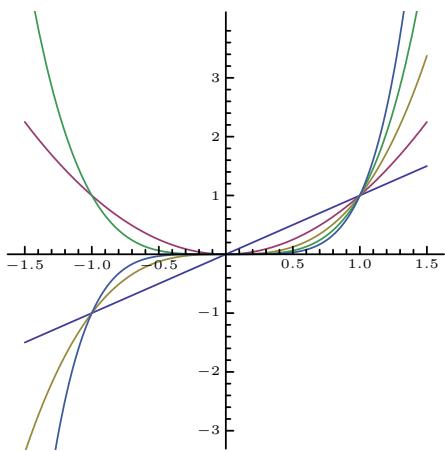
```
Power[a, b]  
a ^ b  
represents a raised to the power of b.
```

```
>> 4 ^ (1/2)  
2
```

```

>> 4 ^ (1/3)
22/3
>> 3^123
48519278097689642681155855396759336072749841943521979872827
>> (y ^ 2) ^ (1/2)
 $\sqrt{y^2}$ 
>> (y ^ 2) ^ 3
y6
>> Plot[Evaluate[Table[x^y, {y, 1, 5}]], {x, -1.5, 1.5}, AspectRatio -> 1]

```



Use a decimal point to force numeric evaluation:

```

>> 4.0 ^ (1/3)
1.5874

```

Power has default value 1 for its second argument:

```

>> DefaultValues[Power]
{HoldPattern[Default[Power, 2]] :> 1}

>> a /. x_ ^ n_. :> {x, n}
{a, 1}

```

Power can be used with complex numbers:

```

>> (1.5 + 1.0 I)^ 3.5
- 3.68294 + 6.95139 I

>> (1.5 + 1.0 I)^ (3.5 + 1.5 I)
- 3.19182 + 0.645659 I

```

Sqrt

Square root (SymPy, WMA)

Sqrt [*expr*]
returns the square root of *expr*.

```

>> Sqrt[4]
2

>> Sqrt[5]
 $\sqrt{5}$ 

>> Sqrt[5] // N
2.23607

>> Sqrt[a]^2
a

```

Complex numbers:

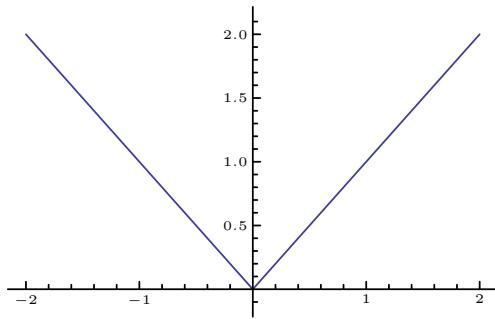
```

>> Sqrt[-4]
2I

>> I == Sqrt[-1]
True

>> Plot[Sqrt[a^2], {a, -2, 2}]

```



Subtract (-)

Subtraction, (WMA)

```

Subtract[a, b]
a - b
represents the subtraction of b from a.

```

```

>> 5 - 3
2

>> a - b // FullForm
Plus[a, Times[-1, b]]

>> a - b - c
a - b - c

>> a - (b - c)
a - b + c

```

Times (*)

Multiplication (SymPy, WMA)

```

Times[a, b, ...]
a * b * ...
a b ...
represents the product of the terms a, b, ...

```

```

>> 10 * 2
20

>> 10 2
20

>> a * a
a2

>> x ^ 10 * x ^ -2
x8

>> {1, 2, 3} * 4
{4,8,12}

>> Times @@ {1, 2, 3, 4}
24

>> IntegerLength[Times@@Range[5000]]
16326

Times has default value 1:
>> DefaultValues[Times]
{HoldPattern[Default[Times]] :> 1}

>> a /. n_. * x_ :> {n, x}
{1,a}

```

Sums, Simple Statistics

Sums, Simple Statistics
These functions perform a simple arithmetic computation over a list.

Accumulate

WMA link

```

Accumulate[list]
accumulates the values of list, returning a new list.

```

```

>> Accumulate[{1, 2, 3}]
{1,3,6}

```

Total

WMA link

```
Total [list]
    adds all values in list.
Total [list, n]
    adds all values up to level n.
Total [list, {n}]
    totals only the values at level n.
Total [list, {n_1, n_2}]
    totals at levels n_1, n_2.
```

```
>> Total[{1, 2, 3}]
6
>> Total[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
{12,15,18}
Total over rows and columns
>> Total[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, 2]
45
Total over rows instead of columns
>> Total[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, {2}]
{6,15,24}
```

3. Assignments

Assignments allow you to set or clear variables, indexed variables, structure elements, functions, and general transformations.

You can also get assignment and documentation information about symbols.

Contents

Clearing Assignments	52	UpSet (^=)	59	SubtractFrom (-=)	61
Clear	53	UpSetDelayed		TimesBy (*=)	62
ClearAll	53	(^:=)	59	Types of Values	62
Remove	54	In-place binary		DefaultValues	62
Unset (=.)	55	assignment		Messages	63
Forms of Assignment	55	operator	59	NValues	63
LoadModule	55	AddTo (+=)	59	SubValues	64
Set (=)	56	Decrement (--)	60	UpValue-related	
SetDelayed (:=)	57	DivideBy (/=)	60	assignments	64
TagSet	58	Increment (++)	60	UpValues	64
TagSetDelayed	58	PreDecrement (--)	61		
		PreIncrement (++)	61		

Clearing Assignments

Clearing Assignment

Clear

WMA link

`Clear[symb1, symb2, ...]`

clears all values of the given symbols. The arguments can also be given as strings containing symbol names.

```
>> x = 2;  
  
>> Clear[x]  
  
>> x  
  
>> x = 2;  
  
>> y = 3;  
  
>> Clear["Global`*"]  
  
>> x
```

```
>> y  
y
```

ClearAll may not be called for Protected symbols.

```
>> Clear[Sin]
```

The values and rules associated with built-in symbols will not get lost when applying Clear (after unprotecting them):

```
>> Unprotect[Sin]
```

```
>> Clear[Sin]
```

```
>> Sin[Pi]  
0
```

Clear does not remove attributes, messages, options, and default values associated with the symbols.

Use ClearAll to do so.

```
>> Attributes[r] = {Flat, Orderless};
```

```
>> Clear["r"]
```

```
>> Attributes[r]  
{Flat, Orderless}
```

ClearAll

WMA link

```
ClearAll[symb1, symb2, ...]
```

clears all values, attributes, messages and options associated with the given symbols. The arguments can also be given as strings containing symbol names.

```
>> x = 2;  
  
>> ClearAll[x]  
  
>> x  
x  
  
>> Attributes[r] = {Flat, Orderless};  
  
>> ClearAll[r]  
  
>> Attributes[r]  
{}
```

ClearAll may not be called for Protected or Locked symbols.

```
>> Attributes[lock] = {Locked};
```

```
>> ClearAll[lock]  
Symbol lock is locked.
```

Remove

WMA link

```
Remove[x]
removes the definition associated to x.
```

```
>> a := 2
>> Names["Global'a"]
>> Remove[a]
>> Names["Global'a"]
```

Unset (=.)

WMA link

```
Unset[x]
x=.
removes any value belonging to x.
```

```
>> a = 2
2
>> a =.
>> a
a
```

Unsetting an already unset or never defined variable will not change anything:

```
>> a =.
>> b =.
```

Unset can unset particular function values. It will print a message if no corresponding rule is found.

```
>> f[x_] =.
>> f[x_] := x ^ 2
>> f[3]
>> f[x_] =.
>> f[3]
```

You can also unset OwnValues, DownValues, SubValues, and UpValues directly. This is equivalent to setting them to {}.

```
>> f[x_] = x; f[0] = 1;
>> DownValues[f] =.
>> f[2]
f[2]
```

Unset threads over lists:

```
>> a = b = 3;  
  
>> {a, {b}} =.  
{Null, {Null}}
```

Forms of Assignment

Forms of Assignment

LoadModule

```
LoadModule [module]  
'Load Mathics definitions from the python module module'
```

```
>> LoadModule["nomodule"]  
Python import errors with: No module named 'nomodule'.  
$Failed  
  
>> LoadModule["sys"]  
Python module "sys" is not a Mathics3 module.  
$Failed
```

Set (=)

WMA link

```
Set [expr, value]  
expr = value  
evaluates value and assigns it to expr.  
{s1, s2, s3} = {v1, v2, v3}  
sets multiple symbols (s1, s2, ...) to the corresponding values (v1, v2, ...).
```

Set can be used to give a symbol a value:

```
>> a = 3  
3  
  
>> a
```

An assignment like this creates an ownvalue:

```
>> OwnValues[a]  
{HoldPattern[a]:>3}
```

You can set multiple values at once using lists:

```
>> {a, b, c} = {10, 2, 3}  
{10, 2, 3}  
  
>> {a, b, {c, {d}}} = {1, 2, {{c1, c2}, {a}}}  
{1, 2, {{c1, c2}, {10}}}  
  
>> d  
10
```

`Set` evaluates its right-hand side immediately and assigns it to the left-hand side:

```
>> a  
  
>> x = a  
1  
  
>> a = 2  
2  
  
>> x  
1
```

`Set` always returns the right-hand side, which you can again use in an assignment:

```
>> a = b = c = 2;  
  
>> a == b == c == 2  
True
```

`Set` supports assignments to parts:

```
>> A = {{1, 2}, {3, 4}};  
  
>> A[[1, 2]] = 5  
5  
  
>> A  
  
>> A[;;, 2] = {6, 7}  
{6,7}  
  
>> A
```

`Set` a submatrix:

```
>> B = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  
  
>> B[[1;;2, 2;;-1]] = {{t, u}, {y, z}};  
  
>> B  
{\{1,t,u\},\{4,y,z\},\{7,8,9\}}
```

SetDelayed (:=)

WMA link

```
SetDelayed[expr, value]  
expr := value  
assigns value to expr, without evaluating value.
```

`SetDelayed` is like `Set`, except it has attribute `HoldAll`, thus it does not evaluate the right-hand side immediately, but evaluates it when needed.

```
>> Attributes[SetDelayed]  
{HoldAll, Protected, SequenceHold}  
  
>> a = 1  
1  
  
>> x := a
```

```
>> x
```

Changing the value of a affects x :

```
>> a = 2  
2  
>> x
```

Condition (/;) can be used with SetDelayed to make an assignment that only holds if a condition is satisfied:

```
>> f[x_] := p[x] /; x>0  
  
>> f[3]  
p[3]  
  
>> f[-3]  
f[-3]
```

It also works if the condition is set in the LHS:

```
>> F[x_, y_] /; x < y /; x>0 := x / y;  
  
>> F[x_, y_] := y / x;  
  
>> F[2, 3]  
2  
—  
3  
  
>> F[3, 2]  
2  
—  
3  
  
>> F[-3, 2]  
-2  
—  
3
```

We can use conditional delayed assignments to define symbols with values conditioned to the context.
For example,

```
>> ClearAll[a,b]; a/; b>0:= 3
```

Set a to have a value of 3 if certain variable b is positive. So, if this variable is not set, a stays unevaluated:

```
>> a  
a
```

If now we assign a positive value to b , then a is evaluated:

```
>> b=2; a  
3
```

TagSet

WMA link

```
TagSet[f, expr, value]  
f /: expr = value  
assigns value to expr, associating the corresponding assignment with the symbol f.
```

Create an upvalue without using UpSet:

```
>> x /: f[x] = 2
2
>> f[x]
2
>> DownValues[f]
{}
>> UpValues[x]
{HoldPattern[f[x]] :> 2}
```

The symbol *f* must appear as the ultimate head of *lhs* or as the head of an element in *lhs*:

```
>> x /: f[g[x]] = 3;
      Tag x not found or too deep for an assigned rule.

>> g /: f[g[x]] = 3;
>> f[g[x]]
3
```

TagSetDelayed

WMA link

```
TagSetDelayed[f, expr, value]
f /: expr := value
is the delayed version of TagSet.
```

UpSet ($\wedge=$)

WMA link

```
f[x]  $\wedge=$  expression
evaluates expression and assigns it to the value of f[x], associating the value with x.
```

UpSet creates an upvalue:

```
>> a[b]  $\wedge=$  3;
>> DownValues[a]
{}
>> UpValues[b]
{HoldPattern[a[b]] :> 3}
>> a  $\wedge=$  3
      Nonatomic expression expected.
3
```

You can use UpSet to specify special values like format values. However, these values will not be saved in UpValues:

```
>> Format[r]  $\wedge=$  "custom";
```

```

>> r
custom

>> UpValues[r]
{}
```

UpSetDelayed ($\wedge :=$)

WMA link

UpSetDelayed [*expression*, *value*]
expression $\wedge :=$ *value*
 assigns *expression* to the value of $f[x]$ (without evaluating *expression*), associating the value with *x*.

```

>> a[b]  $\wedge :=$  x

>> x = 2;

>> a[b]
2

>> UpValues[b]
{HoldPattern[a[b]] :> x}
```

In-place binary assignment operator

In-place binary assignment operator

There are a number operators and functions that combine assignment with some sort of binary operator. Sometimes a value is returned *before* the assignment occurs. When there is an operator for this, the operator is a prefix operator and the function name starts with `Pre`.

Sometimes the binary operation occurs first, and *then* the assignment occurs. When there is an operator for this, the operator is a postfix operator.

Infix operators combined with assignment end in `By`, `From`, or `To`.

AddTo ($+=$)

WMA link

AddTo [*x*, *dx*]
 $x += dx$
 is equivalent to $x = x + dx$.

```

>> a = 10;

>> a += 2
12

>> a
12
```

Decrement (--)

WMA link

```
Decrement[x]
x--
decrements x by 1, returning the original value of x.
```

```
>> a = 5;
```

```
>> a--
5
```

```
>> a
4
```

DivideBy (/=)

WMA link

```
DivideBy[x, dx]
x /= dx
is equivalent to x = x / dx.
```

```
>> a = 10;
```

```
>> a /= 2
5
```

```
>> a
5
```

Increment (++)

WMA link

```
Increment[x]
x++
increments x by 1, returning the original value of x.
```

```
>> a = 2;
```

```
>> a++
2
```

```
>> a
3
```

Grouping of Increment, PreIncrement and Plus:

```
>> ++++a+++++2//Hold//FullForm
Hold [Plus [PreIncrement [PreIncrement [Increment [Increment [a]]]], 2]]
```

PreDecrement (--)

WMA link

```
PreDecrement[x]
--x
    decrements x by 1, returning the new value of x.
```

--a is equivalent to $a = a - 1$:

```
>> a = 2;
```

```
>> --a
1
>> a
1
```

PreIncrement (++)

WMA link

```
PreIncrement[x]
++x
    increments x by 1, returning the new value of x.
```

++a is equivalent to $a = a + 1$:

```
>> a = 2;
```

```
>> ++a
3
>> a
3
```

SubtractFrom (-=)

WMA link

```
SubtractFrom[x, dx]
x -= dx
    is equivalent to  $x = x - dx$ .
```

```
>> a = 10;
>> a -= 2
8
>> a
8
```

TimesBy (*=)

WMA link

```
TimesBy[x, dx]
x *= dx
    is equivalent to  $x = x * dx$ .
```

```
>> a = 10;  
  
>> a *= 2  
20  
  
>> a  
20
```

Types of Values

Types of Value

DefaultValues

WMA link

DefaultValues [*symbol*]

gives the list of default values associated with *symbol*.

Note: this function is in Mathematica 5 but has been removed from current Mathematica.

```
>> Default[f, 1] = 4  
4  
  
>> DefaultValues[f]  
{HoldPattern[Default[f, 1]] :> 4}
```

You can assign values to DefaultValues:

```
>> DefaultValues[g] = {Default[g] -> 3};  
  
>> Default[g, 1]  
3  
  
>> g[x_.] := {x}  
  
>> g[a]  
{a}  
  
>> g[]  
{3}
```

Messages

WMA link

Messages [*symbol*]

gives the list of messages associated with *symbol*.

```
>> a::b = "foo"  
foo  
  
>> Messages[a]  
{HoldPattern[a::b] :> foo}
```

```

>> Messages[a] = {a::c :> "bar"};
>> a::c // InputForm
"bar"
>> Message[a::c]
bar

```

NValues

NValues[*symbol*]

gives the list of numerical values associated with *symbol*.

Note: this function is in Mathematica 5 but has been removed from current Mathematica.

```

>> NValues[a]
>> N[a] = 3;
>> NValues[a]

```

You can assign values to NValues:

```

>> NValues[b] := {N[b, MachinePrecision] :> 2}
>> N[b]
2.

```

Be sure to use SetDelayed, otherwise the left-hand side of the transformation rule will be evaluated immediately, causing the head of N to get lost. Furthermore, you have to include the precision in the rules; MachinePrecision will not be inserted automatically:

```

>> NValues[c] := {N[c] :> 3}
>> N[c]
c

```

Mathics will assign any list of rules to NValues; however, inappropriate rules will never be used:

```

>> NValues[d] = {foo -> bar};
>> NValues[d]
{HoldPattern[foo]:>bar}
>> N[d]
d

```

SubValues

WMA link

SubValues[*symbol*]

gives the list of subvalues associated with *symbol*.

Note: this function is not in current Mathematica.

```

>> f[1][x_] := x
>> f[2][x_] := x ^ 2

```

```

>> SubValues[f]
{HoldPattern[f[2][x_]] :> x2, HoldPattern[f[1][x_]] :> x}

>> Definition[f]
f[2][x_] = x2
f[1][x_] = x

```

UpValue-related assignments

UpValue-related assignments

An <i>UpValue</i> is a definition associated with a symbols that does not appear directly its head.
See [Associating Definitions with Different Symbols](#).

UpValues

WMA

UpValues [*symbol*]

gives the list of transformation rules corresponding to upvalues define with *symbol*.

```

>> a + b ^= 2
2

>> UpValues[a]
{HoldPattern[a + b] :> 2}

>> UpValues[b]
{HoldPattern[a + b] :> 2}

```

You can assign values to UpValues:

```

>> UpValues[pi] := {Sin[pi] :> 0}

>> Sin[pi]
0

```

4. Atomic Elements of Expressions

Expressions are ultimately built from a small number of distinct types of atomic elements.

Contents

Atomic Primitives	65	String Manipulation	72	ToExpression	77
AtomQ	66	Alphabet	73	ToString	77
Head	66	\$CharacterEncoding	73	Transliterate	78
Representation of Numbers	66	\$CharacterEncod-		Whitespace	78
Accuracy	67	ings	73	Symbol Handling	78
ExactNumberQ	68	Hexadeci-		Context	78
InexactNumberQ	68	malChar-		Definition	80
IntegerExponent	69	acter	73	DownValues	81
IntegerLength	69	InterpretedBox (\\!)	74	Information (???)	81
\$MachineEpsilon	69	LetterNumber	74	Names	82
MachinePrecision	70	NumberString	74	OwnValues	82
\$MachinePrecision	70	RemoveDiacritics	75	SymbolName	82
\$MaxPrecision	70	StringContainsQ	75	SymbolQ	82
\$MinPrecision	71	StringQ	75	Symbol	83
Precision	71	StringRepeat	76	ValueQ	83
RealDigits	72	String	76		
		\$SystemCharac-			
		terEncoding	76		

Atomic Primitives

Atomic Primitive

AtomQ

WMA link

`AtomQ[expr]`

returns True if *expr* is an expression which cannot be divided into subexpressions, or False otherwise.

An expression that cannot be divided into subparts is called called an “atom”.

Strings and expressions that produce strings are atoms:

```
>> Map[AtomQ, {"x", "x" <> "y", StringReverse["live"]}]  
{True, True, True}
```

Numeric literals are atoms:

```
>> Map[AtomQ, {2, 2.1, 1/2, 2 + I, 2^^101}]  
{True, True, True, True, True}
```

So are Mathematical Constants:

```
>> Map[AtomQ, {Pi, E, I, Degree}]  
{True, True, True, True}
```

A Symbol not bound to a value is an atom too:

```
>> AtomQ[x]  
True
```

On the other hand, expressions with more than one Part after evaluation, even those resulting in numeric values, aren't atoms:

```
>> AtomQ[2 + Pi]  
False
```

Similarly any compound Expression, even lists of literals, aren't atoms:

```
>> Map[AtomQ, {{}, {1}, {2, 3, 4}}]  
{False, False, False}
```

Note that evaluation or the binding of "x" to an expression is taken into account:

```
>> x = 2 + Pi; AtomQ[x]  
False
```

Again, note that the expression evaluation to a number occurs before AtomQ evaluated:

```
>> AtomQ[2 + 3.1415]  
True
```

Head

WMA link

Head [*expr*]
returns the head of the expression or atom *expr*.

```
>> Head[a * b]  
Times  
>> Head[6]  
Integer  
>> Head[x]  
Symbol
```

Representation of Numbers

Representation of Numbers

Integers and Real numbers with any number of digits, automatically tagging numerical precision when appropriate.

Precision is not "guarded" through the evaluation process. Only integer precision is supported. However, things like N[Pi, 100] should work as expected.

Accuracy

Accuracy (WMA Accuracy)

Accuracy[x]

examines the number of significant digits of *expr* after the decimal point in the number x.

No-

tice that the result could be slightly different than the obtained in WMA, due to differences in the internal representation of the real numbers.

Accuracy of a real number is estimated from its value and its precision:

```
>> Accuracy[3.1416`2]
```

1.50298

Notice that the value is not exactly equal to the obtained in WMA: This is due to the different way in which Precision is handled in SymPy.

Accuracy for exact atoms is *Infinity*:

```
>> Accuracy[1]
```

∞

```
>> Accuracy[A]
```

∞

For Complex numbers, the accuracy is estimated as (minus) the base-10 log of the square root of the squares of the errors on the real and complex parts:

```
>> z=Complex[3.00``2, 4..00``2];
```

```
>> Accuracy[z] == -Log[10, Sqrt[10^(-2 Accuracy[Re[z]])+ 10^(-2 Accuracy[Im[z]])]]
```

True

Accuracy of expressions is given by the minimum accuracy of its elements:

```
>> Accuracy[F[1, Pi, A]]
```

∞

```
>> Accuracy[F[1.3, Pi, A]]
```

15.8406

Accuracy for the value 0 is a fixed-precision Real number:

```
>> 0``2
```

0.00

```
>> Accuracy[0.`2]
```

2.

For 0., the accuracy satisfies:

```
>> Accuracy[0.] == $MachinePrecision - Log[10, $MinMachineNumber]
```

True

In compound expressions, the Accuracy is fixed by the number with the lowest Accuracy:

```
>> Accuracy[{{1, 1.'}, {1.''5, 1.''10}}]
```

5.

See also Precision of section 4.

ExactNumberQ

WMA link

```
ExactNumberQ[expr]  
returns True if expr is an exact number, and False otherwise.
```

```
>> ExactNumberQ[10]  
True  
>> ExactNumberQ[4.0]  
False  
>> ExactNumberQ[n]  
False  
  
ExactNumberQ can be applied to complex numbers:  
>> ExactNumberQ[1 + I]  
True  
>> ExactNumberQ[1 + 1. I]  
False
```

InexactNumberQ

WMA link

```
InexactNumberQ[expr]  
returns True if expr is not an exact number, and False otherwise.
```

```
>> InexactNumberQ[a]  
False  
>> InexactNumberQ[3.0]  
True  
>> InexactNumberQ[2/3]  
False  
  
InexactNumberQ can be applied to complex numbers:
```

```
>> InexactNumberQ[4.0+I]  
True
```

IntegerExponent

WMA link

```
IntegerExponent[n, b]  
gives the highest exponent of b that divides n.
```

```
>> IntegerExponent[16, 2]  
4  
>> IntegerExponent[-510000]  
4  
>> IntegerExponent[10, b]  
IntegerExponent[10, b]
```

IntegerLength

WMA link

```
IntegerLength[x]
gives the number of digits in the base-10 representation of x.
IntegerLength[x, b]
gives the number of base-b digits in x.
```

```
>> IntegerLength[123456]
6
>> IntegerLength[10^10000]
10001
>> IntegerLength[-10^1000]
1001
```

IntegerLength with base 2:

```
>> IntegerLength[8, 2]
4
```

Check that **IntegerLength** is correct for the first 100 powers of 10:

```
>> IntegerLength /@ (10 ^ Range[100]) == Range[2, 101]
True
```

The base must be greater than 1:

```
>> IntegerLength[3, -2]
Base -2 is not an integer greater than 1.
IntegerLength[3, -2]
```

0 is a special case:

```
>> IntegerLength[0]
0
```

\$MachineEpsilon

WMA link

```
$MachineEpsilon
is the distance between 1.0 and the next nearest representable machine-precision number.
```

```
>> $MachineEpsilon
2.22045*^-16
>> x = 1.0 + {0.4, 0.5, 0.6} $MachineEpsilon;
>> x - 1
{0., 0., 2.22045*^-16}
```

MachinePrecision

WMA link

MachinePrecision

represents the precision of machine precision numbers.

```
>> N[MachinePrecision]
15.9546
>> N[MachinePrecision, 30]
15.9545897701910033463281614204
```

\$MachinePrecision

WMA link

\$MachinePrecision

is the number of decimal digits of precision for machine-precision numbers.

```
>> $MachinePrecision
15.9546
```

\$MaxPrecision

WMA link

\$MaxPrecision

represents the maximum number of digits of precision permitted in arbitrary-precision numbers.

```
>> $MaxPrecision
∞
>> $MaxPrecision = 10;
>> N[Pi, 11]
Requested precision 11 is larger than $MaxPrecision. Using current
$MaxPrecision of 10. instead. $MaxPrecision = Infinity specifies
that any precision should be allowed.
3.141592654
```

\$MinPrecision

WMA link

\$MinPrecision

represents the minimum number of digits of precision permitted in arbitrary-precision numbers.

```
>> $MinPrecision
0
>> $MinPrecision = 10;
```

```

>> N[Pi, 9]
Requested precision 9 is smaller than $MinPrecision. Using current
$MinPrecision of 10. instead.
3.141592654

```

Precision

Precision (WMA)

Precision[*expr*]
examines the number of significant digits of *expr*.

Note

that the result could be slightly different than the obtained in WMA, due to differences in the internal representation of the real numbers.

The precision of an exact number, e.g. an Integer, is *Infinity*:

```

>> Precision[1]
∞

```

A fraction is an exact number too, so its Precision is *Infinity*:

```

>> Precision[1/2]
∞

```

Numbers entered in the form *digits'p* are taken to have precision *p*:

```

>> Precision[1.23`10]
10.

```

Precision of a machineprecision number is *MachinePrecision*:

```

>> Precision[0.5]
MachinePrecision

```

In compound expressions, the Precision is fixed by the number with the lowest Precision:

```

>> Precision[{{1, 1.'}, {1.'5, 1.'10}}]
5.

```

For non-zero Real values, it holds in general:

```

Accuracy[z] == Precision[z] + Log[z]
>> (Accuracy[z] == Precision[z] + Log[z])/.z-> 37.‘
True

```

The case of '0.' values is special. Following WMA, in a Machine Real representation, the precision is set to *MachinePrecision*:

```

>> Precision[0.]
MachinePrecision

```

On the other hand, for a Precision Real with fixed accuracy, the precision is evaluated to 0.:

```

>> Precision[0.‘‘3]
0.

```

See also Accuracy of section 4.

RealDigits

WMA link

RealDigits[n]

returns the decimal representation of the real number n as list of digits, together with the number of digits that are to the left of the decimal point.

RealDigits[n, b]

returns a list of base _{b} representation of the real number n .

RealDigits[n, b, len]

returns a list of len digits.

RealDigits[n, b, len, p]

return len digits starting with the coefficient of b^p

Return the list of digits and exponent:

```
>> RealDigits[123.55555]
```

```
{\{1,2,3,5,5,5,5,0,0,0,0,0,0,0,0\},3}
```

Return an explicit recurring decimal form:

```
>> RealDigits[19 / 7]
```

```
{\{2,\{7,1,4,2,8,5\}\},1}
```

The 500th digit of Pi is 2:

```
>> RealDigits[Pi, 10, 1, -500]
```

```
{\{2\}, -499}
```

11 digits starting with the coefficient of 10^{-3} :

```
>> RealDigits[Pi, 10, 11, -3]
```

```
{\{1,5,9,2,6,5,3,5,8,9,7\},-2}
```

RealDigits gives Indeterminate if more digits than the precision are requested:

```
>> RealDigits[123.45, 10, 18]
```

```
{\{1,2,3,4,5,0,0,0,0,0,0,0,0,0,0,Indeterminate,Indeterminate\},3}
```

Return 25 digits of in base 10:

```
>> RealDigits[Pi, 10, 25]
```

```
{\{3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8,4,6,2,6,4,3\},1}
```

String Manipulation

String Manipulation

Alphabet

WMA link

Alphabet[]

gives the list of lowercase letters a-z in the English alphabet .

Alphabet[type]

gives the alphabet for the language or class *type*.

```
>> Alphabet[]
```

```
{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}
```

```
>> Alphabet["German"]
{a, ä, b, c, d, e, f, g, h, i, j, k, l, m, n, o, ö, p, q, r, s, SS, t, u, ü, v, w, x, y, z}
```

Some languages are aliases. "Russian" is the same letter set as "Cyrillic"

```
>> Alphabet["Russian"] == Alphabet["Cyrillic"]
True
```

\$CharacterEncoding

WMA link

\$CharacterEncoding

specifies the default raw character encoding to use for input and output when no encoding is explicitly specified. Initially this is set to \$SystemCharacterEncoding.

See the character encoding current is in effect and used in input and output functions functions like OpenRead[] :

```
>> $CharacterEncoding
ASCII
```

See also \$SystemCharacterEncoding of section 4.

\$CharacterEncodings

WMA link

\$CharacterEncodings

stores the list of available character encodings.

```
>> $CharacterEncodings
{ASCII, CP949, CP950, EUC-JP, IBM-850, ISOLatin1, ISOLatin2, ISOLatin3, ISOLatin4, ISOLatinCyrillic, ISO8859-1,
```

HexadecimalCharacter

WMA link

HexadecimalCharacter

represents the characters 0-9, a-f and A-F.

```
>> StringMatchQ[#, HexadecimalCharacter] & /@ {"a", "1", "A", "x", "H",
" ", ".."}
{True, True, True, False, False, False}
```

InterpretedBox (\!)

WMA link

InterpretedBox [box]

is the ad hoc fullform for
! box. just for internal use...

```
>> \! \((2+2\)
4
```

LetterNumber

WMA link

```
LetterNumber[c]
    returns the position of the character c in the English alphabet.
LetterNumber[{"string"}]
    returns a list of the positions of characters in string.
LetterNumber[{"string"}, alpha]
    returns a list of the positions of characters in string, regarding the alphabet alpha.
```

```
>> LetterNumber["b"]
2
```

LetterNumber also works with uppercase characters

```
>> LetterNumber["B"]
2
>> LetterNumber["ss2!"]
{19, 19, 0, 0}
```

Get positions of each of the letters in a string:

```
>> LetterNumber[Characters["Peccary"]]
{16, 5, 3, 3, 1, 18, 25}
>> LetterNumber[{"P", "Pe", "P1", "eck"}]
{16, {16, 5}, {16, 0}, {5, 3, 11}}
>> LetterNumber["\[Beta]", "Greek"]
2
```

NumberString

WMA link

```
NumberString
    represents the characters in a number.
```

```
>> StringMatchQ["1234", NumberString]
True
>> StringMatchQ["1234.5", NumberString]
True
>> StringMatchQ["1.2'20", NumberString]
False
```

RemoveDiacritics

WMA link

```
RemoveDiacritics[s]
    returns a version of s with all diacritics removed.
```

```
>> RemoveDiacritics["en prononçant pêcher et pécher"]
    en prononcant pecher et pecher

>> RemoveDiacritics["piñata"]
    pinata
```

StringContainsQ

WMA link

```
StringContainsQ["string", patt]
    returns True if any part of string matches patt, and returns False otherwise.
StringContainsQ[{“s1”, “s2”, …}, patt]
    returns the list of results for each element of string list.
StringContainsQ[patt]
    represents an operator form of StringContainsQ that can be applied to an expression.
```

```
>> StringContainsQ["mathics", "m" ~~__ ~~"s"]
    True

>> StringContainsQ["mathics", "a" ~~__ ~~"m"]
    False

>> StringContainsQ["Mathics", "MA" , IgnoreCase -> True]
    True

>> StringContainsQ[{"g", "a", "laxy", "universe", "sun"}, "u"]
    {False, False, False, True, True}

>> StringContainsQ["e" ~~__ ~~"u"] /@ {"The Sun", "Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune"}
    {True, True, True, False, False, False, False, True}
```

StringQ

WMA link

```
StringQ[expr]
    returns True if expr is a String, or False otherwise.
```

```
>> StringQ["abc"]
    True

>> StringQ[1.5]
    False

>> Select[{"12", 1, 3, 5, "yz", x, y}, StringQ]
    {12, yz}
```

StringRepeat

WMA link

```
StringRepeat["string", n]
gives string repeated n times.
StringRepeat["string", n, max]
gives string repeated n times, but not more than max characters.
```

```
>> StringRepeat["abc", 3]
abcabca
>> StringRepeat["abc", 10, 7]
abcabca
```

String

WMA link

```
String
is the head of strings.
```

```
>> Head["abc"]
String
>> "abc"
abc
```

Use InputForm to display quotes around strings:

```
>> InputForm["abc"]
"abc"
```

FullForm also displays quotes:

```
>> FullForm["abc" + 2]
Plus[2, "abc"]
```

\$SystemCharacterEncoding

WMA link

```
$SystemCharacterEncoding
gives the default character encoding of the system.
On startup, the value of environment variable MATHICS_CHARACTER_ENCODING sets this value.
However if that environment variable is not set, set the value is set in Python using
sys.getdefaultencoding().
```

```
>> $SystemCharacterEncoding
ASCII
```

ToExpression

WMA link

```
ToExpression[input]
    interprets a given string as Mathics input.
ToExpression[input, form]
    reads the given input in the specified form.
ToExpression[input, form, h]
    applies the head h to the expression before evaluating it.
```

```
>> ToExpression["1 + 2"]
3
>> ToExpression["{2, 3, 1}", InputForm, Max]
3
>> ToExpression["2 3", InputForm]
6
```

Note that newlines are like semicolons, not blanks. So so the return value is the second-line value.

```
>> ToExpression["2\[NewLine]3"]
3
```

Tostring

[WMA link](#)

```
ToString[expr]
    returns a string representation of expr.
ToString[expr, form]
    returns a string representation of expr in the form form.
```

```
>> ToString[2]
2
>> ToString[2] // InputForm
"2"
>> ToString[a+b]
a + b
>> "U" <> 2
String expected.
U<>2
>> "U" <> ToString[2]
U2
>> ToString[Integrate[f[x],x], TeXForm]
\int f\left[x\right]\,\mathrm{d}x
```

Transliterate

[WMA link](#)

`Transliterate[s]`

transliterates a text in some script into an ASCII string.

ASCII transliteration examples:

- Russian language
- Hiragana

Whitespace

WMA link

`Whitespace`

represents a sequence of whitespace characters.

```
>> StringMatchQ["\r \n", Whitespace]
True

>> StringSplit["a \n b \r\n c d", Whitespace]
{a,b,c,d}

>> StringReplace[" this has leading and trailing whitespace \n ", (
StartOfString ~~Whitespace) | (Whitespace ~~EndOfString) -> ""] <> "
removed" // FullForm
"this has leading and trailing whitespace removed"
```

Symbol Handling

Symbol Handling

Symbolic data. Every symbol has a unique name, exists in a certain context or namespace, and can have a variety of type of values and attributes.

Context

WMA

`Context[symbol]`

yields the name of the context where *symbol* is defined in.

`Context[]`

returns the value of \$Context.

```
>> Context[a]
Global'

>> Context[b'c]
b'

>> InputForm[Context[]]
"Global"
```

Definition

WMA

Definition [*symbol*]

prints as the definitions given for *symbol*. This is in a form that can be stored in a package.

Definition does not print information for ReadProtected symbols. Definition uses InputForm to format values.

```
>> a = 2;  
  
>> Definition[a]  
a = 2  
  
>> f[x_] := x ^ 2  
  
>> g[f] ^:= 2  
  
>> Definition[f]  
f[x_] = x2  
g[f] ^=2
```

Definition of a rather evolved (though meaningless) symbol:

```
>> Attributes[r] := {Orderless}  
  
>> Format[r[args___]] := Infix[{args}, "~"]  
  
>> N[r] := 3.5  
  
>> Default[r, 1] := 2  
  
>> r::msg := "My message"  
  
>> Options[r] := {Opt -> 3}  
  
>> r[arg_, OptionsPattern[r]] := {arg, OptionValue[Opt]}
```

Some usage:

```
>> r[z, x, y]  
x ~ y ~ z  
  
>> N[r]  
3.5  
  
>> r[]  
{2,3}  
  
>> r[5, Opt->7]  
{5,7}
```

Its definition:

```
>> Definition[r]
```

For ReadProtected symbols, Definition just prints attributes, default values and options:

```
>> SetAttributes[r, ReadProtected]  
  
>> Definition[r]
```

This is the same for built-in symbols:

```

>> Definition[Plus]
Attributes [Plus]
= {Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
Default [Plus] = 0

>> Definition[Level]
Attributes [Level] = {Protected}
Options [Level] = {Heads- > False}

```

ReadProtected can be removed, unless the symbol is locked:

```
>> ClearAttributes[r, ReadProtected]
```

Clear clears values:

```
>> Clear[r]
```

```
>> Definition[r]
```

ClearAll clears everything:

```
>> ClearAll[r]
```

```
>> Definition[r]
```

If a symbol is not defined at all, Null is printed:

```
>> Definition[x]
```

Null

DownValues

WMA

`DownValues[symbol]`

gives the list of downvalues associated with *symbol*.

DownValues uses HoldPattern and RuleDelayed to protect the downvalues from being evaluated. Moreover, it has attribute HoldAll to get the specified symbol instead of its value.

```
>> f[x_] := x ^ 2
```

```
>> DownValues[f]
```

Mathics will sort the rules you assign to a symbol according to their specificity. If it cannot decide which rule is more special, the newer one will get higher precedence.

```
>> f[x_Integer] := 2
```

```
>> f[x_Real] := 3
```

```
>> DownValues[f]
```

```
>> f[3]
```

2

```
>> f[3.]
```

3

```
>> f[a]
```

a^2

The default order of patterns can be computed using Sort with PatternsOrderedQ:

```
>> Sort[{x_, x_Integer}, PatternsOrderedQ]
{x_Integer, x_}
```

By assigning values to DownValues, you can override the default ordering:

```
>> DownValues[g] := {g[x_] :> x^2, g[x_Integer] :> x}
>> g[2]
4
```

Fibonacci numbers:

```
>> DownValues[fib] := {fib[0] -> 0, fib[1] -> 1, fib[n_] :> fib[n - 1] +
  fib[n - 2]}
>> fib[5]
5
```

Information (??)

WMA

Information [*symbol*]
Prints information about a *symbol*

Information

does not print information for ReadProtected symbols.

Information uses InputForm to format values.

Names

WMA

Names ["*pattern*"]
returns the list of names matching *pattern*.

```
>> Names["List"]
{List}
```

The wildcard * matches any character:

```
>> Names["List*"]
{List, ListLinePlot, ListLogPlot, ListPlot, ListQ, Listable}
```

The wildcard @ matches only lowercase characters:

```
>> Names["List@"]
{Listable}
```

```
>> x = 5;
```

```
>> Names["Global`*"]
{x}
```

The number of built-in symbols:

```
>> Length[Names["System`*"]]
1222
```

OwnValues

WMA

OwnValues[*symbol*]
gives the list of ownvalue associated with *symbol*.

```
>> x = 3;  
  
>> x = 2;  
  
>> OwnValues[x]  
  
>> x := y  
  
>> OwnValues[x]  
  
>> y = 5;  
  
>> OwnValues[x]  
  
>> Hold[x] /. OwnValues[x]  
Hold[y]  
  
>> Hold[x] /. OwnValues[x] // ReleaseHold  
5
```

SymbolName

WMA

SymbolName[*s*]
returns the name of the symbol *s* (without any leading context name).

```
>> SymbolName[x] // InputForm  
"x"
```

SymbolQ

WMA

SymbolQ[*x*]
is True if *x* is a symbol, or False otherwise.

```
>> SymbolQ[a]  
True  
  
>> SymbolQ[1]  
False  
  
>> SymbolQ[a + b]  
False
```

Symbol

WMA

Symbol
is the head of symbols.

```
>> Head[x]  
Symbol
```

You can use **Symbol** to create symbols from strings:

```
>> Symbol["x"] + Symbol["x"]  
2x
```

ValueQ

WMA

ValueQ [*expr*]
returns True if and only if *expr* is defined.

```
>> ValueQ[x]  
>> x = 1;  
>> ValueQ[x]
```

5. Binary Data

Binary data is a type of data that is represented in the binary, sequences of zeros or ones. Computer-generated information often comes in this form.

Contents

Binary Reading and Writing	Binary Types	86	System-related binary handling	87
Writing	Byte	86	ByteOrdering	87
BinaryRead	Byte Arrays	86	\$ByteOrdering	87
BinaryWrite	ByteArray	87		

Binary Reading and Writing

Binary Reading and Writing

BinaryRead

WMA link

`BinaryRead[stream]`

reads one byte from the stream as an integer from 0 to 255.

`BinaryRead[stream, type]`

reads one object of specified type from the stream.

`BinaryRead[stream, {type1, type2, ...}]`

reads a sequence of objects of specified types.

```
>> strm = OpenWrite[BinaryFormat -> True]
OutputStream [/tmp/tmp756siqpb,3]

>> BinaryWrite[strm, {97, 98, 99}]
OutputStream [/tmp/tmp756siqpb,3]

>> Close[strm];

>> strm = OpenRead[%], BinaryFormat -> True]
InputStream [/tmp/tmp756siqpb,3]

>> BinaryRead[strm, {"Character8", "Character8", "Character8"}]
{a,b,c}

>> DeleteFile[Close[strm]];
```

BinaryWrite

WMA link

```
BinaryWrite[channel, b]
  writes a single byte given as an integer from 0 to 255.
BinaryWrite[channel, {b1, b2, ...}]
  writes a sequence of bytes.
BinaryWrite[channel, "string"]
  writes the raw characters in a string.
BinaryWrite[channel, x, type]
  writes x as the specified type.
BinaryWrite[channel, {x1, x2, ...}, type]
  writes a sequence of objects as the specified type.
BinaryWrite[channel, {x1, x2, ...}, {type1, type2, ...}]
  writes a sequence of objects using a sequence of specified types.
```

```
>> strm = OpenWrite[BinaryFormat -> True]

>> BinaryWrite[strm, {39, 4, 122}]
OutputStream [/tmp/tmpsp15eur6,3]

>> Close[strm];

>> strm = OpenRead[%], BinaryFormat -> True]

>> BinaryRead[strm]
39

>> BinaryRead[strm, "Byte"]
4

>> BinaryRead[strm, "Character8"]
z

>> DeleteFile[Close[strm]];
```

Write a String

```
>> strm = OpenWrite[BinaryFormat -> True]

>> BinaryWrite[strm, "abc123"]
OutputStream [/tmp/tmpul7gpcvf,3]

>> pathname = Close[%]
/tmp/tmpul7gpcvf
```

Read as Bytes

```
>> strm = OpenRead[%], BinaryFormat -> True]

>> BinaryRead[strm, {"Character8", "Character8", "Character8", "Character8", "Character8", "Character8", "Character8"}]
{a,b,c,1,2,3,EndOfFile}

>> pathname = Close[strm]
/tmp/tmpul7gpcvf
```

```

Read as Characters
>> strm = OpenRead[%], BinaryFormat -> True]

>> BinaryRead[strm, {"Byte", "Byte", "Byte", "Byte", "Byte", "Byte", "Byte"}]
{97,98,99,49,50,51,EndOfFile}

>> DeleteFile[Close[strm]];

Write Type
>> strm = OpenWrite[BinaryFormat -> True]

>> BinaryWrite[strm, 97, "Byte"]
OutputStream [/tmp/tmp_628b9ea,3]

>> BinaryWrite[strm, {97, 98, 99}, {"Byte", "Byte", "Byte"}]
OutputStream [/tmp/tmp_628b9ea,3]

>> DeleteFile[Close[%]];

```

Binary Types

Binary Type

Byte

WMA link

Byte
is a data type for Read.

Byte Arrays

Byte Array

ByteArray

WMA link

ByteArray[{*b_1*, *b_2*, ...}]
Represents a sequence of Bytes *b_1*, *b_2*, ...
ByteArray[“string”]
Constructs a byte array where bytes comes from decode a b64-encoded String

```

>> A=ByteArray[{1, 25, 3}]
ByteArray[<3>]

>> A[[2]]
25

```

```

>> Normal[A]
{1,25,3}

>> ToString[A]
ByteArray[<3>]

>> ByteArray["ARkD"]
ByteArray[<3>]

>> B=ByteArray["asy"]
The first argument in Bytearray[asy] should be a B64 encoded string
or a vector of integers.

$Failed

```

System-related binary handling

System-related binary handling

ByteOrdering

WMA link

ByteOrdering
is an option for BinaryRead, BinaryWrite, and related functions that specifies what ordering of bytes should be assumed for your computer system..

```

>> ByteOrdering
-1

```

\$ByteOrdering

WMA link

\$ByteOrdering
returns the native ordering of bytes in binary data on your computer system.

```

>> $ByteOrdering
-1

```

6. Code Compilation

Code compilation allows Mathics functions to be run faster.

When LLVM and Python libraries are available, compilation produces LLVM code.

Contents

Compile	88	CompiledFunction	89
--------------------------	-----------	-----------------------------------	-----------

Compile

WMA link

```
Compile[{x1, x2, ...}, expr]
  Compiles expr assuming each xi is a Real number.
Compile[{{x1, t1} {x2, t2} ...}, expr]
  Compiles assuming each xi matches type ti.
```

Compilation is performed using `llvmlite`, or Python's builtin "compile" function.

```
>> cf = Compile[{x, y}, x + 2 y]

>> cf[2.5, 4.3]
11.1

>> cf = Compile[{{x, _Real}}, Sin[x]]

>> cf[1.4]
0.98545
```

Compile supports basic flow control:

```
>> cf = Compile[{{x, _Real}, {y, _Integer}}, If[x == 0.0 && y <= 0, 0.0,
   Sin[x ^ y] + 1 / Min[x, 0.5]] + 0.5]

>> cf[3.5, 2]
2.18888
```

Loops and variable assignments are supported usinv Python builtin "compile" function:

```
>> Compile[{{a, _Integer}, {b, _Integer}}, While[b != 0, {a, b} = {b,
   Mod[a, b]}]; a] (* GCD of a, b *)
```

CompiledFunction

WMA link

```
CompiledFunction[args...]
represents compiled code for evaluating a compiled function.
```

```
>> sqr = Compile[{x}, x x]
>> Head[sqr]
CompiledFunction
>> sqr[2]
4.
```

7. Colors

Programmatic support for symbolic colors.

Contents

Color Directives	90	Darker	97	LightGreen	107
CMYKColor	90	DominantColors .	98	LightMagenta . . .	107
ColorDistance	91	Lighter	99	LightOrange	108
GrayLevel	91	RGBColor	100	LightPink	108
Hue	92	Named Colors	100	LightPurple	109
LABColor	92	Black	100	LightRed	109
LCHColor	92	Blue	101	LightYellow	110
LUVColor	93	Brown	101	Magenta	110
Opacity	94	Cyan	102	Orange	111
RGBColor	94	Gray	103	Pink	111
XYZColor	94	Green	104	Purple	112
Color Operations	95	LightBlue	105	Red	112
Blend	95	LightBrown	105	White	113
ColorConvert	95	LightCyan	106	Yellow	114
ColorNegate	96	LightGray	106		

Color Directives

Color Directives

There are many different way to specify color, and we support many of these.
We can convert between the different color formats.

CMYKColor

WMA link

`CMYKColor[c, m, y, k]`

represents a color with the specified cyan, magenta, yellow and black components.

```
>> Graphics[MapIndexed[{CMYKColor @@ #1, Disk[2*#2 ~Join~{0}]} &,  
IdentityMatrix[4]], ImageSize->Small]
```



ColorDistance

WMA link

```
ColorDistance[c1, c2]
    returns a measure of color distance between the colors c1 and c2.
ColorDistance[list, c2]
    returns a list of color distances between the colors in list and c2.
```

The option `DistanceFunction` specifies the method used to measure the color distance. Available options are:

- CIE76: Euclidean distance in the `LABColor` space
- CIE94: Euclidean distance in the `LCHColor` space
- CIE2000 or CIEDE2000: CIE94 distance with corrections
- CMC: Color Measurement Committee metric (1984)
- DeltaL: difference in the L component of `LCHColor`
- DeltaC: difference in the C component of `LCHColor`
- DeltaH: difference in the H component of `LCHColor`

It is also possible to specify a custom distance.

```
>> ColorDistance[Magenta, Green]
2.2507

>> ColorDistance[{Red, Blue}, {Green, Yellow}, DistanceFunction -> {"CMC
", "Perceptibility"}]
{1.0495, 1.27455}
```

GrayLevel

WMA link

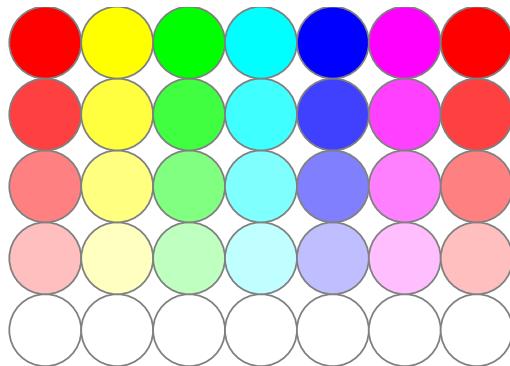
```
GrayLevel[g]
    represents a shade of gray specified by g, ranging from 0 (black) to 1 (white).
GrayLevel[g, a]
    represents a shade of gray specified by g with opacity a.
```

Hue

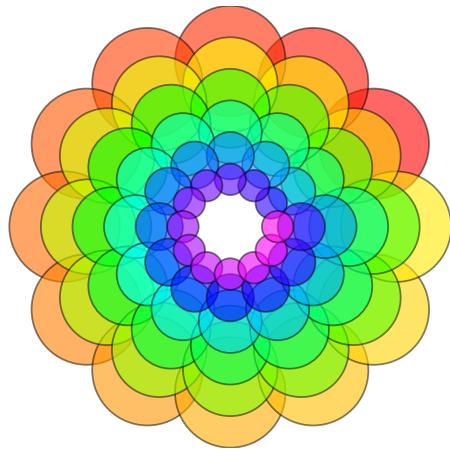
WMA link

```
Hue[h, s, l, a]
    represents the color with hue h, saturation s, lightness l and opacity a.
Hue[h, s, l]
    is equivalent to Hue[h, s, l, 1].
Hue[h, s]
    is equivalent to Hue[h, s, 1, 1].
Hue[h]
    is equivalent to Hue[h, 1, 1, 1].
```

```
>> Graphics[Table[{EdgeForm[Gray], Hue[h, s], Disk[{12h, 8s}]}, {h, 0, 1, 1/6}, {s, 0, 1, 1/4}]]
```



```
>> Graphics[Table[{EdgeForm[{GrayLevel[0, 0.5]}], Hue[(-11+q+10r)/72, 1, 1, 0.6], Disk[(8-r){Cos[2Pi q/12], Sin[2Pi q/12]}, (8-r)/3]}, {r, 6}, {q, 12}]]
```



LABColor

WMA link

`LABColor[l, a, b]`

represents a color with the specified lightness, red/green and yellow/blue components in the CIE 1976 L*a*b* (CIELAB) color space.

LCHColor

WMA link

`LCHColor[l, c, h]`

represents a color with the specified lightness, chroma and hue components in the CIELCh CIELab cube color space.

LUVColor

WMA link

`LCHColor[l, u, v]`

represents a color with the specified components in the CIE 1976 L*u*v* (CIELUV) color space.

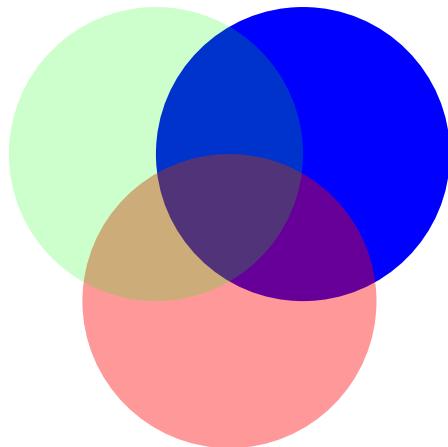
Opacity

WMA link

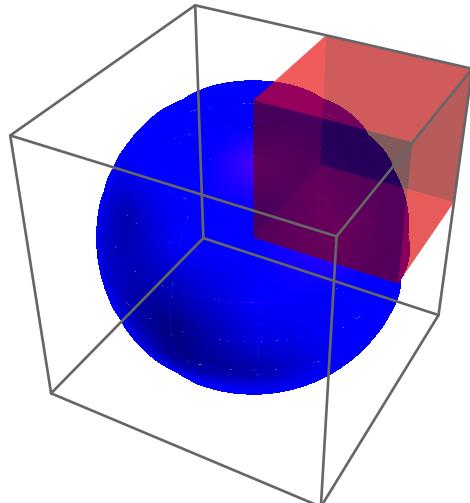
`Opacity[level]`

is a graphics directive that sets the opacity to *level*.

```
>> Graphics[{Blue, Disk[{.5, 1}, 1], Opacity[.4], Red, Disk[], Opacity  
[.2], Green, Disk[{-5, 1}, 1]}]
```

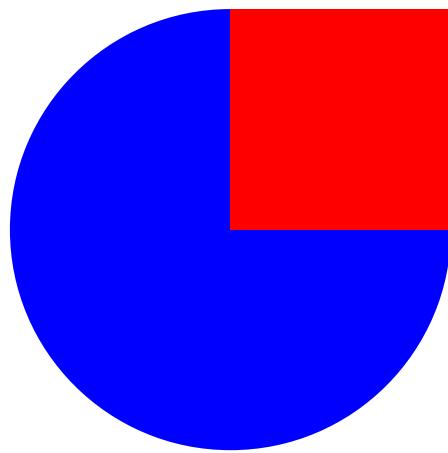


```
>> Graphics3D[{Blue, Sphere[], Opacity[.4], Red, Cuboid[]}]
```



Notice that `Opacity` does not overwrite the value of the alpha channel if it is set in a color directive:

```
>> Graphics[{\{Blue, Disk[], RGBColor[1,0,0,1], Opacity[.2], Rectangle[{0,0},{1,1}]\}}]
```



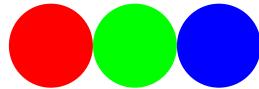
RGBColor

WMA link

```
RGBColor[r, g, b]
```

represents a color with the specified red, green and blue components.

```
>> Graphics[MapIndexed[{RGBColor @@ #1, Disk[2*#2 ~Join~ {0}]} &, IdentityMatrix[3]], ImageSize->Small]
```



```
>> RGBColor[0, 1, 0]
```



```
>> RGBColor[0, 1, 0] // ToBoxes
```

```
StyleBox[GraphicsBox[{\{EdgeForm[RGBColor[0,0,0]],RGBColor[0,1,0],RectangleBox[{0,0}],AspectRatio->Automatic,Axes->False,AxesStyle->{},Background->Automatic,ImageSize->16,LabelStyle->{},PlotRange->Automatic,PlotRangePadding->Automatic,TicksStyle->{}},ImageSizeMultipliers->{1,1},ShowStringCharacters->True]]
```

XYZColor

WMA link

```
XYZColor[x, y, z]
```

represents a color with the specified components in the CIE 1931 XYZ color space.

Color Operations

Color Operations

Functions for manipulating colors and color images.

Blend

WMA link

```
Blend[{c1, c2}]
```

represents the color between $c1$ and $c2$.

```
Blend[{c1, c2}, x]
```

represents the color formed by blending $c1$ and $c2$ with factors $1 - x$ and x respectively.

```
Blend[{c1, c2, ..., cn}, x]
```

blends between the colors $c1$ to cn according to the factor x .

```
>> Blend[{Red, Blue}]
```



```
>> Blend[{Red, Blue}, 0.3]
```



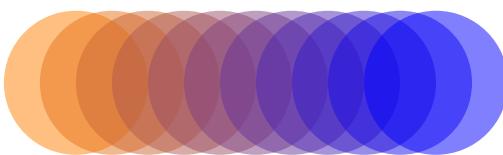
```
>> Blend[{Red, Blue, Green}, 0.75]
```



```
>> Graphics[Table[{Blend[{Red, Green, Blue}, x], Rectangle[{10 x, 0}]}, {x, 0, 1, 1/10}]]
```



```
>> Graphics[Table[{Blend[{RGBColor[1, 0.5, 0, 0.5], RGBColor[0, 0, 1, 0.5]}, x], Disk[{5x, 0}]}, {x, 0, 1, 1/10}]]
```



ColorConvert

WMA link

```
ColorConvert[c, colspace]
```

returns the representation of c in the color space $colspace$. c may be a color or an image.

Valid values for $colspace$ are:

CMYK: convert to CMYKColor
Grayscale: convert to GrayLevel
HSB: convert to Hue
LAB: convert to LABColor
LCH: convert to LCHColor
LUV: convert to LUVColor
RGB: convert to RGBColor
XYZ: convert to XYZColor

ColorNegate

Color Inversion (WMA)

ColorNegate[*color*]

returns the negative of a color, that is, the RGB color subtracted from white.

ColorNegate[*image*]

returns an image where each pixel has its color negated.

Yellow is `RGBColor[1.0, 1.0, 0.0]` So when inverted or subtracted from `White`, we get blue:

```
>> ColorNegate[Yellow] == Blue
```

True

```
>> ColorNegate[Import["ExampleData/sunflowers.jpg"]]
```



Darker

WMA link

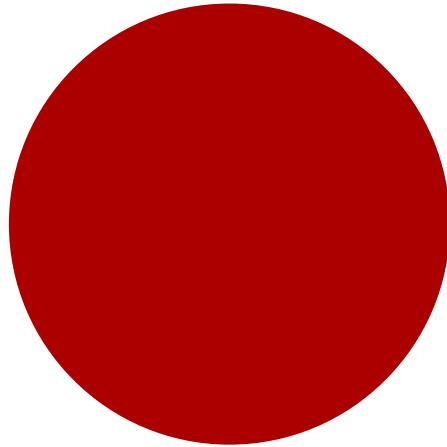
Darker[*c*, *f*]

is equivalent to `Blend[{c, Black}, f]`.

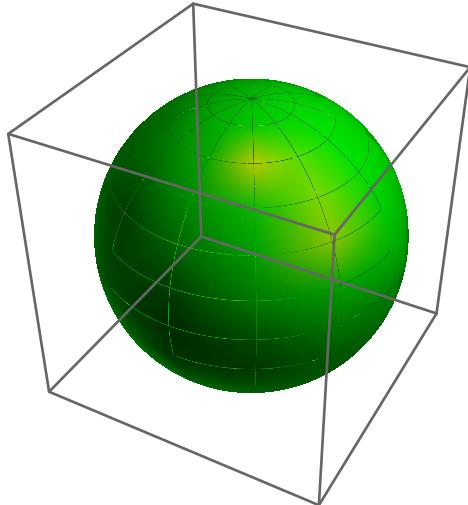
Darker[*c*]

is equivalent to `Darker[c, 1/3]`.

```
>> Graphics[{Darker[Red], Disk[]}]
```



```
>> Graphics3D[{Darker[Green], Sphere[]}]
```



```
>> Graphics[Table[{Darker[Yellow, x], Disk[{12x, 0}]}, {x, 0, 1, 1/6}]]
```



DominantColors

WMA link

```
DominantColors[image]
```

gives a list of colors which are dominant in the given image.

```
DominantColors[image, n]
```

returns at most n colors.

```
DominantColors[image, n, prop]
```

returns the given property $prop$, which may be:

- “Color”: return RGB colors,
- “LABColor”: return LAB colors,
- “Count”: return the number of pixels a dominant color covers,
- “Coverage”: return the fraction of the image a dominant color covers, or
- “CoverageImage”: return a black and white image indicating with white the parts that are covered by a dominant color.

The option “ColorCoverage” specifies the minimum amount of coverage needed to include a dominant color in the result.

The option “MinColorDistance” specifies the distance (in LAB color space) up to which colors are merged and thus regarded as belonging to the same dominant color.

```
>> img = Import["ExampleData/hedy.tif"]
```



```
>> DominantColors[img]
{█, █, █}

>> DominantColors[img, 3]
{█, █, █}

>> DominantColors[img, 3, "Coverage"]
{ $\frac{68817}{103360}, \frac{62249}{516800}, \frac{37953}{516800}$ }

>> DominantColors[img, 3, "CoverageImage"]

>> DominantColors[img, 3, "Count"]
{344085, 62249, 37953}

>> DominantColors[img, 2, "LABColor"]
{LABColor[0.00581591, 0.00207458, -0.00760911], █}

>> DominantColors[img, MinColorDistance -> 0.5]
{█, █}

>> DominantColors[img, ColorCoverage -> 0.15]
{█}
```

Lighter

WMA link

```

Lighter[c, f]
  is equivalent to Blend[{c, White}, f].
Lighter[c]
  is equivalent to Lighter[c, 1/3].

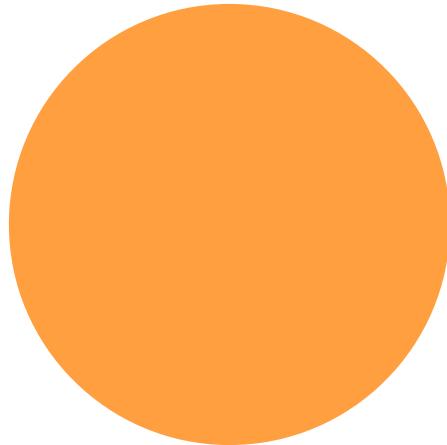
```

```

>> Lighter[Orange, 1/4]

>> Graphics[{Lighter[Orange, 1/4], Disk[]}]


```



```

>> Graphics[Table[{Lighter[Orange, x], Disk[{12x, 0}]}, {x, 0, 1, 1/6}]]


```

RGBColor

[WMA link](#)

```

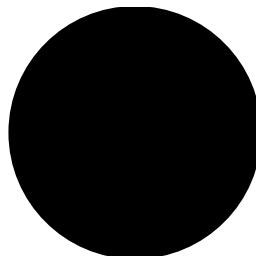
RGBColor[r, g, b]
  represents a color with the specified red, green and blue components.

```

```

>> Graphics[MapIndexed[{RGBColor @@ #1, Disk[2*#2 ~Join~{0}]} &,
  IdentityMatrix[3]], ImageSize->Small]

```



```

>> RGBColor[0, 1, 0]


```

```

>> RGBColor[0, 1, 0] // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0, 1, 0], RectangleBox[{0, 0}]}], AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]

```

Named Colors

Named Colors

Mathics has definitions for the most common color names which can be used in a graphics or style specification.

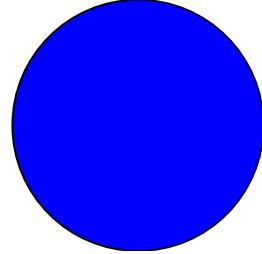
Black

WMA link

Black

represents the color black in graphics.

```
>> Graphics[{EdgeForm[Black], Black, Disk[]}, ImageSize -> Small]
```



```
>> Black // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0, 0, 0], RectangleBox[{0, 0}]}], AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

WMA link

```
>> Black
```



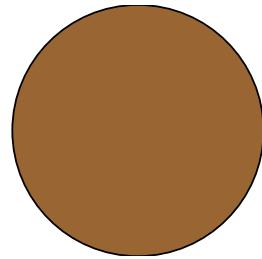
Blue

WMA link

Blue

represents the color blue in graphics.

```
>> Graphics[{EdgeForm[Black], Blue, Disk[]}, ImageSize->Small]
```



```
>> Blue // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0,0,0]],RGBColor[0,0,1],RectangleBox[{0,0}],AspectRatio->Automatic,Axes->False,AxesStyle->{},Background->Automatic,ImageSize->16,LabelStyle->{},PlotRange->Automatic,PlotRangePadding->Automatic,TicksStyle->{}},ImageSizeMultipliers->{1,1},ShowStringCharacters->True]]
```

WMA link

```
>> Blue
```



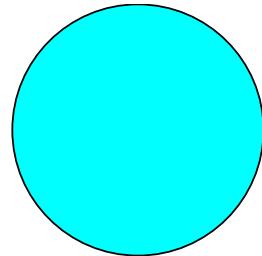
Brown

WMA link

Brown

represents the color brown in graphics.

```
>> Graphics[{EdgeForm[Black], Brown, Disk[]}, ImageSize->Small]
```



```
>> Brown // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0,0,0]],RGBColor[0.6,0.4,0.2],RectangleBox[{0,0}],AspectRatio->Automatic,Axes->False,AxesStyle->{},Background->Automatic,ImageSize->16,LabelStyle->{},PlotRange->Automatic,PlotRangePadding->Automatic,TicksStyle->{}},ImageSizeMultipliers->{1,1},ShowStringCharacters->True]]
```

WMA link

```
>> Brown
```



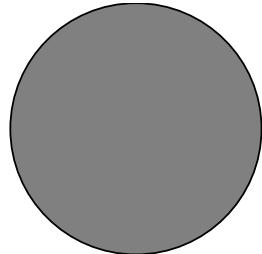
Cyan

WMA link

Cyan

represents the color cyan in graphics.

```
>> Graphics[{EdgeForm[Black], Cyan, Disk[]}, ImageSize->Small]
```



```
>> Cyan // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[0, 1, 1], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes -> False, AxesStyle -> {}, Background -> Automatic, ImageSize -> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}}, ImageSizeMultipliers -> {1, 1}, ShowStringCharacters -> True]]
```

WMA link

```
>> Cyan
```



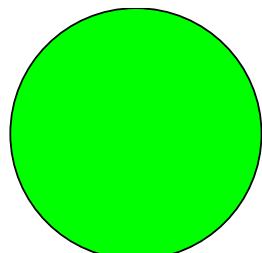
Gray

WMA link

Gray

represents the color gray in graphics.

```
>> Graphics[{EdgeForm[Black], Gray, Disk[]}, ImageSize->Small]
```



```
>> Gray // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], GrayLevel[
0.5], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}}, ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]]
```

WMA link

```
>> Gray
```



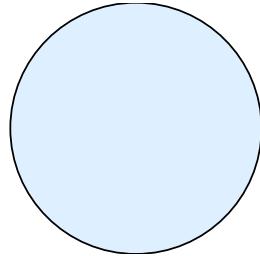
Green

WMA link

Green

represents the color green in graphics.

```
>> Graphics[{EdgeForm[Black], Green, Disk[]}, ImageSize -> Small]
```

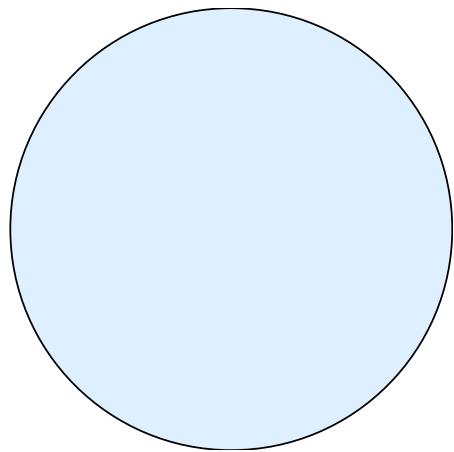


```
>> Green // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0, 1, 0], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}}, ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]]
```

WMA link

>> **Green**



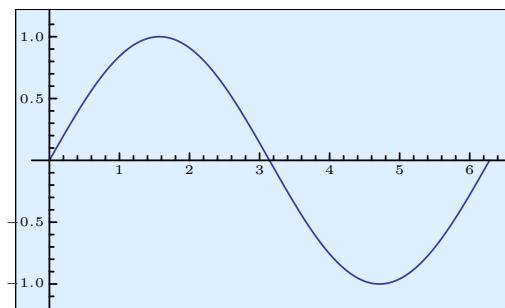
LightBlue

WMA link

LightBlue

represents the color light blue in graphics.

>> `Graphics[{EdgeForm[Black], LightBlue, Disk[]}, ImageSize->Small]`

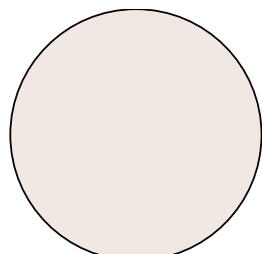


>> `LightBlue // ToBoxes`

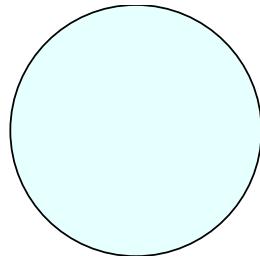
```
StyleBox[GraphicsBox[ {EdgeForm[RGBColor[0,0,0]], RGBColor[
 0.87, 0.94, 1], RectangleBox[ {0,0}], AspectRatio->Automatic, Axes-
 ->False, AxesStyle->{}, Background->Automatic, ImageSize-
 ->16, LabelStyle->{}, PlotRange->Automatic, PlotRangePadding-
 ->Automatic, TicksStyle->{}], ImageSizeMultipliers-
 ->{1,1}, ShowStringCharacters->True]]
```

WMA link

>> `Graphics[{LightBlue, EdgeForm[Black], Disk[]}]`



```
>> Plot[Sin[x], {x, 0, 2 Pi}, Background -> LightBlue]
```



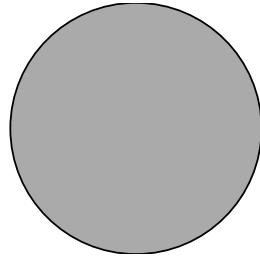
LightBrown

WMA link

LightBrown

represents the color light brown in graphics.

```
>> Graphics[{EdgeForm[Black], LightBrown, Disk[]}, ImageSize->Small]
```



```
>> LightBrown // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[0.94, 0.91, 0.88], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes -> False, AxesStyle -> {}, Background -> Automatic, ImageSize -> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}}, ImageSizeMultipliers -> {1, 1}, ShowStringCharacters -> True]]
```

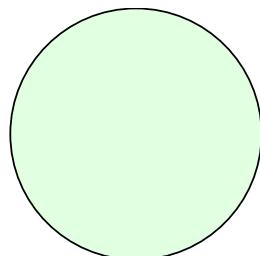
LightCyan

WMA link

LightCyan

represents the color light cyan in graphics.

```
>> Graphics[{EdgeForm[Black], LightCyan, Disk[]}, ImageSize->Small]
```



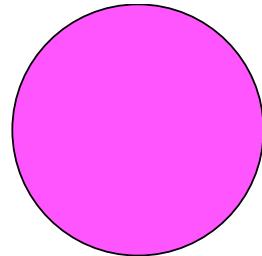
```
>> LightCyan // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0.9, 1., 1.], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightGray

[WMA link](#)

LightGray
represents the color light gray in graphics.

```
>> Graphics[{EdgeForm[Black], LightGray, Disk[]}, ImageSize -> Small]
```



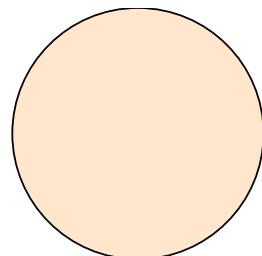
```
>> LightGray // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], GrayLevel[
0.666667, 1.], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightGreen

[WMA link](#)

LightGreen
represents the color light green in graphics.

```
>> Graphics[{EdgeForm[Black], LightGreen, Disk[]}, ImageSize -> Small]
```



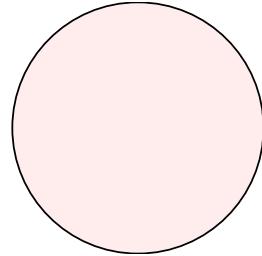
```
>> LightGreen // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0.88, 1., 0.88], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightMagenta

[WMA link](#)

LightMagenta
represents the color light magenta in graphics.

```
>> Graphics[{EdgeForm[Black], LightMagenta, Disk[]}, ImageSize -> Small]
```



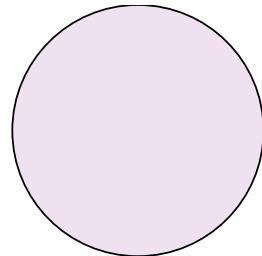
```
>> LightMagenta // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1., 0.333333, 1.], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightOrange

[WMA link](#)

LightOrange
represents the color light orange in graphics.

```
>> Graphics[{EdgeForm[Black], LightOrange, Disk[]}, ImageSize -> Small]
```



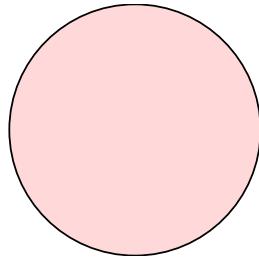
```
>> LightOrange // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1, 0.9, 0.8], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightPink

WMA link

LightPink
represents the color light pink in graphics.

```
>> Graphics[{EdgeForm[Black], LightPink, Disk[]}, ImageSize -> Small]
```



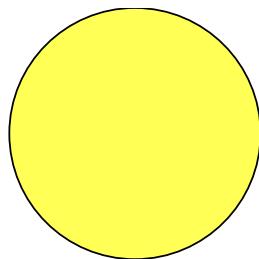
```
>> LightPink // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1., 0.925, 0.925], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightPurple

WMA link

LightPurple
represents the color light purple in graphics.

```
>> Graphics[{EdgeForm[Black], LightPurple, Disk[]}, ImageSize -> Small]
```



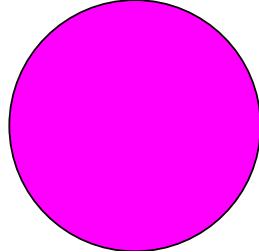
```
>> LightPurple // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0.94, 0.88, 0.94], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightRed

WMA link

LightRed
represents the color light red in graphics.

```
>> Graphics[{EdgeForm[Black], LightRed, Disk[]}, ImageSize -> Small]
```



```
>> LightRed // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1., 0.85, 0.85], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

LightYellow

WMA link

LightYellow
represents the color light yellow in graphics.

```
>> Graphics[{EdgeForm[Black], LightYellow, Disk[]}, ImageSize -> Small]
```



```
>> LightYellow // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1., 1., 0.333333], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

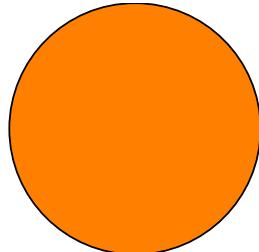
Magenta

WMA link

Magenta

represents the color magenta in graphics.

```
>> Graphics[{EdgeForm[Black], Magenta, Disk[]}, ImageSize->Small]
```

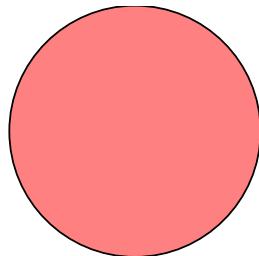


```
>> Magenta // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[1, 0, 1], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes -> False, AxesStyle -> {}, Background -> Automatic, ImageSize -> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}}, ImageSizeMultipliers -> {1, 1}, ShowStringCharacters -> True]]
```

WMA link

```
>> Magenta
```



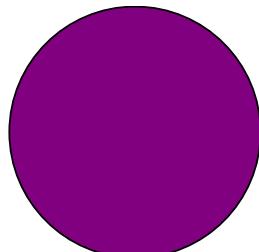
Orange

WMA link

Orange

represents the color orange in graphics.

```
>> Graphics[{EdgeForm[Black], Orange, Disk[]}, ImageSize->Small]
```



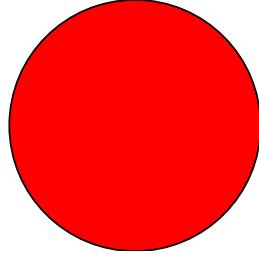
```
>> Orange // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1, 0.5, 0], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

Pink

WMA link

Pink
represents the color pink in graphics.

```
>> Graphics[{EdgeForm[Black], Pink, Disk[]}, ImageSize -> Small]
```



```
>> Pink // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1., 0.5, 0.5], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

Purple

WMA link

Purple
represents the color purple in graphics.

```
>> Graphics[{EdgeForm[Black], Purple, Disk[]}, ImageSize -> Small]
```



```
>> Purple // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0.5, 0, 0.5], RectangleBox[{0, 0}]}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

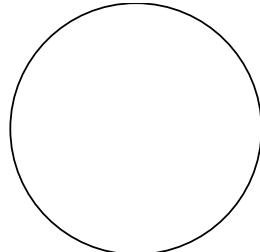
Red

WMA link

Red

represents the color red in graphics.

```
>> Graphics[{EdgeForm[Black], Red, Disk[]}, ImageSize->Small]
```



```
>> Red // ToBoxes
```

```
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[1, 0, 0], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes -> False, AxesStyle -> {}, Background -> Automatic, ImageSize -> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}}, ImageSizeMultipliers -> {1, 1}, ShowStringCharacters -> True]]
```

WMA link

```
>> Red
```



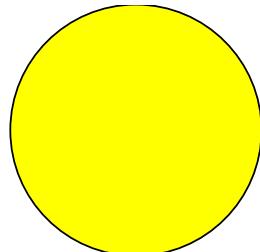
White

WMA link

White

represents the color white in graphics.

```
>> Graphics[{EdgeForm[Black], White, Disk[]}, ImageSize->Small]
```



```

>> White // ToBoxes
StyleBox[GraphicsBox[{{EdgeForm[RGBColor[0, 0, 0]], GrayLevel[
1], RectangleBox[{0, 0}]}}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]

```

WMA link

```
>> White
```



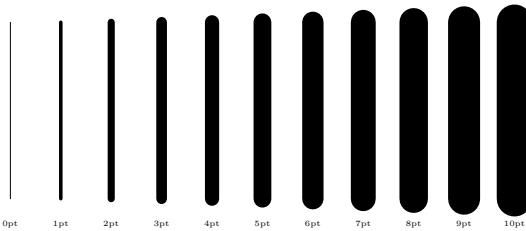
Yellow

WMA link

Yellow

represents the color yellow in graphics.

```
>> Graphics[{EdgeForm[Black], Yellow, Disk[]}, ImageSize -> Small]
```

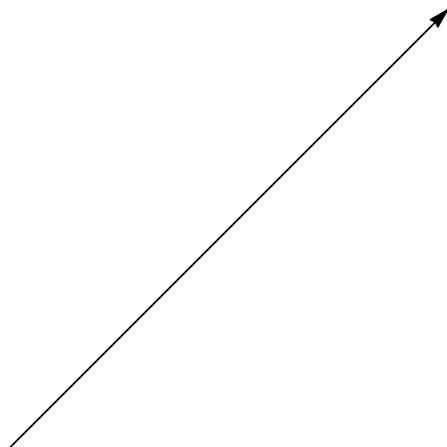


```
>> Yellow // ToBoxes
```

```
StyleBox[GraphicsBox[{{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
1, 1, 0], RectangleBox[{0, 0}]}}, AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}], ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]
```

WMA link

```
>> Yellow
```



8. Date and Time

Dates and times are represented symbolically; computations can be performed on them. Date object can also input and output dates and times in a wide range of formats, as well as handle calendars.

Contents

\$DateStringFormat	114	DateList	117	SessionTime	119
\$SystemTimeZone	114	DateObject	117	TimeConstrained	119
\$TimeZone	115	DatePlus	117	TimeRemaining	120
AbsoluteTime	115	DateString	118	TimeUsed	120
AbsoluteTiming	115	EasterSunday	118	Timing	120
DateDifference	116	Now	119		
		Pause	119		

\$DateStringFormat

WMA link

`$DateStringFormat`
gives the format used for dates generated by `DateString`.

```
>> $DateStringFormat  
{DateTimeShort}
```

\$SystemTimeZone

WMA link

`$SystemTimeZone`
gives the current time zone for the computer system on which Mathics is being run.

```
>> $SystemTimeZone  
-5.
```

\$TimeZone

Time Zone (WMA)

`$TimeZone`
gives the current time zone to assume for dates and times.

```
>> $TimeZone  
-5.
```

AbsoluteTime

WMA link

```
AbsoluteTime[]  
gives the local time in seconds since epoch January 1, 1900, in your time zone.  
AbsoluteTime[{y, m, d, h, m, s}]  
gives the absolute time specification corresponding to a date list.  
AbsoluteTime["string"]  
gives the absolute time specification for a given date string.  
AbsoluteTime[{"string", {e1, e2, ...}}]  
tags the date string to contain the elements "ei".
```

```
>> AbsoluteTime[]  
3.88632*^9  
  
>> AbsoluteTime[{2000}]  
3155673600  
  
>> AbsoluteTime[{"01/02/03", {"Day", "Month", "YearShort"}}]  
3253046400  
  
>> AbsoluteTime["6 June 1991"]  
2885155200  
  
>> AbsoluteTime[{"6-6-91", {"Day", "Month", "YearShort"}}]  
2885155200
```

AbsoluteTiming

WMA link

```
AbsoluteTiming[expr]  
evaluates expr, returning a list of the absolute number of seconds in real time that have elapsed, together with the result obtained.
```

```
>> AbsoluteTiming[50!]  
{0.000150919, 30414093201713378043612608166064768844377641568960512000000000000}  
  
>> Attributes[AbsoluteTiming]  
{HoldAll, Protected}
```

DateDifference

WMA link

```

DateDifference[date1, date2]
    returns the difference between date1 and date2 in days.
DateDifference[date1, date2, unit]
    returns the difference in the specified unit.
DateDifference[date1, date2, {unit1, unit2, ...}]
    represents the difference as a list of integer multiples of each unit, with any remainder expressed in the smallest unit.

```

```

>> DateDifference[{2042, 1, 4}, {2057, 1, 1}]
5476

>> DateDifference[{1936, 8, 14}, {2000, 12, 1}, "Year"]
{64.3425,Year}

>> DateDifference[{2010, 6, 1}, {2015, 1, 1}, "Hour"]
{40200,Hour}

>> DateDifference[{2003, 8, 11}, {2003, 10, 19}, {"Week", "Day"}]
{{9,Week},{6,Day}}

```

DateList

[WMA link](#)

```

DateList[]
    returns the current local time in the form {year, month, day, hour, minute, second}.

DateList[time]
    returns a formatted date for the number of seconds time since epoch Jan 1 1900.

DateList[{y, m, d, h, m, s}]
    converts an incomplete date list to the standard representation.

```

```

>> DateList[0]
{1900,1,1,0,0,0.}

>> DateList[3155673600]
{2000,1,1,0,0,0.}

>> DateList[{2003, 5, 0.5, 0.1, 0.767}]
{2003,4,30,12,6,46.02}

>> DateList[{2012, 1, 300., 10}]
{2012,10,26,10,0,0.}

>> DateList["31/10/1991"]
{1991,10,31,0,0,0.}

>> DateList["1/10/1991"]
The interpretation of 1/10/1991 is ambiguous.
{1991,1,10,0,0,0.}

>> DateList[{"31/10/91", {"Day", "Month", "YearShort"}}]
{1991,10,31,0,0,0.}

```

```
>> DateList[>{"31 10/91", {"Day", " ", "Month", "/", "YearShort"}}]
{1991, 10, 31, 0, 0, 0.}
```

If not specified, the current year assumed

```
>> DateList[{"5/18", {"Month", "Day"}}
{2023, 5, 18, 0, 0, 0.}
```

DateObject

WMA link

DateObject[...]

Returns an object codifiyng DateList....

```
>> DateObject[{2020, 4, 15}]
[Wed 15 Apr 2020 00:00:00 GMT -5]
```

DatePlus

WMA link

DatePlus[*date*, *n*]

finds the date *n* days after *date*.

DatePlus[*date*, {*n*, "unit"}]

finds the date *n* units after *date*.

DatePlus[*date*, {{*n*₁, "unit"₁}, {*n*₂, "unit"₂}, ...}]

finds the date which is *n_i* specified units after *date*.

DatePlus[*n*]

finds the date *n* days after the current date.

DatePlus[*offset*]

finds the date which is offset from the current date.

Add 73 days to Feb 5, 2010:

```
>> DatePlus[{2010, 2, 5}, 73]
{2010, 4, 19}
```

Add 8 weeks and 1 day to March 16, 1999:

```
>> DatePlus[{2010, 2, 5}, {{8, "Week"}, {1, "Day"}}]
{2010, 4, 3}
```

DateString

WMA link

```

DateString[]
    returns the current local time and date as a string.
DateString[elem]
    returns the time formatted according to elems.
DateString[{e1, e2, ...}]
    concatenates the time formatted according to elements ei.
DateString[time]
    returns the date string of an AbsoluteTime.
DateString[{y, m, d, h, m, s}]
    returns the date string of a date list specification.
DateString[string]
    returns the formatted date string of a date string specification.
DateString[spec, elems]
    formats the time in turns of elems. Both spec and elems can take any of the above formats.

```

The current date and time:

```

>> DateString[];

>> DateString[{1991, 10, 31, 0, 0}, {"Day", " ", "MonthName", " ", "Year"}]
    31 October 1991

>> DateString[{2007, 4, 15, 0}]
    Sun 15 Apr 2007 00:00:00

>> DateString[{1979, 3, 14}, {"DayName", " ", "Month", "-", "YearShort"}]
    Wednesday 03-79

```

Non-integer values are accepted too:

```

>> DateString[{1991, 6, 6.5}]
    Thu 6 Jun 1991 12:00:00

```

EasterSunday

Date of Easter (WMA link)

```

EasterSunday[year]
    returns the date of the Gregorian Easter Sunday as {year, month, day}.

```

```

>> EasterSunday[2000]
    {2000,4,23}

>> EasterSunday[2030]
    {2030,4,21}

```

Now

WMA link

Now
gives the current time on the system.

```
>> Now  
[Sat 25 Feb 2023 13:25:34 GMT - 5]
```

Pause

WMA link

Pause[n]
pauses for n seconds.

```
>> Pause[0.5]
```

SessionTime

WMA link

SessionTime[]
returns the total time in seconds since this session started.

```
>> SessionTime[]  
113.223
```

TimeConstrained

WMA link

TimeConstrained[expr, t]
evaluates $expr$, stopping after t seconds.
TimeConstrained[expr, t, failexpr]
returns $failexpr$ if the time constraint is not met.

Possible issues: for certain time-consuming functions (like `simplify`) which are based on `sympy` or other libraries, it is possible that the evaluation continues after the timeout. However, at the end of the evaluation, the function will return `$Aborted` and the results will not affect the state of the `Mathics3` kernel.

TimeRemaining

WMA link

TimeRemaining[]
Gives the number of seconds remaining until the earliest enclosing `TimeConstrained` will request the current computation to stop.
TimeConstrained[expr, t, failexpr]
returns $failexpr$ if the time constraint is not met.

If TimeConstrained is called out of a TimeConstrained expression, returns 'Infinity'

```
>> TimeRemaining[]  
∞  
  
>> TimeConstrained[1+2; Print[TimeRemaining[], 0.9]  
0.899139
```

TimeUsed

WMA link

TimeUsed[]
returns the total CPU time used for this session, in seconds.

```
>> TimeUsed[]  
116.248
```

Timing

WMA link

Timing[*expr*]
measures the processor time taken to evaluate *expr*. It returns a list containing the measured time in seconds and the result of the evaluation.

```
>> Timing[50!]  
{0.00018476, 30414093201713378043612608166064768844377641568960512000000000000}  
  
>> Attributes[Timing]  
{HoldAll, Protected}
```

9. Definition Attributes

While a definition like `cube[x_] = x^3` gives a way to specify *values* of a function, *attributes* allow a way to specify general properties of functions and symbols. This is independent of the parameters they take and the values they produce.

The builtin-attributes having a predefined meaning in *Mathics3* which are described below. However in contrast to *Mathematica®*, you can set any symbol as an attribute.

Contents

Attributes	122	HoldRest	124	Orderless	127
ClearAttributes	122	Listable	124	Protect	127
Constant	122	Locked	125	Protected	128
Flat	123	NHoldAll	125	ReadProtected	128
HoldAll	123	NHoldFirst	125	SequenceHold	129
HoldAllComplete . . .	123	NHoldRest	125	SetAttributes	129
HoldFirst	124	NumericFunction . . .	126	Unprotect	129
		OneIdentity	126		

Attributes

[WMA link](#)

Attributes[*symbol*]
returns the attributes of *symbol*.

Attributes["*string*"]
returns the attributes of *Symbol["string"]*.

Attributes[*symbol*] = {*attr1*, *attr2*}
sets the attributes of *symbol*, replacing any existing attributes.

```
>> Attributes[Plus]
{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}

>> Attributes["Plus"]
{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
```

Attributes always considers the head of an expression:

```
>> Attributes[a + b + c]
{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
```

You can assign values to Attributes to set attributes:

```
>> Attributes[f] = {Flat, Orderless}
{Flat, Orderless}

>> f[b, f[a, c]]
f[a, b, c]
```

Attributes must be symbols:

```
>> Attributes[f] := {a + b}
    Argument a + b at position 1 is expected to be a symbol.
$Failed
```

Use Symbol to convert strings to symbols:

```
>> Attributes[f] = Symbol["Listable"]
Listable
>> Attributes[f]
{Listable}
```

ClearAttributes

WMA link

```
ClearAttributes[symbol, attrib]
removes attrib from symbol's attributes.
```

```
>> SetAttributes[f, Flat]
>> Attributes[f]
{Flat}
>> ClearAttributes[f, Flat]
>> Attributes[f]
{}
```

Attributes that are not even set are simply ignored:

```
>> ClearAttributes[{f}, {Flat}]
>> Attributes[f]
{}
```

Constant

WMA link

```
Constant
is an attribute that indicates that a symbol is a constant.
```

Mathematical constants like E have attribute Constant :

```
>> Attributes[E]
{Constant, Protected, ReadProtected}
```

Constant symbols cannot be used as variables in Solve and related functions:

```
>> Solve[x + E == 0, E]
E is not a valid variable.
Solve[x + E==0, E]
```

Flat

WMA link

Flat

is an attribute that specifies that nested occurrences of a function should be automatically flattened.

A symbol with the Flat attribute represents an associative mathematical operation:

```
>> SetAttributes[f, Flat]
```

```
>> f[a, f[b, c]]  
f[a, b, c]
```

Flat is taken into account in pattern matching:

```
>> f[a, b, c] /. f[a, b] -> d  
f[d, c]
```

HoldAll

WMA link

HoldAll

is an attribute specifying that all arguments of a function should be left unevaluated.

```
>> Attributes[Function]  
{HoldAll, Protected}
```

HoldAllComplete

WMA link

HoldAllComplete

is an attribute that includes the effects of HoldAll and SequenceHold, and also protects the function from being affected by the upvalues of any arguments.

HoldAllComplete even prevents upvalues from being used, and includes SequenceHold.

```
>> SetAttributes[f, HoldAllComplete]
```

```
>> f[a] ^= 3;  
  
>> f[a]  
f[a]  
  
>> f[Sequence[a, b]]  
f[Sequence[a, b]]
```

HoldFirst

WMA link

HoldFirst

is an attribute specifying that the first argument of a function should be left unevaluated.

```
>> Attributes[Set]
{HoldFirst, Protected, SequenceHold}
```

HoldRest

WMA link

HoldRest

is an attribute specifying that all but the first argument of a function should be left unevaluated.

```
>> Attributes[If]
{HoldRest, Protected}
```

Listable

WMA link

Listable

is an attribute specifying that a function should be automatically applied to each element of a list.

```
>> SetAttributes[f, Listable]
>> f[{1, 2, 3}, {4, 5, 6}]
{f[1,4], f[2,5], f[3,6]}
>> f[{1, 2, 3}, 4]
{f[1,4], f[2,4], f[3,4]}
>> {{1, 2}, {3, 4}} + {5, 6}
{{6,7}, {9,10}}
```

Locked

WMA link

Locked

is an attribute that prevents attributes on a symbol from being modified.

The attributes of Locked symbols cannot be modified:

```
>> Attributes[lock] = {Flat, Locked};

>> SetAttributes[lock, {}]
Symbol lock is locked.

>> ClearAttributes[lock, Flat]
Symbol lock is locked.
```

```
>> Attributes[lock] = {}
   Symbol lock is locked.
{ }

>> Attributes[lock]
{Flat, Locked}
```

However, their values might be modified (as long as they are not Protected too):

```
>> lock = 3
3
```

NHoldAll

WMA link

NHoldAll

is an attribute that protects all arguments of a function from numeric evaluation.

```
>> N[f[2, 3]]
f[2.,3.]
>> SetAttributes[f, NHoldAll]
>> N[f[2, 3]]
f[2,3]
```

NHoldFirst

WMA link

NHoldFirst

is an attribute that protects the first argument of a function from numeric evaluation.

NHoldRest

WMA link

NHoldRest

is an attribute that protects all but the first argument of a function from numeric evaluation.

NumericFunction

WMA link

NumericFunction

is an attribute that indicates that a symbol is the head of a numeric function.

Mathematical functions like Sqrt have attribute NumericFunction:

```
>> Attributes[Sqrt]
{Listable, NumericFunction, Protected}
```

Expressions with a head having this attribute, and with all the elements being numeric expressions, are considered numeric expressions:

```
>> NumericQ[Sqrt[1]]
True

>> NumericQ[a]=True; NumericQ[Sqrt[a]]
True

>> NumericQ[a]=False; NumericQ[Sqrt[a]]
False
```

OneIdentity

WMA link

OneIdentity

is an attribute assigned to a symbol, say f , indicating that $f[x]$, $f[f[x]]$, etc. are all equivalent to x in pattern matching.

```
>> a /. f[x_:0, u_] -> {u}
a
```

Here is how OneIdentity changes the pattern matched above :

```
>> SetAttributes[f, OneIdentity]

>> a /. f[x_:0, u_] -> {u}
{a}
```

However, without a default argument, the pattern does not match:

```
>> a /. f[u_] -> {u}
a
```

OneIdentity does not change evaluation:

```
>> f[a]
f[a]
```

Orderless

WMA link

Orderless

is an attribute that can be assigned to a symbol f to indicate that the elements e_i in expressions of the form $f[e_1, e_2, \dots]$ should automatically be sorted into canonical order. This property is accounted for in pattern matching.

The elements of an Orderless function are automatically sorted:

```
>> SetAttributes[f, Orderless]
```

```
>> f[c, a, b, a + b, 3, 1.0]
f[1.,3,a,b,c,a+b]
```

A symbol with the `Orderless` attribute represents a commutative mathematical operation.

```
>> f[a, b] == f[b, a]
True
```

`Orderless` affects pattern matching:

```
>> SetAttributes[f, Flat]
>> f[a, b, c] /. f[a, c] -> d
f[b,d]
```

Protect

WMA link

```
Protect[s1, s2, ...]
sets the attribute Protected for the symbols si.
Protect[str1, str2, ...]
protects all symbols whose names textually match stri.
```

```
>> A = {1, 2, 3};
>> Protect[A]
>> A[[2]] = 4;
Symbol A is Protected.
>> A
{1,2,3}
```

Protected

WMA link

```
Protected
is an attribute that prevents values on a symbol from being modified.
```

Values of `Protected` symbols cannot be modified:

```
>> Attributes[p] = {Protected};
>> p = 2;
Symbol p is Protected.
>> f[p] ^= 3;
Tag p in f[p] is Protected.
>> Format[p] = "text";
Symbol p is Protected.
```

However, attributes might still be set:

```
>> SetAttributes[p, Flat]
```

```
>> Attributes[p]
{Flat, Protected}
```

Thus, you can easily remove the attribute Protected:

```
>> Attributes[p] = {};
```

```
>> p = 2
2
```

You can also use Protect or Unprotect, resp.

```
>> Protect[p]
```

```
>> Attributes[p]
{Protected}
```

```
>> Unprotect[p]
```

If a symbol is Protected and Locked, it can never be changed again:

```
>> SetAttributes[p, {Protected, Locked}]
```

```
>> p = 2
Symbol p is Protected.
2
```

```
>> Unprotect[p]
Symbol p is locked.
```

ReadProtected

WMA link

ReadProtected

is an attribute that prevents values on a symbol from being read.

Values associated with ReadProtected symbols cannot be seen in Definition:

```
>> ClearAll[p]
```

```
>> p = 3;
```

```
>> Definition[p]
```

$p = 3$

```
>> SetAttributes[p, ReadProtected]
```

```
>> Definition[p]
```

Attributes [p] = {ReadProtected}

SequenceHold

WMA link

SequenceHold

is an attribute that prevents Sequence objects from being spliced into a function's arguments.

Normally, Sequence will be spliced into a function:

```
>> f[Sequence[a, b]]  
f[a, b]
```

It does not for SequenceHold functions:

```
>> SetAttributes[f, SequenceHold]  
  
>> f[Sequence[a, b]]  
f[Sequence[a, b]]
```

E.g., Set has attribute SequenceHold to allow assignment of sequences to variables:

```
>> s = Sequence[a, b];  
  
>> s  
Sequence[a, b]  
  
>> Plus[s]  
a + b
```

SetAttributes

WMA link

SetAttributes[symbol, attrib]

adds *attrib* to the list of *symbol*'s attributes.

```
>> SetAttributes[f, Flat]  
  
>> Attributes[f]  
{Flat}
```

Multiple attributes can be set at the same time using lists:

```
>> SetAttributes[{f, g}, {Flat, Orderless}]  
  
>> Attributes[g]  
{Flat, Orderless}
```

Unprotect

WMA link

Unprotect[s1, s2, ...]

removes the attribute Protected for the symbols *si*.

Unprotect[str]

unprotects symbols whose names textually match *str*.

10. Descriptive Statistics

Function which operate on explicit data and symbolic representations of statistical distributions.

Contents

Dependency and Dispursion Statistics	Location Statistics	131	Sort	133	
Correlation	Mean	131	TakeLargest	134	
Covariance	Order Statistics	TakeSmallest	134	
General Statistics	Quantile	132	Shape Statistics	134
CentralMoment	Quartiles	132	Kurtosis	134	
	RankedMax	132	Skewness	134	
	RankedMin	133			

Dependency and Dispursion Statistics

Dependency and Dispursion Statistic

Correlation

Pearson correlation coefficient (WMA)

`Correlation[a, b]`

computes Pearson's correlation of two equal-sized vectors a and b .

An example from Wikipedia:

```
>> Correlation[{10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5}, {8.04, 6.95,
    7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68}]
0.816421
```

Covariance

Covariance (WMA)

`Covariance[a, b]`

computes the covariance between the equal-sized vectors a and b .

```
>> Covariance[{0.2, 0.3, 0.1}, {0.3, 0.3, -0.2}]
0.025
```

General Statistics

General Statistic

CentralMoment

Central moment (WMA)

`CentralMoment [list, r]`

gives the the r th central moment (i.e. the r th moment about the mean) of *list*.

```
>> CentralMoment[{1.1, 1.2, 1.4, 2.1, 2.4}, 4]  
0.100845
```

Location Statistics

Location Statistic

Mean

WMA link

`Mean [list]`

returns the statistical mean of *list*.

```
>> Mean[{26, 64, 36}]  
42  
>> Mean[{1, 1, 2, 3, 5, 8}]  

$$\frac{10}{3}$$
  
>> Mean[{a, b}]  

$$\frac{a+b}{2}$$

```

Order Statistics

Order Statistics

In statistics, an order statistic gives the k -th smallest value.

Together with rank statistics these are fundamental tools in non-parametric statistics and inference.

Important special cases of order statistics are finding minimum and maximum value of a sample and sample quantiles.

Quantile

Quantile (WMA)

In statistics and probability, quantiles are cut points dividing the range of a probability distribution into continuous intervals with equal probabilities, or dividing the observations in a sample in the same way. Quantile is also known as value at risk (VaR) or fractile.

Quantile[*list*, *q*]
 returns the *q*th quantile of *list*.
Quantile[*list*, *q*, {{*a*,*b*}, {*c*,*d*}]}
 uses the quantile definition specified by parameters *a*, *b*, *c*, *d*.
 For a list of length *n*, Quantile[*list*, *q*, {{*a*,*b*}, {*c*,*d*}}] depends on $x=a+(n+b)q$.
 If *x* is an integer, the result is *s*[*x*], where *s*=Sort[list,Less].
 Otherwise, the result is $s[[Floor[x]]]+(s[[Ceiling[x]]]-s[[Floor[x]]])(c+dFractionalPart[x])$, with the indices taken to be 1 or *n* if they are out of range.
 The default choice of parameters is {{0,0},{1,0}}.

Common choices of parameters include:

- {{0, 0}, {1, 0}} inverse empirical CDF (default)
- {{0, 0}, {0, 1}} linear interpolation (California method)

Quantile[*list*,*q*] always gives a result equal to an element of *list*.

```
>> Quantile[Range[11], 1/3]
4
>> Quantile[Range[16], 1/4]
4
>> Quantile[{1, 2, 3, 4, 5, 6, 7}, {1/4, 3/4}]
{2,6}
```

Quartiles

Quartile (WMA)

Quartiles[*list*]
 returns the 1/4, 1/2, and 3/4 quantiles of *list*.

```
>> Quartiles[Range[25]]
{27/4, 13, 77/4}
```

RankedMax

WMA link

RankedMax[*list*, *n*]
 returns the *n*th largest element of *list* (with *n* = 1 yielding the largest element, *n* = 2 yielding the second largest element, and so on).

```
>> RankedMax[{482, 17, 181, -12}, 2]
181
```

RankedMin

WMA link

RankedMin[*list*, *n*]

returns the *n*th smallest element of *list* (with *n* = 1 yielding the smallest element, *n* = 2 yielding the second smallest element, and so on).

```
>> RankedMin[{482, 17, 181, -12}, 2]
17
```

Sort

WMA link

Sort[*list*]

sorts *list* (or the elements of any other expression) according to canonical ordering.

Sort[*list*, *p*]

sorts using *p* to determine the order of two elements.

```
>> Sort[{4, 1.0, a, 3+I}]
{1., 3 + I, 4, a}
```

Sort uses `OrderedQ` to determine ordering by default. You can sort patterns according to their precedence using `PatternsOrderedQ`:

```
>> Sort[{items___, item_, OptionsPattern[], item_symbol, item_?test},
  PatternsOrderedQ]
{item_symbol, item_?test, item_, items___, OptionsPattern[]}
```

When sorting patterns, values of atoms do not matter:

```
>> Sort[{a, b/;t}, PatternsOrderedQ]
{b/;t, a}

>> Sort[{2+c_, 1+b__}, PatternsOrderedQ]
{2 + c_, 1 + b__}

>> Sort[{x_ + n_*y_, x_ + y_}, PatternsOrderedQ]
{x_ + n_y_, x_ + y_}
```

TakeLargest

WMA link

TakeLargest[*list*, *f*, *n*]

returns the a sorted list of the *n* largest items in *list*.

```
>> TakeLargest[{100, -1, 50, 10}, 2]
{100, 50}
```

None, Null, Indeterminate and expressions with head Missing are ignored by default:

```
>> TakeLargest[{-8, 150, Missing[abc]}, 2]
{150, -8}
```

You may specify which items are ignored using the option `ExcludedForms`:

```
>> TakeLargest[{-8, 150, Missing[abc]}, 2, ExcludedForms -> {}]
{Missing[abc], 150}
```

TakeSmallest

WMA link

```
TakeSmallest[list, n]
returns the a sorted list of the n smallest items in list.
```

For details on how to use the ExcludedForms option, see TakeLargest[].

```
>> TakeSmallest[{100, -1, 50, 10}, 2]
{-1, 10}
```

Shape Statistics

Shape Statistic

Kurtosis

Kurtosis (WMA)

```
Kurtosis[list]
gives the Pearson measure of kurtosis for list (a measure of existing outliers).
```

```
>> Kurtosis[{1.1, 1.2, 1.4, 2.1, 2.4}]
1.42098
```

Skewness

Skewness (WMA)

```
Skewness[list]
gives Pearson's moment coefficient of skewness for list (a measure for estimating the symmetry of a distribution).
```

```
>> Skewness[{1.1, 1.2, 1.4, 2.1, 2.4}]
0.407041
```

11. Distance and Similarity Measures

Different measures of distance or similarity for different types of analysis.

Contents

Cluster Analysis	135	ChessboardDistance	138	String Distances and Similarity Measures	139
ClusteringComponents	135	CosineDistance . .	138	DamerauLevenshteinDistance	140
FindClusters	136	EuclideanDistance	139	EditDistance	140
Nearest	137	ManhattanDistance	139	HammingDistance	141
Numerical Data	137	SquaredEuclideanDistance	139		
BrayCurtisDistance	137				
CanberraDistance	138				

Cluster Analysis

Cluster Analysis

ClusteringComponents

WMA link

`ClusteringComponents[list]`

forms clusters from *list* and returns a list of cluster indices, in which each element shows the index of the cluster in which the corresponding element in *list* ended up.

`ClusteringComponents[list, k]`

forms *k* clusters from *list* and returns a list of cluster indices, in which each element shows the index of the cluster in which the corresponding element in *list* ended up.

For more detailed documentation regarding options and behavior, see `FindClusters[]`.

```
>> ClusteringComponents[{1, 2, 3, 1, 2, 10, 100}]
{1,1,1,1,1,2}

>> ClusteringComponents[{10, 100, 20}, Method -> "KMeans"]
{1,0,1}
```

FindClusters

WMA link

`FindClusters[list]`

returns a list of clusters formed from the elements of *list*. The number of cluster is determined automatically.

`FindClusters[list, k]`

returns a list of *k* clusters formed from the elements of *list*.

```

>> FindClusters[{1, 2, 20, 10, 11, 40, 19, 42}]
{{1,2,20,10,11,19},{40,42}}

>> FindClusters[{25, 100, 17, 20}]
{{25,17,20},{100} }

>> FindClusters[{3, 6, 1, 100, 20, 5, 25, 17, -10, 2}]
{{3,6,1,5,-10,2},{100},{20,25,17} }

>> FindClusters[{1, 2, 10, 11, 20, 21}]
{{1,2},{10,11},{20,21} }

>> FindClusters[{1, 2, 10, 11, 20, 21}, 2]
{{1,2,10,11},{20,21} }

>> FindClusters[{1 -> a, 2 -> b, 10 -> c}]
{{a,b},{c} }

>> FindClusters[{1, 2, 5} -> {a, b, c}]
{{a,b},{c} }

>> FindClusters[{1, 2, 3, 1, 2, 10, 100}, Method -> "Agglomerate"]
{{1,2,3,1,2,10},{100} }

>> FindClusters[{1, 2, 3, 10, 17, 18}, Method -> "Agglomerate"]
{{1,2,3},{10},{17,18} }

>> FindClusters[{{1}, {5, 6}, {7}, {2, 4}}, DistanceFunction -> (Abs[
Length[#1] - Length[#2]] &)]
{{{1},{7}},{ {5,6},{2,4}}}

>> FindClusters[{"meep", "heap", "deep", "weep", "sheep", "leap", "keep
"}, 3]
{{meep,deep,weep,keep},{heap,leap},{sheep}}

```

FindClusters' automatic distance function detection supports scalars, numeric tensors, boolean vectors and strings.

The Method option must be either "Agglomerate" or "Optimize". If not specified, it defaults to "Optimize". Note that the Agglomerate and Optimize methods usually produce different clusterings.

The runtime of the Agglomerate method is quadratic in the number of clustered points n, builds the clustering from the bottom up, and is exact (no element of randomness). The Optimize method's runtime is linear in n, Optimize builds the clustering from top down, and uses random sampling.

Nearest

[WMA link](#)

```

Nearest[list, x]
    returns the one item in list that is nearest to x.
Nearest[list, x, n]
    returns the n nearest items.
Nearest[list, x, {n, r}]
    returns up to n nearest items that are not farther from x than r.
Nearest[{p1 → q1, p2 → q2, ...}, x]
    returns q1, q2, ... but measures the distances using p1, p2, ...
Nearest[{p1, p2, ...} → {q1, q2, ...}, x]
    returns q1, q2, ... but measures the distances using p1, p2, ...

```

```

>> Nearest[{5, 2.5, 10, 11, 15, 8.5, 14}, 12]
{11}

```

Return all items within a distance of 5:

```

>> Nearest[{5, 2.5, 10, 11, 15, 8.5, 14}, 12, {All, 5}]
{11,10,14}

>> Nearest[{Blue → "blue", White → "white", Red → "red", Green → "green"}, {Orange, Gray}]
{{red},{white}}

>> Nearest[{{0, 1}, {1, 2}, {2, 3}} → {a, b, c}, {1.1, 2}]
{b}

```

Numerical Data

Numerical Data

BrayCurtisDistance

Bray-Curtis Dissimilarity (WMA)

```

BrayCurtisDistance[u, v]
    returns the Bray-Curtis distance between u and v.

```

The Bray-Curtis distance is equivalent to $\text{Total}[\text{Abs}[u-v]]/\text{Total}[\text{Abs}[u+v]]$.

```

>> BrayCurtisDistance[-7, 5]
6

>> BrayCurtisDistance[{-1, -1}, {10, 10}]
11
9

```

CanberraDistance

Canberra distance (WMA)

```

CanberraDistance[u, v]
    returns the canberra distance between u and v, which is a weighted version of the Manhattan distance.

```

```

>> CanberraDistance[-7, 5]
1
>> CanberraDistance[{-1, -1}, {1, 1}]
2

```

ChessboardDistance

Chebyshev distance (WMA)

`ChessboardDistance[u, v]`

returns the chessboard distance (also known as Chebyshev distance) between u and v , which is the number of moves a king on a chessboard needs to get from square u to square v .

```

>> ChessboardDistance[-7, 5]
12
>> ChessboardDistance[{-1, -1}, {1, 1}]
2

```

CosineDistance

Cosine similarity
(WMA)

`CosineDistance[u, v]`

returns the cosine distance between u and v .

The cosine distance is given by $\frac{u \cdot v}{\|u\| \|v\|}$.
 $\sin(\theta) = \frac{u \cdot v}{\|u\| \|v\|}$ with θ the angle between both vectors.

```

>> N[CosineDistance[{7, 9}, {71, 89}]]
0.0000759646

>> CosineDistance[{a, b}, {c, d}]

$$\frac{-ac - bd}{\sqrt{a^2 + b^2} \sqrt{c^2 + d^2}}$$


```

EuclideanDistance

Euclidean similarity (WMA)

`EuclideanDistance[u, v]`

returns the euclidean distance between u and v .

```

>> EuclideanDistance[-7, 5]
12
>> EuclideanDistance[{-1, -1}, {1, 1}]
2\sqrt{2}

```

```
>> EuclideanDistance[{a, b}, {c, d}]

$$\sqrt{\text{Abs}[a - c]^2 + \text{Abs}[b - d]^2}$$

```

ManhattanDistance

Manhattan distance (WMA)

`ManhattanDistance[u, v]`

returns the Manhattan distance between u and v , which is the number of horizontal or vertical moves in the gridlike Manhattan city layout to get from u to v .

```
>> ManhattanDistance[-7, 5]
12
>> ManhattanDistance[{-1, -1}, {1, 1}]
4
```

SquaredEuclideanDistance

WMA link

`SquaredEuclideanDistance[u, v]`

returns squared the euclidean distance between u and v .

```
>> SquaredEuclideanDistance[-7, 5]
144
>> SquaredEuclideanDistance[{-1, -1}, {1, 1}]
8
```

String Distances and Similarity Measures

String Distances and Similarity Measure

DamerauLevenshteinDistance

WMA link

`DamerauLevenshteinDistance[a, b]`

returns the Damerau-Levenshtein distance of a and b , which is defined as the minimum number of transpositions, insertions, deletions and substitutions needed to transform one into the other. In contrast to EditDistance, DamerauLevenshteinDistance counts transposition of adjacent items (e.g. “ab” into “ba”) as one operation of change.

```
>> DamerauLevenshteinDistance["kitten", "kitchen"]
2
>> DamerauLevenshteinDistance["abc", "ac"]
1
```

```

>> DamerauLevenshteinDistance["abc", "acb"]
1
>> DamerauLevenshteinDistance["azbc", "abxyc"]
3

```

The IgnoreCase option makes DamerauLevenshteinDistance ignore the case of letters:

```

>> DamerauLevenshteinDistance["time", "Thyme"]
3
>> DamerauLevenshteinDistance["time", "Thyme", IgnoreCase -> True]
2

```

DamerauLevenshteinDistance also works on lists:

```

>> DamerauLevenshteinDistance[{1, E, 2, Pi}, {1, E, Pi, 2}]
1

```

EditDistance

WMA link

`EditDistance[a, b]`

returns the Levenshtein distance of a and b , which is defined as the minimum number of insertions, deletions and substitutions on the constituents of a and b needed to transform one into the other.

```

>> EditDistance["kitten", "kitchen"]
2
>> EditDistance["abc", "ac"]
1
>> EditDistance["abc", "acb"]
2
>> EditDistance["azbc", "abxyc"]
3

```

The IgnoreCase option makes EditDistance ignore the case of letters:

```

>> EditDistance["time", "Thyme"]
3
>> EditDistance["time", "Thyme", IgnoreCase -> True]
2

```

EditDistance also works on lists:

```

>> EditDistance[{1, E, 2, Pi}, {1, E, Pi, 2}]
2

```

HammingDistance

WMA link

```
HammingDistance[u, v]
```

returns the Hamming distance between u and v , i.e. the number of different elements. u and v may be lists or strings.

```
>> HammingDistance[{1, 0, 1, 0}, {1, 0, 0, 1}]  
2  
>> HammingDistance["time", "dime"]  
1  
>> HammingDistance["TIME", "dime", IgnoreCase -> True]  
1
```

12. Drawing Graphics

Contents

AbsoluteThickness	142	GrayLevel	153	Polygon	160
Arrow	144	Hue	153	RGBColor	161
Arrowheads	145	Inset	154	Rectangle	162
CMYKColor	146	LABColor	154	RegularPolygon	163
Circle	147	LCHColor	154	Show	164
Directive	147	LUVColor	154	Small	164
Disk	149	Large	154	Text	165
EdgeForm	150	Line	155	Thick	165
FaceForm	150	Medium	155	Thickness	165
FilledCurve	151	Offset	155	Thin	166
FontColor	151	Opacity	156	Tiny	166
Graphics	152	Point	157	XYZColor	166
		PointSize	159		

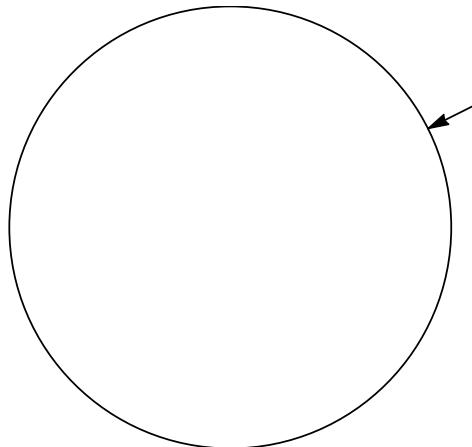
AbsoluteThickness

WMA link

AbsoluteThickness[*p*]

sets the line thickness for subsequent graphics primitives to *p* points.

```
>> Graphics[Table[{AbsoluteThickness[t], Line[{{20 t, 10}, {20 t, 80}}],  
Text[ToString[t]<>"pt", {20 t, 0}]}, {t, 0, 10}]]
```



Arrow

WMA link

```
Arrow[{p1, p2}]
```

represents a line from $p1$ to $p2$ that ends with an arrow at $p2$.

```
Arrow[{p1, p2}, s]
```

represents a line with arrow that keeps a distance of s from $p1$ and $p2$.

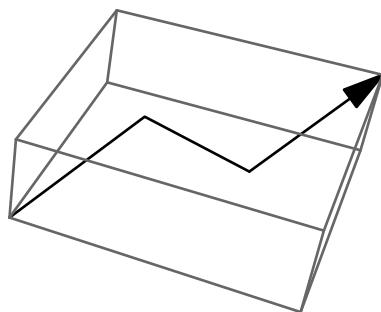
```
Arrow[{point_1, point_2}, {s1, s2}]
```

represents a line with arrow that keeps a distance of $s1$ from $p1$ and a distance of $s2$ from $p2$.

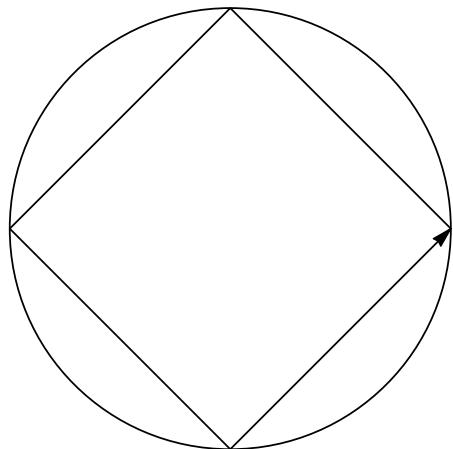
```
Arrow[{point_1, point_2}, {s1, s2}]
```

represents a line with arrow that keeps a distance of $s1$ from $p1$ and a distance of $s2$ from $p2$.

```
>> Graphics[Arrow[{{0,0}, {1,1}}]]
```

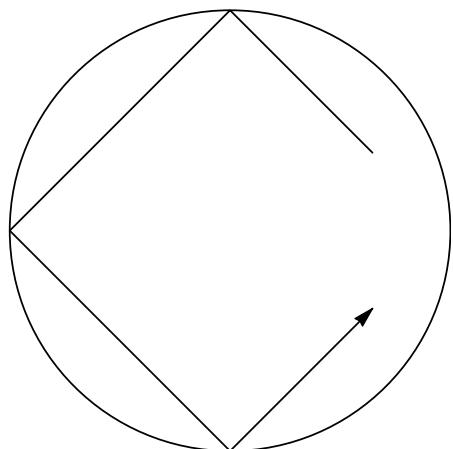


```
>> Graphics[{Circle[], Arrow[{{2, 1}, {0, 0}}, 1]}]
```



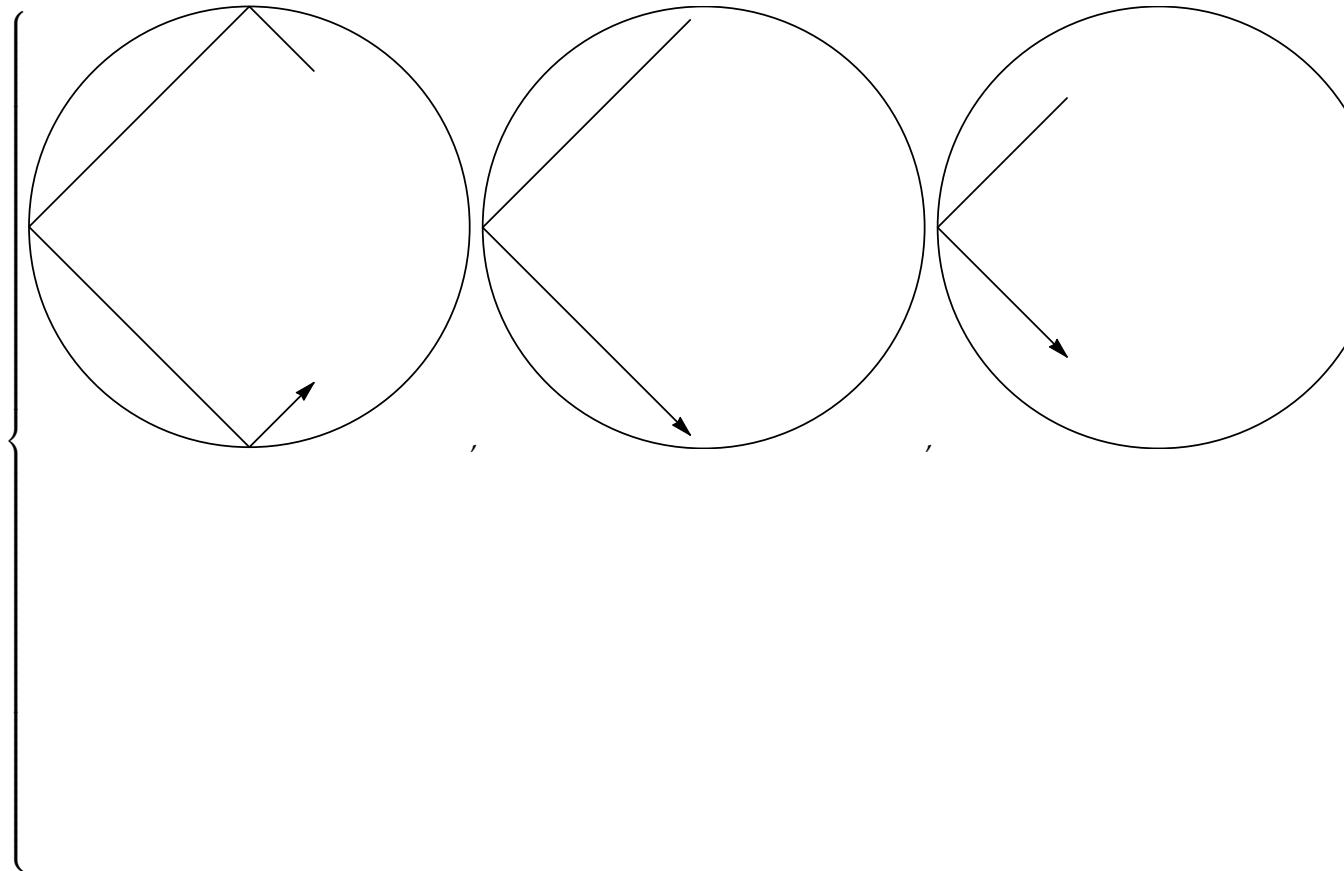
Arrows can also be drawn in 3D by giving point in three dimensions:

```
>> Graphics3D[Arrow[{{1, 1, -1}, {2, 2, 0}, {3, 3, -1}, {4, 4, 0}}]]
```



Keeping distances may happen across multiple segments:

```
>> Table[Graphics[{Circle[], Arrow[Table[{Cos[phi], Sin[phi]}, {phi, 0, 2*Pi, Pi/2}], {d, d}]], {d, 0, 2, 0.5}]
```



Arrowheads

WMA link

Arrowheads[s]

specifies that `Arrow[]` draws one arrow of size s (relative to width of image, defaults to 0.04).

Arrowheads[{spec1, spec2, ..., specn}]

specifies that `Arrow[]` draws n arrows as defined by $spec1, spec2, \dots, specn$.

Arrowheads[{{s}}]

specifies that one arrow of size s should be drawn.

Arrowheads[{{s, pos}}]

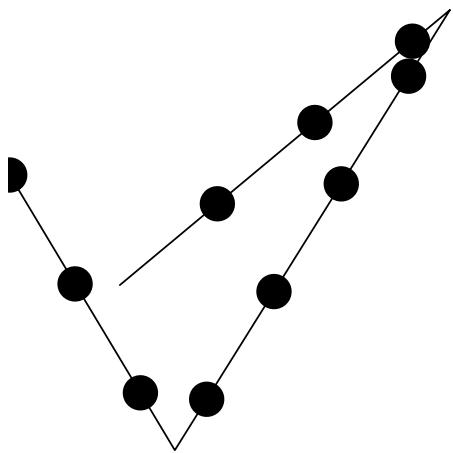
specifies that one arrow of size s should be drawn at position pos (for the arrow to be on the line, pos has to be between 0, i.e. the start for the line, and 1, i.e. the end of the line).

Arrowheads[{{s, pos, g}}]

specifies that one arrow of size s should be drawn at position pos using Graphics g .

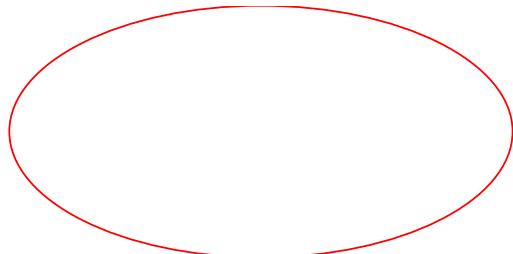
Arrows on both ends can be achieved using negative sizes:

```
>> Graphics[{Circle[], Arrowheads[{-0.04, 0.04}], Arrow[{{0, 0}, {2, 2}}, {1, 1}]}]
```

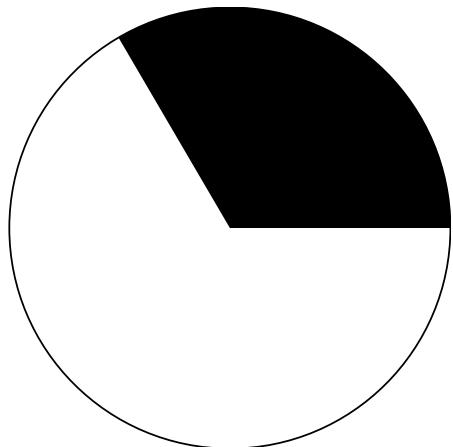


You may also specify our own arrow shapes:

```
>> Graphics[{Circle[], Arrowheads[{{0.04, 1, Graphics[{Red, Disk[]}]}]}, Arrow[{{0, 0}, {Cos[Pi/3], Sin[Pi/3]}}]}]
```



```
>> Graphics[{Arrowheads[Table[{0.04, i/10, Graphics[Disk[]]}, {i, 1, 10}]], Arrow[{{0, 0}, {6, 5}, {1, -3}, {-2, 2}}]}]
```



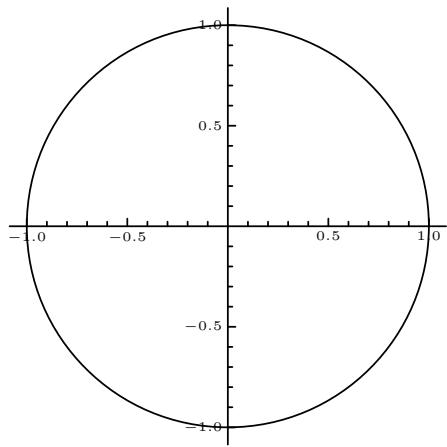
CMYKColor

WMA link

```
CMYKColor[c, m, y, k]
```

represents a color with the specified cyan, magenta, yellow and black components.

```
>> Graphics[MapIndexed[{CMYKColor @@ #1, Disk[2*#2 ~Join~{0}]} &,
IdentityMatrix[4]], ImageSize->Small]
```

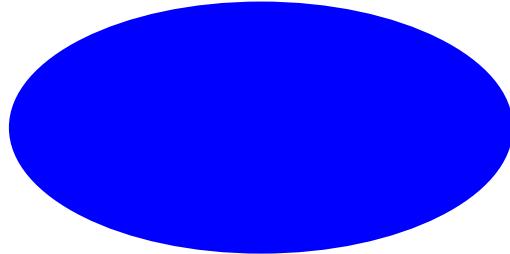


Circle

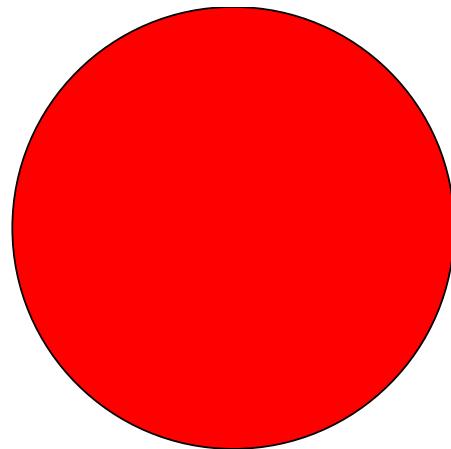
WMA link

```
Circle[{cx, cy}, r]
  draws a circle with center (cx, cy) and radius r.
Circle[{cx, cy}, {rx, ry}]
  draws an ellipse.
Circle[{cx, cy}]
  chooses radius 1.
Circle[]
  chooses center (0, 0) and radius 1.
```

```
>> Graphics[{Red, Circle[{0, 0}, {2, 1}]}]
```

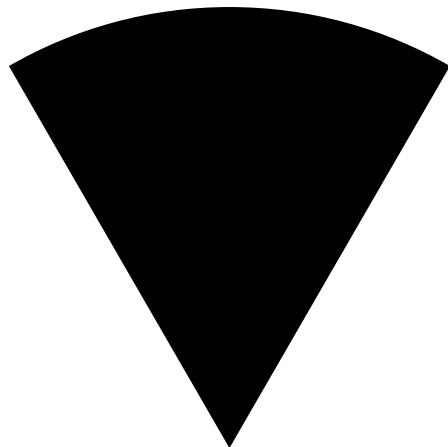


```
>> Graphics[{Circle[], Disk[{0, 0}, {1, 1}, {0, 2.1}]}]
```



Target practice:

```
>> Graphics[Circle[], Axes-> True]
```



Directive

WMA link

`Directive[g_1, g_2, ...]`

represents a single graphics directive composed of the directives g_1, g_2, \dots

Disk

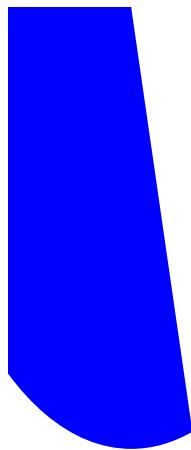
WMA link

```

Disk[{cx, cy}, r]
  fills a circle with center (cx, cy) and radius r.
Disk[{cx, cy}, {rx, ry}]
  fills an ellipse.
Disk[{cx, cy}]
  chooses radius 1.
Disk[]
  chooses center (0, 0) and radius 1.
Disk[{x, y}, ..., {t1, t2}]
  is a sector from angle t1 to t2.

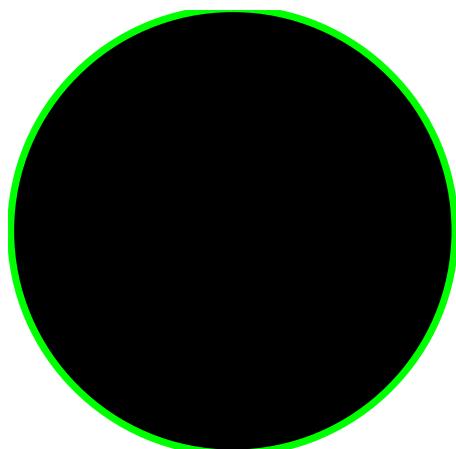
```

```
>> Graphics[{Blue, Disk[{0, 0}, {2, 1}]}]
```



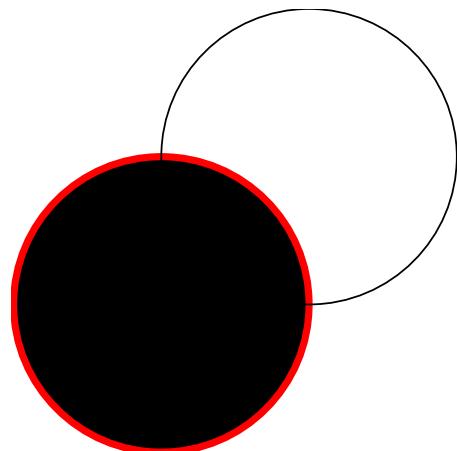
The outer border can be drawn using EdgeForm:

```
>> Graphics[{EdgeForm[Black], Red, Disk[]}]
```

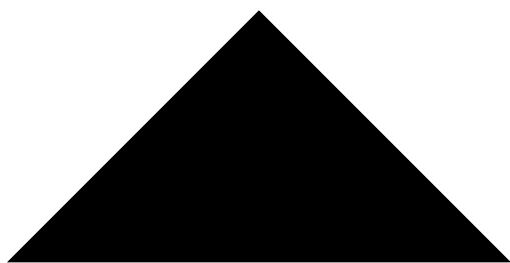


Disk can also draw sectors of circles and ellipses

```
>> Graphics[Disk[{0, 0}, 1, {Pi / 3, 2 Pi / 3}]]
```



```
>> Graphics[{Blue, Disk[{0, 0}, {1, 2}, {Pi / 3, 5 Pi / 3}]}]
```



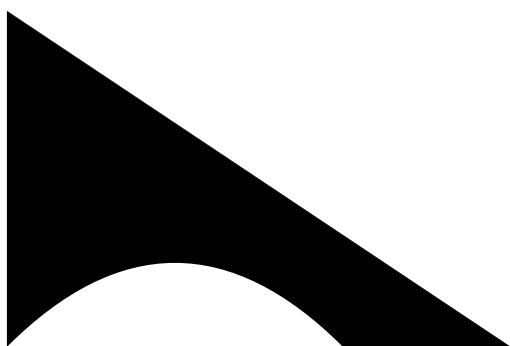
EdgeForm

WMA link

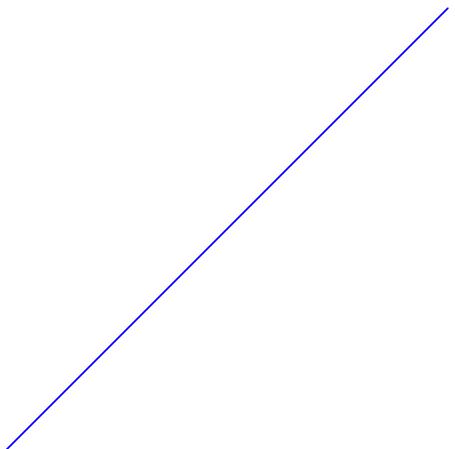
`EdgeForm[g]`

is a graphics directive that specifies that edges of filled graphics objects are to be drawn using the graphics directive or list of directives g .

```
>> Graphics[{EdgeForm[{Thick, Green}], Disk[]}]
```



```
>> Graphics[{Style[Disk[], EdgeForm[{Thick, Red}], Circle[{1, 1}]}]
```



FaceForm

WMA link

`FaceForm[g]`

is a graphics directive that specifies that faces of filled graphics objects are to be drawn using the graphics directive or list of directives g .

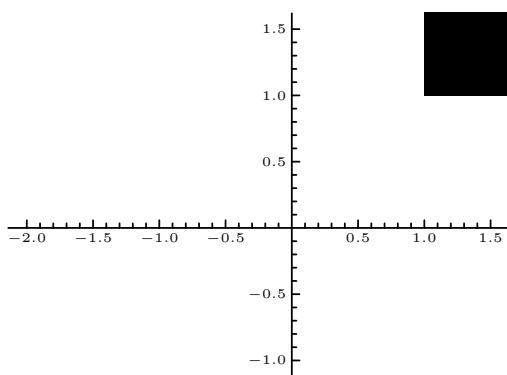
FilledCurve

WMA link

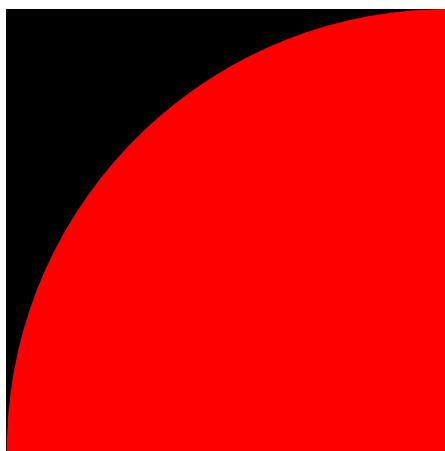
`FilledCurve[{segment1, segment2 ...}]`

represents a filled curve.

```
>> Graphics[FilledCurve[{Line[{{0, 0}, {1, 1}, {2, 0}}]}]]
```



```
>> Graphics[FilledCurve[{{BezierCurve[{{0, 0}, {1, 1}, {2, 0}}]}, Line[{{3, 0}, {0, 2}}]}]]
```



FontColor

WMA link

FontColor
is an option for Style to set the font color.

Graphics

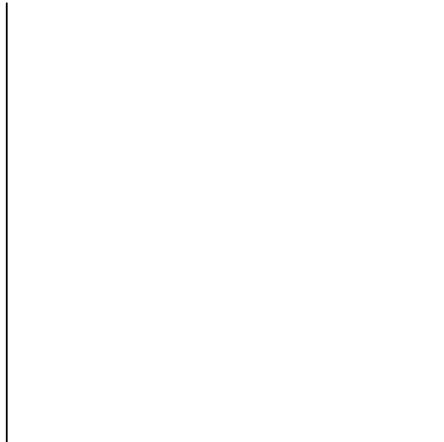
WMA link

Graphics [*primitives*, *options*]
represents a graphic.

Options include:

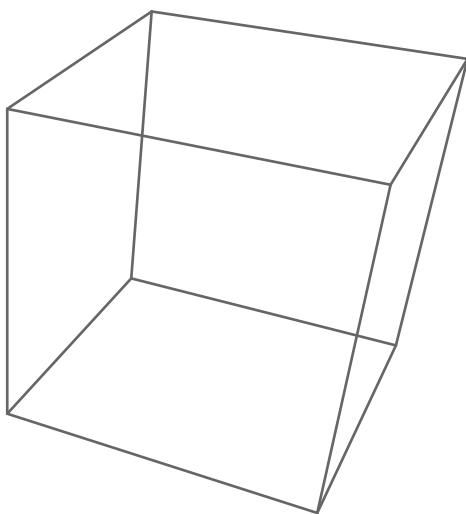
- Axes
- TicksStyle
- AxesStyle
- LabelStyle
- AspectRatio
- PlotRange
- PlotRangePadding
- ImageSize
- Background

```
>> Graphics[{Blue, Line[{{0,0}, {1,1}}]}]
```



Graphics supports PlotRange:

```
>> Graphics[{Rectangle[{1, 1}], Axes -> True, PlotRange -> {{-2, 1.5}, {-1, 1.5}}}]
```



```
>> Graphics[{Rectangle[], Red, Disk[{1,0}]}], PlotRange->{{0,1},{0,1}}
```

Graphics produces GraphicsBox boxes:

```
>> Graphics[Rectangle[]] // ToBoxes // Head
GraphicsBox
```

In TeXForm, Graphics produces Asymptote figures:

```
>> Graphics[Circle[]] // TeXForm
\begin{asy}
usepackage("amsmath");
size(5.8556cm, 5.8333cm);
draw(ellipse((175,175),175,175), rgb(0, 0, 0)+linewidth(0.66667));
clip(box((-0.33333,0.33333), (350.33,349.67)));
\end{asy}
```

GrayLevel

WMA link

`GrayLevel[g]`

represents a shade of gray specified by g , ranging from 0 (black) to 1 (white).

`GrayLevel[g, a]`

represents a shade of gray specified by g with opacity a .

Hue

WMA link

`Hue[h, s, l, a]`

represents the color with hue h , saturation s , lightness l and opacity a .

`Hue[h, s, l]`

is equivalent to `Hue[h, s, l, 1]`.

`Hue[h, s]`

is equivalent to `Hue[h, s, 1, 1]`.

`Hue[h]`

is equivalent to `Hue[h, 1, 1, 1]`.

```
>> Graphics[Table[{EdgeForm[Gray], Hue[h, s], Disk[{12h, 8s}]}, {h, 0, 1, 1/6}, {s, 0, 1, 1/4}]]
```

```
>> Graphics[Table[{EdgeForm[{GrayLevel[0, 0.5]}], Hue[(-11+q+10r)/72, 1, 1, 0.6], Disk[(8-r){Cos[2Pi q/12], Sin[2Pi q/12]}, (8-r)/3]}, {r, 6}, {q, 12}]]
```

•

Inset

WMA link

`Text [obj]`

represents an object *obj* inset in a graphic.

`Text [obj, pos]`

represents an object *obj* inset in a graphic at position *pos*.

`Text [obj, pos, $$]`

represents an object *obj* inset in a graphic at position *pos*, in away that the position *opos* of *obj* coincides with *pos* in the enclosing graphic.

LABColor

WMA link

`LABColor [l, a, b]`

represents a color with the specified lightness, red/green and yellow/blue components in the CIE 1976 L*a*b* (CIELAB) color space.

LCHColor

WMA link

`LCHColor [l, c, h]`

represents a color with the specified lightness, chroma and hue components in the CIELCh CIELab cube color space.

UVColor

WMA link

`UVColor [l, u, v]`

represents a color with the specified components in the CIE 1976 L*u*v* (CIELUV) color space.

Large

WMA link

`ImageSize -> Large`

produces a large image.

Line

WMA link

`Line [{point_1, point_2 ...}]`

represents the line primitive.

`Line[{{p_11, p_12, ...}, {p_21, p_22, ...}, ...}]`

represents a number of line primitives.

```
>> Graphics[Line[{{0,1},{0,0},{1,0},{1,1}}]]  
●  
>> Graphics3D[Line[{{0,0,0},{0,1,1},{1,0,0}}]]  
●
```

Medium

WMA link

`ImageSize -> Medium`
produces a medium-sized image.

Offset

WMA link

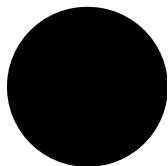
`Offset[{dx, dy}, position]`
gives the position of a graphical object obtained by starting at the specified *position* and then moving by absolute offset $\{dx,dy\}$.

Opacity

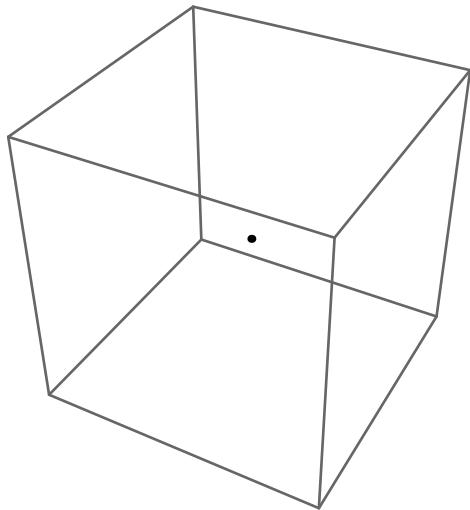
WMA link

`Opacity[level]`
is a graphics directive that sets the opacity to *level*.

```
>> Graphics[{Blue, Disk[{.5, 1}, 1], Opacity[.4], Red, Disk[], Opacity  
[.2], Green, Disk[{-5, 1}, 1]}]
```

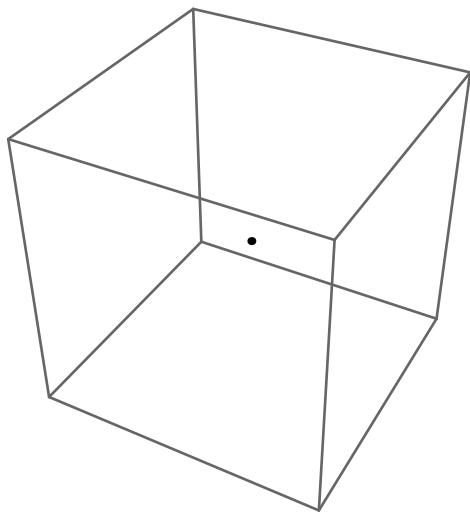


```
>> Graphics3D[{Blue, Sphere[], Opacity[.4], Red, Cuboid[]}]
```



Notice that `Opacity` does not overwrite the value of the alpha channel if it is set in a color directive:

```
>> Graphics[{Blue, Disk[], RGBColor[1,0,0,1], Opacity[.2], Rectangle
[{0,0},{1,1}]}]
```



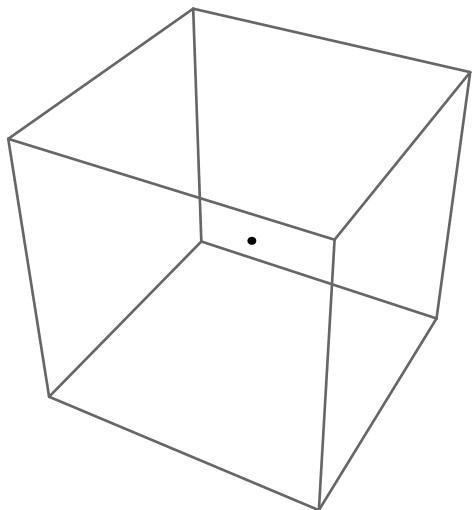
Point

WMA link

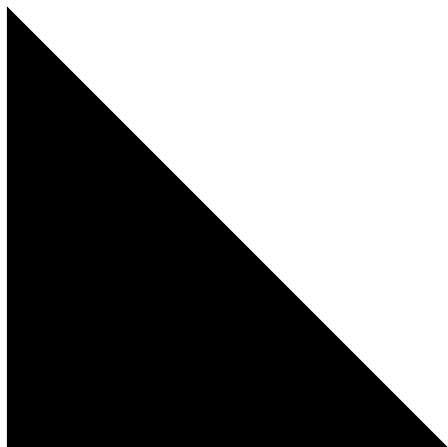
```
Point[{point_1, point_2 ...}]
represents the point primitive.
Point[{{p_11, p_12, ...}, {p_21, p_22, ...}, ...]
represents a number of point primitives.
```

Points are rendered if possible as circular regions. Their diameters can be specified using `PointSize`. Points can be specified as $\{x, y\}$:

```
>> Graphics[Point[{0, 0}]]
```



```
>> Graphics[Point[Table[{Sin[t], Cos[t]}, {t, 0, 2. Pi, Pi / 15.}]]]
```



or as $\{x, y, z\}$:

```
>> Graphics3D[{Orange, PointSize[0.05], Point[Table[{Sin[t], Cos[t], 0}, {t, 0, 2 Pi, Pi / 15.}]]}]
```

PointSize

WMA link

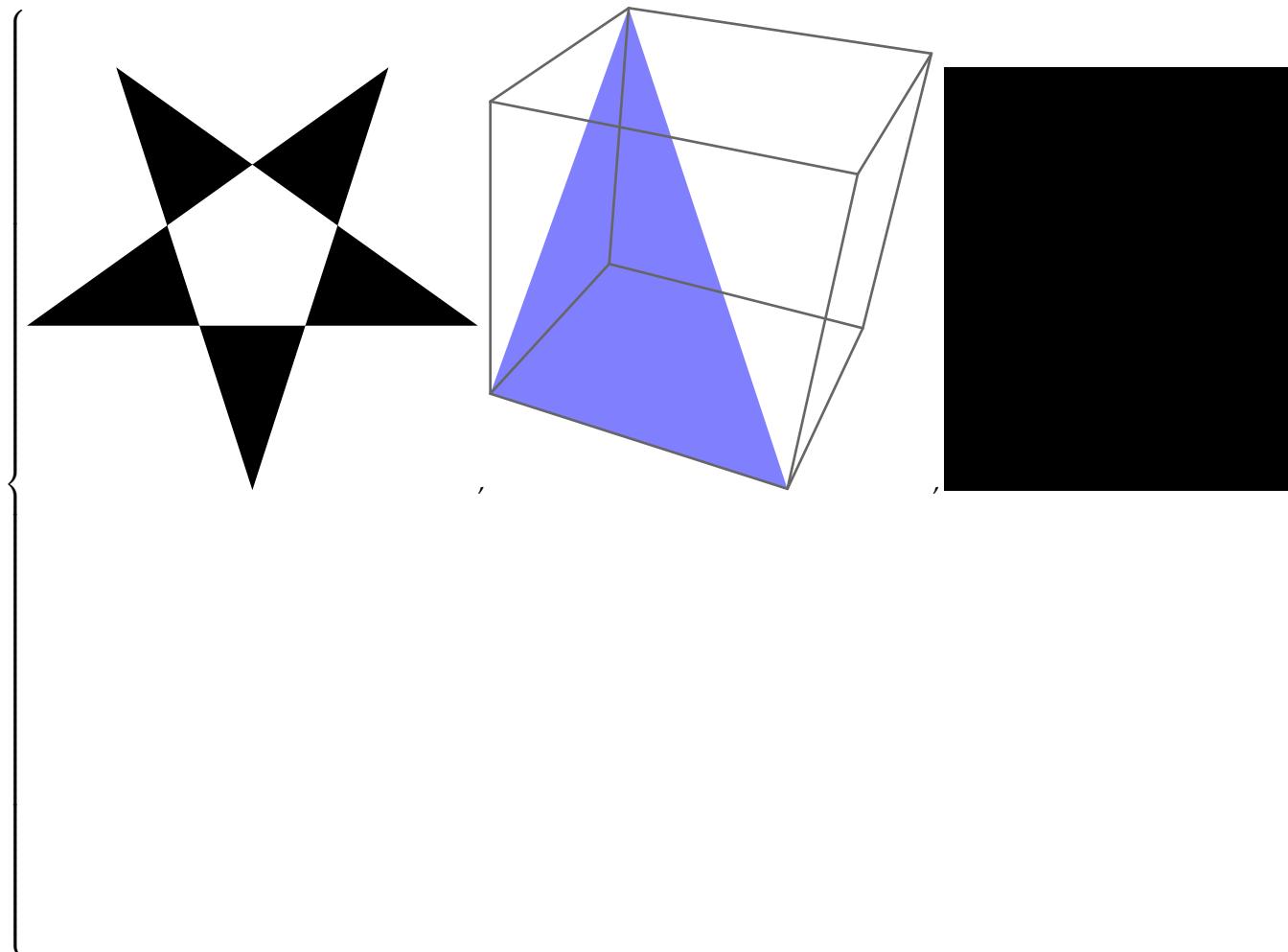
PointSize[t]

sets the diameter of points to t , which is relative to the overall width.

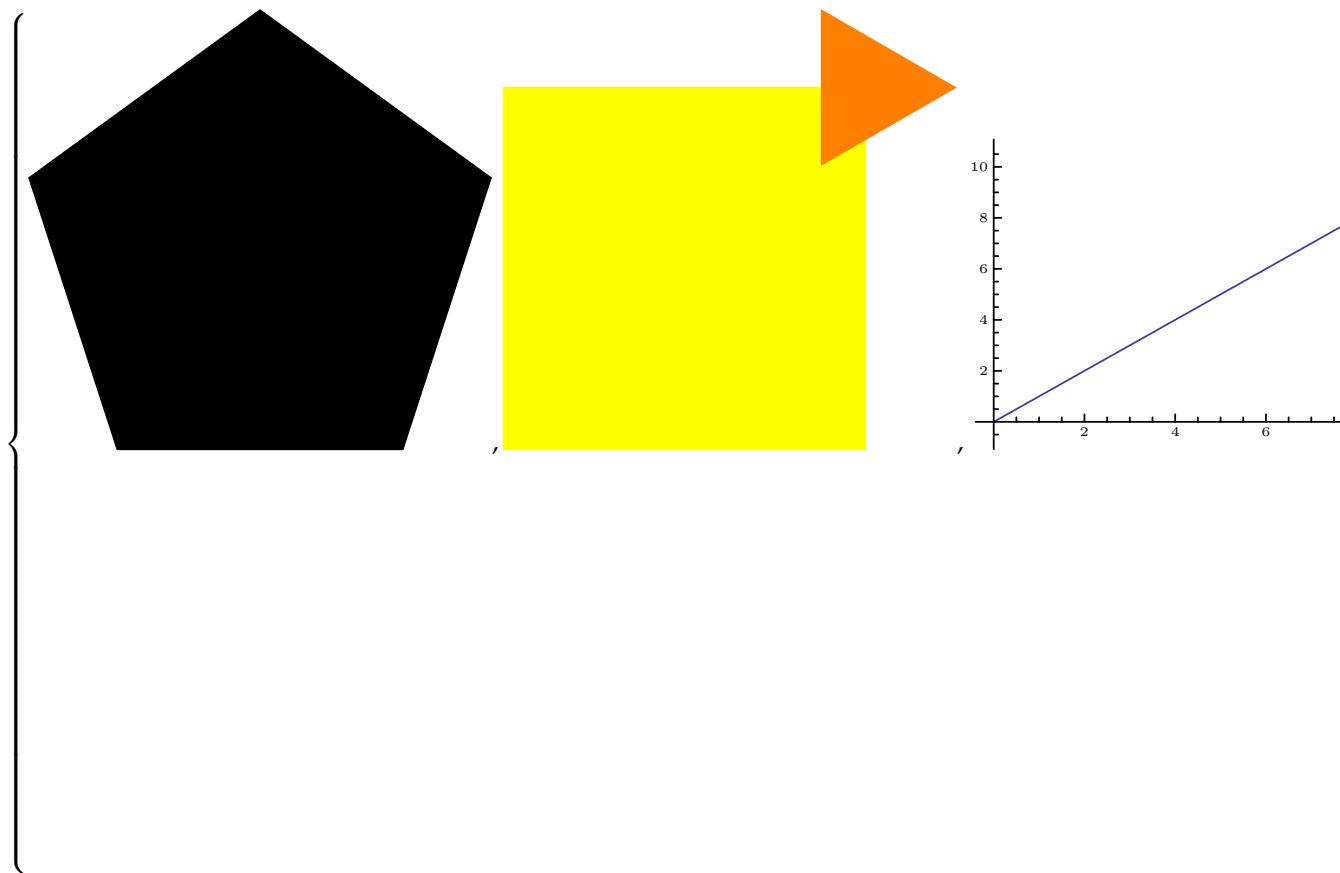
PointSize

can be used for both two- and three-dimensional graphics. The initial default pointsize is 0.008 for two-dimensional graphics and 0.01 for three-dimensional graphics.

```
>> Table[Graphics[{PointSize[r], Point[{0, 0}]}], {r, {0.02, 0.05, 0.1, 0.3}}]
```



```
>> Table[Graphics3D[{PointSize[r], Point[{0, 0, 0}]}], {r, {0.05, 0.1, 0.8}}]
```



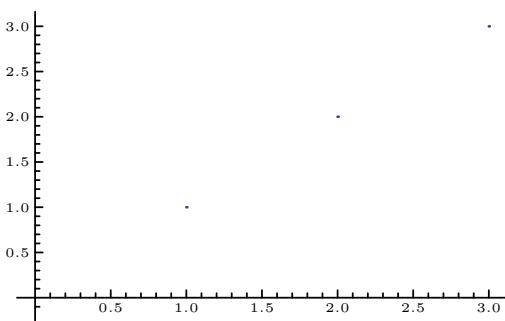
Polygon

WMA link

```
Polygon[{point_1, point_2 ...}]  
represents the filled polygon primitive.  
Polygon[{p_11, p_12, ...}, {p_21, p_22, ...}, ...]  
represents a number of filled polygon primitives.
```

A Right Triangle:

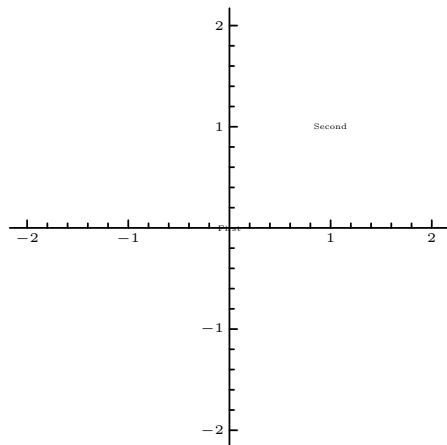
```
>> Graphics[Polygon[{{1,0},{0,0},{0,1}}]]
```



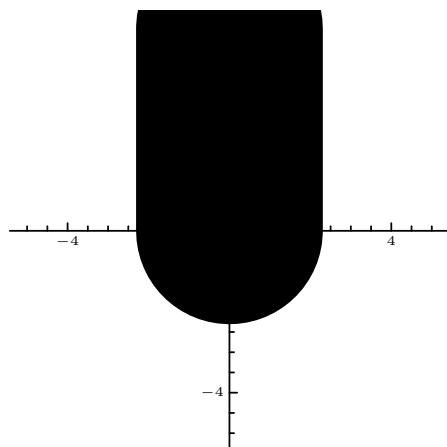
Notice that there is a line connecting from the last point to the first one.

A point is an element of the polygon if a ray from the point in any direction in the plane crosses the boundary line segments an odd number of times.

```
>> Graphics[Polygon[{{150,0},{121,90},{198,35},{102,35},{179,90}}]]
```



```
>> Graphics3D[Polygon[{{0,0,0},{0,1,1},{1,0,0}}]]
```



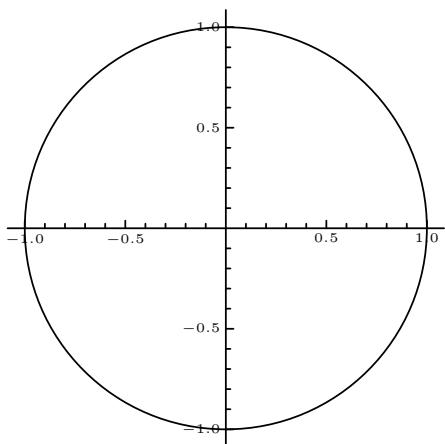
RGBColor

WMA link

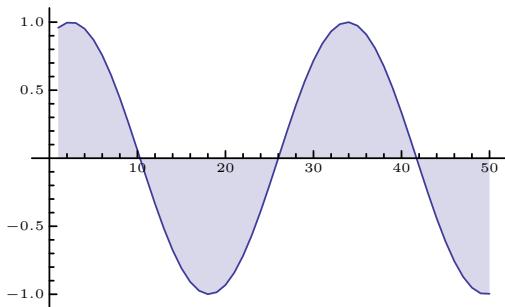
RGBColor[*r*, *g*, *b*]

represents a color with the specified red, green and blue components.

```
>> Graphics[MapIndexed[{RGBColor @@ #1, Disk[2*#2 ~Join~{0}]} &,
IdentityMatrix[3]], ImageSize->Small]
```



```
>> RGBColor[0, 1, 0]
```



```
>> RGBColor[0, 1, 0] // ToBoxes
```

```
StyleBox[GraphicsBox[{{EdgeForm[RGBColor[0,0,0]],RGBColor[
0,1,0],RectangleBox[{0,0}],AspectRatio->Automatic,Axes
->False,AxesStyle->{},Background->Automatic,ImageSize
->16,LabelStyle->{},PlotRange->Automatic,PlotRangePadding
->Automatic,TicksStyle->{}]},ImageSizeMultipliers
->{1,1},ShowStringCharacters->True]]
```

Rectangle

WMA link

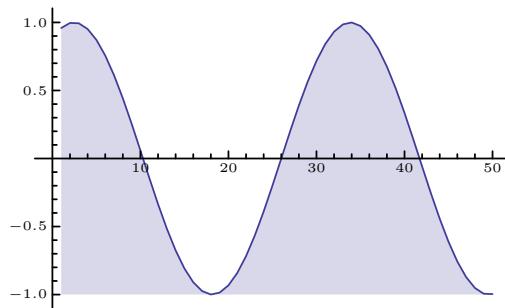
`Rectangle[{xmin, ymin}]`

represents a unit square with bottom-left corner at $\{\text{xmin}, \text{ymin}\}$.

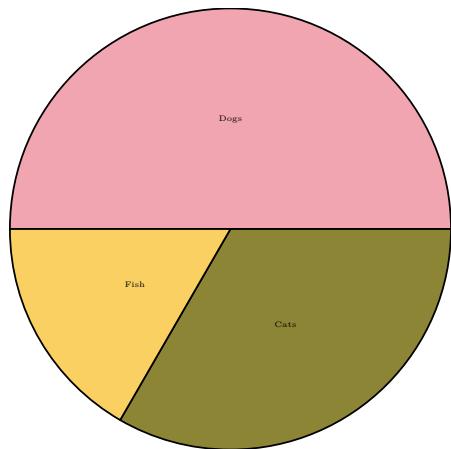
`'Rectangle[{xmin, ymin}, {xmax, ymax}]`

is a rectangle extending from $\{\text{xmin}, \text{ymin}\}$ to $\{\text{xmax}, \text{ymax}\}$.

```
>> Graphics[Rectangle[]]
```



```
>> Graphics[{Blue, Rectangle[{0.5, 0}], Orange, Rectangle[{0, 0.5}]}]
```

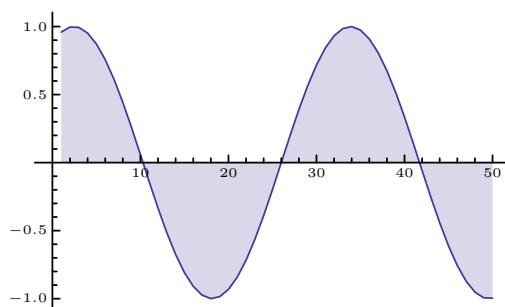


RegularPolygon

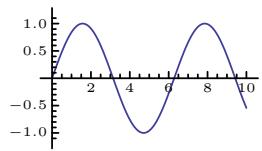
WMA link

```
RegularPolygon[n]
  gives the regular polygon with  $n$  edges.
RegularPolygon[r, n]
  gives the regular polygon with  $n$  edges and radius  $r$ .
RegularPolygon[{r, phi}, n]
  gives the regular polygon with radius  $r$  with one vertex drawn at angle  $\phi$ .
RegularPolygon[{$x$, $y}, r, n]
  gives the regular polygon centered at the position  $\{x, y\}$ .
```

```
>> Graphics[RegularPolygon[5]]
```



```
>> Graphics[{Yellow, Rectangle[], Orange, RegularPolygon[{1, 1}, {0.25, 0}, 3]}]
```



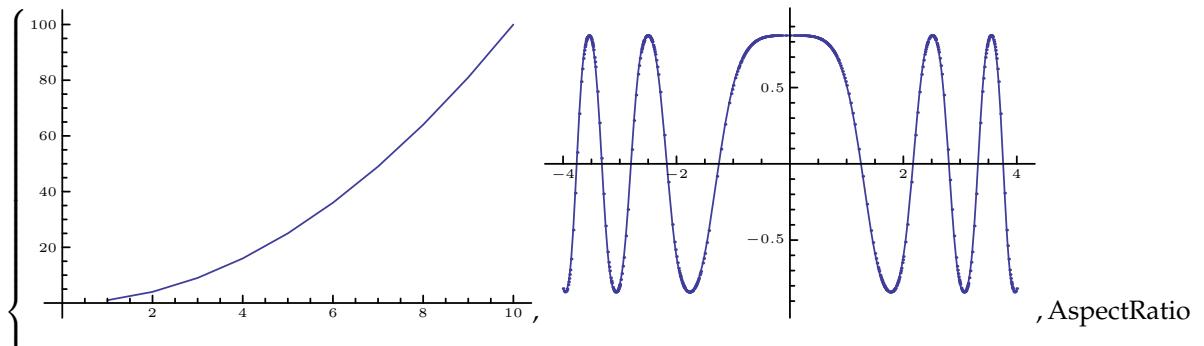
Show

WMA link

`Show[graphics, options]`

shows a list of graphics with the specified options added.

```
>> Show[{\{Plot[x, {x, 0, 10}], ListPlot[{1,2,3}]\}}]
```



- > Automatic, Axes -> False, AxesStyle -> {}, Background
- > Automatic, ImageSize -> Automatic, LabelStyle -> {}, PlotRange

- > Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}

Small

WMA link

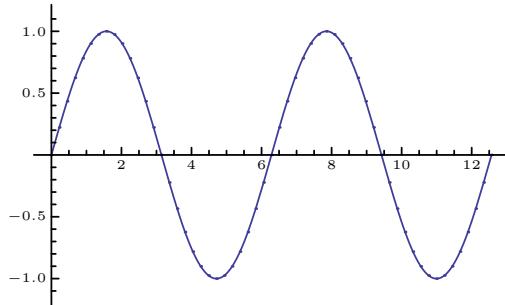
ImageSize -> Small
produces a small image.

Text

WMA link

```
Text["text", {x, y}]  
draws text centered on position {x, y}.
```

```
>> Graphics[{Text["First", {0, 0}], Text["Second", {1, 1}]}, Axes->True,  
PlotRange->{{-2, 2}, {-2, 2}}]
```



Thick

WMA link

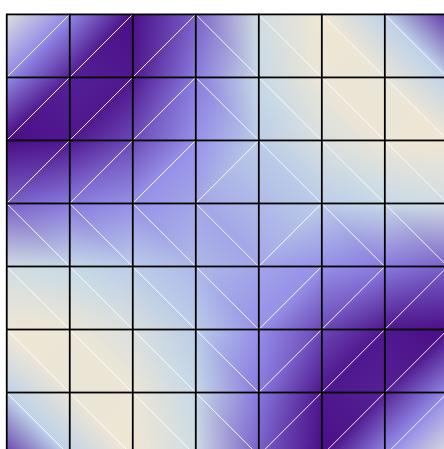
```
Thick  
sets the line width for subsequent graphics primitives to 2pt.
```

Thickness

WMA link

```
Thickness[t]  
sets the line thickness for subsequent graphics primitives to t times the size of the plot area.
```

```
>> Graphics[{Thickness[0.2], Line[{{0, 0}, {0, 5}}]}, Axes->True,  
PlotRange->{{-5, 5}, {-5, 5}}]
```



Thin

WMA link

Thin

sets the line width for subsequent graphics primitives to 0.5pt.

Tiny

WMA link

ImageSize -> Tiny

produces a tiny image.

XYZColor

WMA link

XYZColor[*x*, *y*, *z*]

represents a color with the specified components in the CIE 1931 XYZ color space.

13. Drawing Options and Option Values

The various common Plot and Graphics options, along with the meaning of specific option values are described here.

Contents

Automatic	167	ChartLegends	169	Mesh	172
Axes	168	Filling	170	PlotPoints	172
Axis	168	Full	170	PlotRange	173
Bottom	169	ImageSize	170	TicksStyle	173
ChartLabels	169	Joined	171	Top	174
		MaxRecursion	171		

Automatic

WMA link

Automatic

is used to specify an automatically computed option value.

Automatic is the default for PlotRange, ImageSize, and other graphical options:

```
>> Cases[Options[Plot], HoldPattern[_ :> Automatic]]
```

```
{Background:>Automatic, Exclusions:>Automatic, ImageSize:>Automatic, MaxRecursion:>Automatic, PlotRange:>
```

Axes

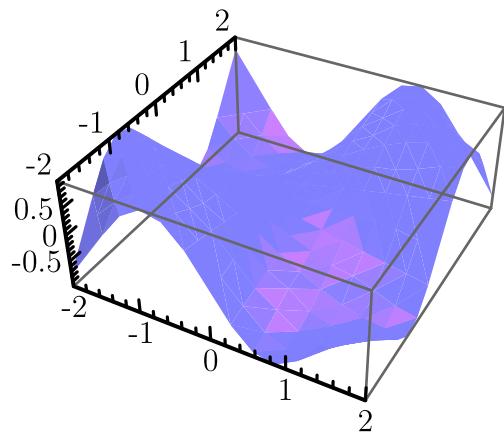
WMA link

Axes

is an option for charting and graphics functions that specifies whether axes should be drawn.

- Axes->True draws all axes.
- Axes->False draws no axes.
- Axes->\{False, True\} draws an axis y but no x axis in two dimensions.

```
>> Graphics[Circle[], Axes -> True]
```



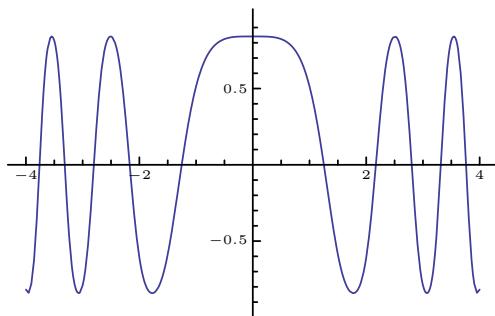
Axis

WMA link

Axis

is a possible value for the Filling option.

```
>> ListLinePlot[Table[Sin[x], {x, -5, 5, 0.2}], Filling->Axis]
```



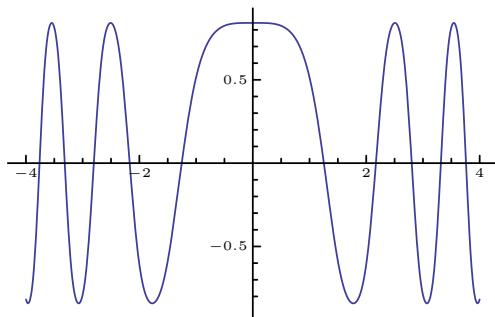
Bottom

WMA link

Bottom

is a possible value for the Filling option.

```
>> ListLinePlot[Table[Sin[x], {x, -5, 5, 0.2}], Filling->Bottom]
```



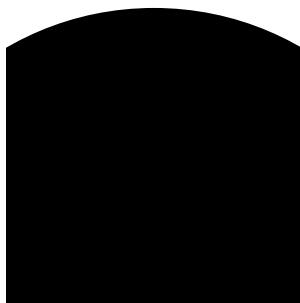
ChartLabels

WMA link

ChartLabels

is a charting option that specifies what labels should be used for chart elements.

```
>> PieChart[{30, 20, 10}, ChartLabels -> {Dogs, Cats, Fish}]
```



ChartLegends

WMA link

ChartLegends

is an option for charting functions that specifies the legends to be used for chart elements.

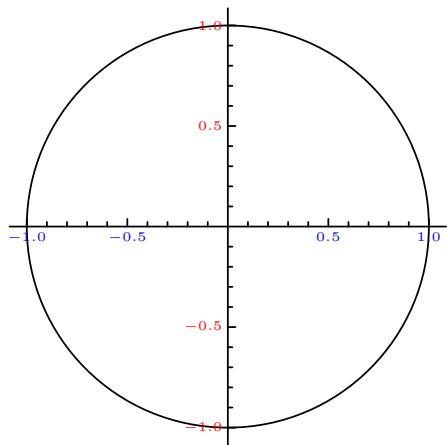
Filling

WMA link

Filling -> [Top | Bottom| Axis]

Filling is a an option to ListPlot, Plot or Plot3D, and related functions that indicates what filling to add under point, curves, and surfaces.

```
>> ListLinePlot[Table[Sin[x], {x, -5, 5, 0.2}], Filling->Axis]
```



Full

WMA link

Full

is a possible value for the Mesh and PlotRange options.

ImageSize

WMA link

ImageSize

is an option that specifies the overall size of an image to display.

Specifications for both width and height can be any of the following:

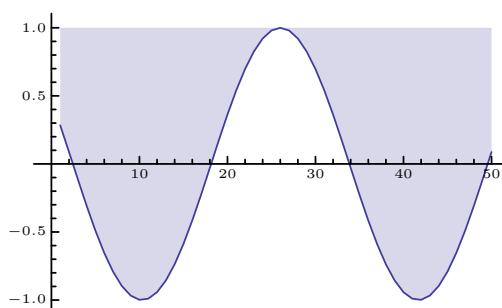
Automatic

determined by location or other dimension (default)

Tiny, Small, Medium, Large

pre defined absolute sizes

```
>> Plot[Sin[x], {x, 0, 10}, ImageSize -> Small]
```



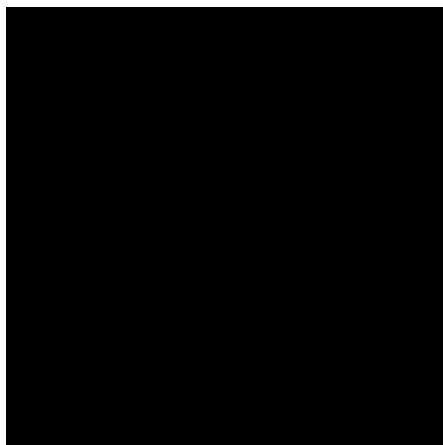
Joined

WMA link

Joined *boolean*

is an option for Plot that gives whether to join points to make lines.

```
>> ListPlot[Table[n ^ 2, {n, 10}], Joined->True]
```



MaxRecursion

WMA link

MaxRecursion

is an option for functions like NIntegrate and Plot that specifies how many recursive subdivisions can be made.

```
>> NIntegrate[Exp[-10^8 x^2], {x, -1, 1}, Method->"Internal",
  MaxRecursion -> 3]
0.0777778

>> NIntegrate[Exp[-10^8 x^2], {x, -1, 1}, Method->"Internal",
  MaxRecursion -> 6]
0.00972222
```

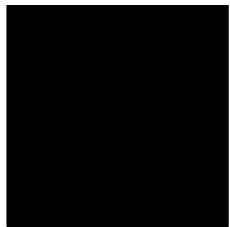
Mesh

WMA link

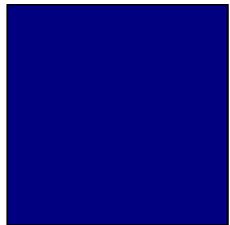
Mesh

is a charting option, such as for Plot, BarChart, PieChart, etc. that specifies the mesh to be drawn. The default is Mesh->None.

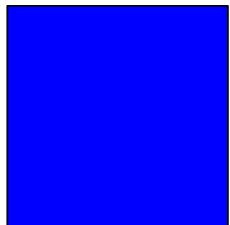
```
>> Plot[Sin[Cos[x^2]], {x, -4, 4}, Mesh->All]
```



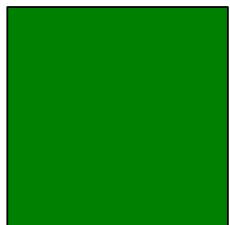
```
>> Plot[Sin[x], {x, 0, 4 Pi}, Mesh->Full]
```



```
>> DensityPlot[Sin[x y], {x, -2, 2}, {y, -2, 2}, Mesh->Full]
```



```
>> Plot3D[Sin[x y], {x, -2, 2}, {y, -2, 2}, Mesh->Full]
```



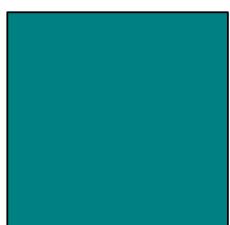
PlotPoints

WMA link

PlotPoints *n*

A number specifies how many initial sample points to use.

```
>> Plot[Sin[Cos[x^2]], {x, -4, 4}, PlotPoints->22]
```



PlotRange

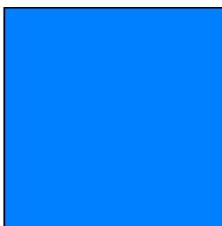
WMA link

PlotRange

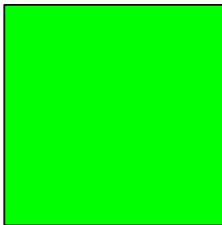
is a charting option, such as for Plot, BarChart, PieChart, etc. that gives the range of coordinates to include in a plot.

- All all points are included.
- Automatic - outlying points are dropped.
- max - explicit limit for each function.
- $\{min, max\}$ - explicit limits for y (2D), z (3D), or array values.
- $\{\{x_{min}, x_{max}\}, \{\$y_{min}, \$y_{max}\}\}$ - explicit limits for x and y .

```
>> Plot[Sin[Cos[x^2]], {x, -4, 4}, PlotRange -> All]
```



```
>> Graphics[Disk[], PlotRange -> {{-.5, .5}, {0, 1.5}}]
```



TicksStyle

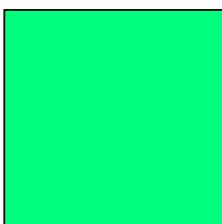
WMA link

TicksStyle

is an option for graphics functions which specifies how ticks should be rendered.

- TicksStyle gives styles for both tick marks and tick labels.
- TicksStyle can be used in both two and three-dimensional graphics.
- TicksStyle->*list* specifies the colors of each of the axes.

```
>> Graphics[Circle[], Axes -> True, TicksStyle -> {Blue, Red}]
```



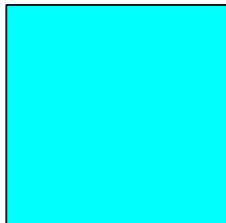
Top

WMA link

Top

is a possible value for the Filling option.

```
>> ListLinePlot[Table[Cos[x], {x, -5, 5, 0.2}], Filling->Top]
```



14. Expression Structure

Contents

Expression Sizes and Signatures	General Structural Expression	Null 179
ByteCount 175	Functions 176	Operate 180
Hash 176	ApplyLevel (@@@) 176	Order 180
LeafCount 176	BinarySearch 177	OrderedQ 180
	Depth 177	PatternsOrderedQ 180
	FreeQ 178	SortBy 181
	Level 179	Through 181

Expression Sizes and Signatures

Expression Sizes and Signature

ByteCount

WMA link

`ByteCount [expr]`
gives the internal memory space used by *expr*, in bytes.

The results may heavily depend on the Python implementation in use.

Hash

Hash function (WMA link)

`Hash [expr]`
returns an integer hash for the given *expr*.
`Hash [expr, type]`
returns an integer hash of the specified *type* for the given *expr*.
The types supported are “MD5”, “Adler32”, “CRC32”, “SHA”, “SHA224”, “SHA256”, “SHA384”, and “SHA512”.
`Hash [expr, type, format]`
Returns the hash in the specified format.

```
> Hash[“The Adventures of Huckleberry Finn”] = 213425047836523694663619736686226550816
> Hash[“The Adventures of Huckleberry Finn”, “SHA256”] = 950926495945903842880571834086092549189343518116698
> Hash[1/3] = 56073172797010645108327809727054836008
> Hash[{a, b, {c, {d, e, f}}}] = 135682164776235407777080772547528225284
> Hash[SomeHead[3.1415]] = 58042316473471877315442015469706095084
```

```
>> Hash[{a, b, c}, "xyzstr"]
Hash [{a,b,c},xyzstr,Integer]
```

LeafCount

WMA link

LeafCount [*expr*]
returns the total number of indivisible subexpressions in *expr*.

```
>> LeafCount[1 + x + y^a]
6
>> LeafCount[f[x, y]]
3
>> LeafCount[{1 / 3, 1 + I}]
7
>> LeafCount[Sqrt[2]]
5
>> LeafCount[100!]
1
```

General Structural Expression Functions

General Structural Expression Function

ApplyLevel (@@)

WMA link

ApplyLevel [*f*, *expr*]
f @@ *expr*
is equivalent to *Apply*[*f*, *expr*, {1}] .

```
>> f @@ {{a, b}, {c, d}}
{f[a,b],f[c,d]}
```

BinarySearch

Binary search algorithm (WMA)

CombinatoricaOld`BinarySearch [*l*, *k*]
searches the list *l*, which has to be sorted, for key *k* and returns its index in *l*.
If *k* does not exist in *l*, *BinarySearch* returns $(a + b) / 2$, where *a* and *b* are the indices between which *k* would have to be inserted in order to maintain the sorting order in *l*.
Please note that *k* and the elements in *l* need to be comparable under a strict total order.
CombinatoricaOld`BinarySearch [*l*, *k*, *f*]
gives the index of *k* in the elements of *l* if *f* is applied to the latter prior to comparison. Note that *f* needs to yield a sorted sequence if applied to the elements of *l*.

Number 100 is found at exactly in the fourth place of the given list:

```
>> CombinatoricaOld`BinarySearch[{3, 4, 10, 100, 123}, 100]
4
```

Number 7 is found in between the second and third place (3, and 9) of the given list. The numerical difference between 3 and 9 does not figure into the .5 part of 2.5:

```
>> CombinatoricaOld`BinarySearch[{2, 3, 9}, 7] // N
2.5
```

0.5 is what you get when the item comes before the given list:

```
>> CombinatoricaOld`BinarySearch[{-10, 5, 8, 10}, -100] // N
0.5
```

And here is what you see when the item comes at the end of the list:

```
>> CombinatoricaOld`BinarySearch[{-10, 5, 8, 10}, 20] // N
4.5
>> CombinatoricaOld`BinarySearch[{{a, 1}, {b, 7}}, 7, #[[2]]&]
2
```

Depth

WMA link

```
Depth[expr]
gives the depth of expr.
```

The depth of an expression is defined as one plus the maximum number of Part indices required to reach any part of *expr*, except for heads.

```
>> Depth[x]
1
>> Depth[x + y]
2
>> Depth[{{{x}}}]
5
```

Complex numbers are atomic, and hence have depth 1:

```
>> Depth[1 + 2 I]
1
```

Depth ignores heads:

```
>> Depth[f[a, b][c]]
2
```

FreeQ

WMA link

```
FreeQ[expr, x]
returns True if expr does not contain the expression x.
```

```

>> FreeQ[y, x]
True

>> FreeQ[a+b+c, a+b]
False

>> FreeQ[{1, 2, a^(a+b)}, Plus]
False

>> FreeQ[a+b, x+y+z]
True

>> FreeQ[a+b+c, x+y+z]
False

>> FreeQ[x+y+z] [a+b]
True

```

Level

[WMA link](#)

Level [*expr*, *levelspec*]
 gives a list of all subexpressions of *expr* at the level(s) specified by *levelspec*.

Level uses standard level specifications:

<i>n</i>	levels 1 through <i>n</i>
Infinity	all levels from level 1
{ <i>n</i> }	level <i>n</i> only
{ <i>m</i> , <i>n</i> }	levels <i>m</i> through <i>n</i>

Level 0 corresponds to the whole expression.

A negative level *-n* consists of parts with depth *n*.

Level -1 is the set of atoms in an expression:

```

>> Level[a + b ^ 3 * f[2 x ^ 2], {-1}]
{a, b, 3, 2, x, 2}

>> Level[{{{a}}}, 3]
{{a}, {{a}}, {{{a}}}}

>> Level[{{{a}}}, -4]
{{{a}}}

>> Level[{{{a}}}, -5]
{}

>> Level[{{{a}}}, {0, -1}]
{a, h3[a], h2[h3[a]], h1[h2[h3[a]]], h0[h1[h2[h3[a]]]]}

```

Use the option Heads \rightarrow True to include heads:

```
>> Level[{{{a}}}], 3, Heads -> True]
{List, List, List, {a}, {{a}}, {{{a}}}}
>> Level[x^2 + y^3, 3, Heads -> True]
{Plus, Power, x, 2, x^2, Power, y, 3, y^3}
>> Level[a ^ 2 + 2 * b, {-1}, Heads -> True]
{Plus, Power, a, 2, Times, 2, b}
>> Level[f[g[h]] [x], {-1}, Heads -> True]
{f, g, h, x}
>> Level[f[g[h]] [x], {-2, -1}, Heads -> True]
{f, g, h, g[h], x, f[g[h]] [x]}
```

Null

WMA link

Null

is the implicit result of expressions that do not yield a result.

```
>> FullForm[a:=b]
Null
```

It is not displayed in StandardForm,

```
>> a:=b
```

in contrast to the empty string:

```
>> ""
```

Operate

WMA link

Operate[p, expr]
applies p to the head of expr.
Operate[p, expr, n]
applies p to the nth head of expr.

```
>> Operate[p, f[a, b]]
p[f][a, b]
```

The default value of n is 1:

```
>> Operate[p, f[a, b], 1]
p[f][a, b]
```

With n=0, Operate acts like Apply:

```
>> Operate[p, f[a][b][c], 0]
p [f [a] [b] [c]]
```

Order

WMA link

Order[*x*, *y*]

returns a number indicating the canonical ordering of *x* and *y*. 1 indicates that *x* is before *y*, -1 that *y* is before *x*. 0 indicates that there is no specific ordering. Uses the same order as **Sort**.

```
>> Order[7, 11]
1
>> Order[100, 10]
-1
>> Order[x, z]
1
>> Order[x, x]
0
```

OrderedQ

WMA link

OrderedQ[{*a*, *b*}]

is True if *a* sorts before *b* according to canonical ordering.

```
>> OrderedQ[{a, b}]
True
>> OrderedQ[{b, a}]
False
```

PatternsOrderedQ

WMA link

PatternsOrderedQ[*patt1*, *patt2*]

returns True if pattern *patt1* would be applied before *patt2* according to canonical pattern ordering.

```
>> PatternsOrderedQ[x___, x_]
False
>> PatternsOrderedQ[x_, x__]
True
>> PatternsOrderedQ[b, a]
True
```

SortBy

WMA link

`SortBy[list, f]`

sorts *list* (or the elements of any other expression) according to canonical ordering of the keys that are extracted from the *list*'s elements using \$f. Chunks of elements that appear the same under \$f are sorted according to their natural order (without applying \$f).

`SortBy[f]`

creates an operator function that, when applied, sorts by \$f.

```
>> SortBy[{{5, 1}, {10, -1}}, Last]
{{10, -1}, {5, 1}}
```

```
>> SortBy[Total] [{{5, 1}, {10, -9}}]
{{10, -9}, {5, 1}}
```

Through

WMA link

`Through[p[f][x]]`

gives $p[f[x]]$.

```
>> Through[f[g][x]]
f[g[x]]
```

```
>> Through[p[f, g][x]]
p[f[x], g[x]]
```

15. File Formats

Built-in Importers.

Contents

HTML	182	ImageLinksImport	183	TagsImport	184
DataImport	182	PlaintextImport	183	XMLElement	184
FullDataImport	182	SourceImport	183	XMLGet	185
HTMLGet	182	TitleImport	184	XMLGetString	185
HTMLGetString	183	XMLObjectImport	184	XMLObject	185
HyperlinksImport	183	XML	184	XMLObjectImport	185
		PlaintextImport	184		

HTML

HTML

Basic implementation for a HTML importer.

DataImport

```
HTML`DataImport[{"filename"}]  
imports data from a HTML file.
```

```
>> Import["ExampleData/PrimeMeridian.html", "Data"][[1, 1, 2, 3]]  
{Washington, D.C., 77°03'56.07 W (1897) or 77°04'02.24 W (NAD  
27) or 77°04'01.16 W (NAD 83), New Naval Observatory meridian}
```

FullDataImport

```
HTML`FullDataImport[{"filename"}]  
imports data from a HTML file.
```

HTMLGet

```
HTMLGet[str]  
Parses str as HTML code.
```

HTMLGetString

```
HTML`Parser`HTMLGetString[{"string"}]  
parses HTML code contained in "string".
```

HyperlinksImport

```
HTML`HyperlinksImport[{"filename"}]  
imports hyperlinks from a HTML file.
```

```
>> Import["ExampleData/PrimeMeridian.html", "Hyperlinks"][[1]]  
/wiki/Prime_meridian_(Greenwich)
```

ImageLinksImport

```
HTML`ImageLinksImport[{"filename"}]  
imports links to the images included in a HTML file.
```

```
>> Import["ExampleData/PrimeMeridian.html", "ImageLinks"][[6]]  
//upload.wikimedia.org/wikipedia/commons/thumb/d/d5/Prime_meridian.jpg/180px-Prime_meridian.jpg
```

PlaintextImport

```
HTML`PlaintextImport[{"filename"}]  
imports plane text from a HTML file.
```

```
>> DeleteDuplicates[StringCases[Import["ExampleData/PrimeMeridian.html"],  
RegularExpression["Wiki[a-z]+"]]]
```

SourceImport

```
HTML`SourceImport[{"filename"}]  
imports source code from a HTML file.
```

```
>> DeleteDuplicates[StringCases[Import["ExampleData/PrimeMeridian.html",  
"Source"], RegularExpression["<t[a-z]+>"]]]  
{<title>, <tr>, <th>, <td>}
```

TitleImport

```
HTML`TitleImport[{"filename"}]  
imports the title string from a HTML file.
```

```
>> Import["ExampleData/PrimeMeridian.html", "Title"]
Prime meridian - Wikipedia
```

XMLObjectImport

HTML `XMLObjectImport[{"filename"}]
imports XML objects from a HTML file.

```
>> Part[Import["ExampleData/PrimeMeridian.html", "XMLObject"], 2, 3, 1,
3, 2]
XMLElement [title, {} , {Prime meridian - Wikipedia}]
```

XML

XML
Basic implementation for an XML importer.

PlaintextImport

WMA link

XML `PlaintextImport[{"string"}]
parses "string" as XML code, and returns it as plain text.

```
>> StringReplace[StringTake[Import["ExampleData/InventionNo1.xml", "Plaintext"], 31], FromCharacterCode[10] -> "/"]
MuseScore 1.2/2012-09-12/5.7/40
```

TagsImport

XML `TagsImport[{"string"}]
parses "string" as XML code, and returns a list with the tags found.

```
>> Take[Import["ExampleData/InventionNo1.xml", "Tags"], 10]
{accidental, alter, arpeggiate, articulations, attributes, backup, bar-style, barline, beam, beat-type}
```

XMLElement

WMA link

XMLElement [tag, {attr_1, val_1, ...}, {data, ...}]
represents an element in symbolic XML.

XMLGet

```
XMLGet[...]  
Internal. Document me.
```

XMLGetString

```
XML`Parser`XMLGetString[{"string"}]  
parsing "string" as XML code, and returns an XMLObject.
```

```
>> Head[XML`Parser`XMLGetString["<a></a>"]]  
XMLObject[Document]
```

XMLObject

WMA link

```
XMLObject[{"type"}]  
represents the head of an XML object in symbolic XML.
```

XMLObjectImport

```
XML`XMLObjectImport[{"string"}]  
parsing "string" as XML code, and returns a list of XMLObjects found.
```

```
>> Part[Import["ExampleData/InventionNo1.xml", "XMLObject"], 2, 3, 1]  
XMLElement[identification, {}], {XMLElement[  
encoding, {}], {XMLElement[software, {}], {MuseScore  
1.2}}, XMLElement[encoding-date, {}, {2012-09-12}]}]]]  
  
>> Part[Import["ExampleData/Namespace.xml"], 2]  
XMLElement[title, {}], {Prime meridian - Wikipedia}]
```

16. File and Stream Operations

Contents

\$Input	186	InputStream	189	Record	193
\$InputFileName	186	Number	189	SetStreamPosition	194
Character	186	OpenAppend	189	Skip	194
Close	187	OpenRead	189	StreamPosition	194
EndOfFile	187	OpenWrite	190	Streams	195
Expression	187	OutputStream	190	StringToStream	195
FilePrint	187	Put (>>)	191	Word	195
Find	188	PutAppend (>>>)	191	Write	196
Get (<<)	188	Read	193	WriteString	196
		ReadList	193		

\$Input

WMA link

\$Input

is the name of the stream from which input is currently being read.

>> \$Input

\$InputFileName

WMA link

\$InputFileName

is the name of the file from which input is currently being read.

While in interactive mode, \$InputFileName is “”.

>> \$InputFileName

Character

WMA link

Character

is a data type for Read.

Close

WMA link

```
Close[stream]  
  closes an input or output stream.
```

```
>> Close[StringToStream["123abc"]]  
String  
>> file=Close[OpenWrite[]]  
/tmp/tmp1ae0ybk7
```

Closing a file doesn't delete it from the filesystem

```
>> DeleteFile[file];
```

EndOfFile

WMA link

```
EndOfFile  
  is returned by Read when the end of an input stream is reached.
```

Expression

WMA link

```
Expression  
  is a data type for Read.
```

For information about underlying data structure Expression (a kind of M-expression) that is central in evaluation, see: AST, M-Expression, General List same thing.

FilePrint

WMA link

```
FilePrint[file]  
  prints the raw contents of file.
```

Find

WMA link

```
Find[stream, text]  
  find the first line in stream that contains text.
```

```
>> stream = OpenRead["ExampleData/EinsteinSzilLetter.txt",  
CharacterEncoding->"UTF8"];
```

```

>> Find[stream, "uranium"]
in manuscript, leads me to expect that the element uranium may be turned into

>> Find[stream, "uranium"]
become possible to set up a nuclear chain reaction in a large mass of uranium,

>> Close[stream]
ExampleData/EinsteinSzilLetter.txt

>> stream = OpenRead["ExampleData/EinsteinSzilLetter.txt",
CharacterEncoding->"UTF8"];

>> Find[stream, {"energy", "power"} ]
a new and important source of energy in the immediate future. Certain aspects

>> Find[stream, {"energy", "power"} ]
by which vast amounts of power and large quantities of new radium-like

>> Close[stream]
ExampleData/EinsteinSzilLetter.txt

```

Get (<<)

WMA link

```

<<name
      reads a file and evaluates each expression, returning only the last one.
Get [name, Trace->True]
      Runs Get tracing each line before it is evaluated.

```

```

>> filename = $TemporaryDirectory <> "/example_file";

>> Put[x + y, filename]

>> Get[filename]
x + y

>> filename = $TemporaryDirectory <> "/example_file";

>> Put[x + y, 2x^2 + 4z!, Cos[x] + I Sin[x], filename]

>> Get[filename]
Cos[x] + ISin[x]

>> DeleteFile[filename]

```

InputStream

WMA link

```

InputStream[name, n]
      represents an input stream for functions such as Read or Find.

```

`StringToStream` opens an input stream:

```
>> stream = StringToStream["Mathics is cool!"]
InputStream [String,3]

>> Close[stream]
String
```

Number

WMA link

`Number`

is a data type for Read.

OpenAppend

WMA link

`OpenAppend[“file”]`

opens a file and returns an OutputStream to which writes are appended.

```
>> OpenAppend[]
OutputStream [/tmp/tmp0hi3hmgy,3]
```

OpenRead

WMA link

`OpenRead[“file”]`

opens a file and returns an InputStream.

```
>> OpenRead["ExampleData/EinsteinSzilLetter.txt", CharacterEncoding->"UTF8"]

InputStream [ExampleData/EinsteinSzilLetter.txt,4]

>> Close[OpenRead["https://raw.githubusercontent.com/Mathics3/mathics-
core/master/README.rst"]];
```

OpenWrite

WMA link

`OpenWrite[“file”]`

opens a file and returns an OutputStream.

```
>> OpenWrite[]
OutputStream [/tmp/tmp46rnxgfg,5]
```

OutputStream

WMA link

```
OutputStream[name, n]
represents an output stream.
```

By default, the list of Streams normally OutputStream entries for stderr and stdout

```
>> Streams[]
{InputStream[stdin, 0], OutputStream[stdout, 1], OutputStream[
stderr, 2], OutputStream[/tmp/tmp0hi3hmgy, 3], InputStream[
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 4], OutputStream[
/tmp/tmp46rnxgfg, 5]}
```

Put (>>)

WMA link

```
expr >> filename
write expr to a file.
Put[expr1, expr2, ..., filename]
write a sequence of expressions to a file.
```

```
>> Put[40!, fortyfactorial]
fortyfactorial is not string, InputStream[], or OutputStream[]
815915283247897734345611269596115894272000000000>fortyfactorial

>> filename = $TemporaryDirectory <> "/fortyfactorial";

>> Put[40!, filename]

>> FilePrint[filename]
815915283247897734345611269596115894272000000000

>> Get[filename]
815915283247897734345611269596115894272000000000

>> DeleteFile[filename]

>> filename = $TemporaryDirectory <> "/fiftyfactorial";

>> Put[10!, 20!, 30!, filename]

>> FilePrint[filename]
3628800
2432902008176640000
265252859812191058636308480000000

>> DeleteFile[filename]

=
>> filename = $TemporaryDirectory <> "/example_file";

>> Put[x + y, 2x^2 + 4z!, Cos[x] + I Sin[x], filename]
```

```
>> FilePrint[filename]
x + y
2*x^2 + 4*z!
Cos[x] + I*Sin[x]

>> DeleteFile[filename]
```

PutAppend (>>>)

WMA link

expr >>> *filename*
append *expr* to a file.
PutAppend[*expr1*, *expr2*, ..., \$“*filename*”]
write a sequence of expressions to a file.

```
>> Put[50!, "factorials"]

>> FilePrint["factorials"]
304140932017133780436126081660647688443776415689605120000000000000

>> PutAppend[10!, 20!, 30!, "factorials"]

>> FilePrint["factorials"]
30414093201713378043612608166064768844377641568960512000000000000
3628800
2432902008176640000
265252859812191058636308480000000

>> 60! >>> "factorials"

>> FilePrint["factorials"]
30414093201713378043612608166064768844377641568960512000000000000
3628800
2432902008176640000
265252859812191058636308480000000
83209871127413901442763411832233643807541726063612459524492776964096000000000000000

>> "string" >>> factorials

>> FilePrint["factorials"]
30414093201713378043612608166064768844377641568960512000000000000
3628800
2432902008176640000
265252859812191058636308480000000
8320987112741390144276341183223364380754172606361245952449277696409600000000000000000
```

Read

WMA link

- Read [*stream*]
reads the input stream and returns one expression.
- Read [*stream*, *type*]
reads the input stream and returns an object of the given type.
- Read [*stream*, *type*]
reads the input stream and returns an object of the given type.
- Read [*stream*, Hold[*Expression*]]
reads the input stream for an *Expression* and puts it inside *Hold*.

is one of:

- Byte
- Character
- Expression
- HoldExpression
- Number
- Real
- Record
- String
- Word

```

>> stream = StringToStream["abc123"];

>> Read[stream, String]
abc123

>> stream = StringToStream["abc 123"];

>> Read[stream, Word]
abc

>> Read[stream, Word]
123

>> stream = StringToStream["123, 4"];

>> Read[stream, Number]
123

>> Read[stream, Number]
4

>> stream = StringToStream["2+2\n2+3"];

```

Read with a Hold[Expression] returns the expression it reads unevaluated so it can be later inspected and evaluated:

```

>> Read[stream, Hold[Expression]]
Hold[2 + 2]

>> Read[stream, Expression]
5

>> Close[stream];

```

Reading a comment however will return the empty list:

```

>> stream = StringToStream["(* ::Package:: *)"];

>> Read[stream, Hold[Expression]]
{}

>> Close[stream];

>> stream = StringToStream["123 abc"];

>> Read[stream, {Number, Word}]
{123,abc}

```

Multiple lines:

```
>> stream = StringToStream["\"Tengo una\nvaca lechera.\""]; Read[stream]
Tengo una
vaca lechera.
```

ReadList

WMA link

```
ReadList["file"]
    Reads all the expressions until the end of file.
ReadList["file", type]
    Reads objects of a specified type until the end of file.
ReadList["file", {type1, type2, ...}]
    Reads a sequence of specified types until the end of file.
```

```
>> ReadList[StringToStream["a 1 b 2"], {Word, Number}]
{{a,1},{b,2}}

>> stream = StringToStream["\"abc123\"];

>> ReadList[stream]
{abc123}

>> InputForm[%]
{"abc123"}
```

Record

WMA link

```
Record
    is a data type for Read.
```

SetStreamPosition

WMA link

```
SetStreamPosition[stream, n]
    sets the current position in a stream.
```

```
>> stream = StringToStream["Mathics is cool!"]
InputStream [String, 19]

>> SetStreamPosition[stream, 8]
8

>> Read[stream, Word]
is
```

```
>> SetStreamPosition[stream, Infinity]  
16
```

Skip

WMA link

```
Skip[stream, type]  
    skips ahead in an input stream by one object of the specified type.  
Skip[stream, type, n]  
    skips ahead in an input stream by n objects of the specified type.
```

```
>> stream = StringToStream["a b c d"];  
  
>> Read[stream, Word]  
a  
  
>> Skip[stream, Word]  
  
>> Read[stream, Word]  
c  
  
>> stream = StringToStream["a b c d"];  
  
>> Read[stream, Word]  
a  
  
>> Skip[stream, Word, 2]  
  
>> Read[stream, Word]  
d
```

StreamPosition

WMA link

```
StreamPosition[stream]  
    returns the current position in a stream as an integer.
```

```
>> stream = StringToStream["Mathics is cool!"]  
InputStream [String, 23]  
  
>> Read[stream, Word]  
Mathics  
  
>> StreamPosition[stream]  
7
```

Streams

WMA link

Streams []
returns a list of all open streams.

```
>> Streams []
{InputStream [stdin, 0], OutputStream [stdout, 1], OutputStream [
stderr, 2], OutputStream [/tmp/tmp0hi3hmgy, 3], InputStream [
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 4], OutputStream [
/tmp/tmp46rnxgfg, 5], InputStream [String, 6], InputStream [
String, 7], InputStream [String, 8], InputStream [String, 9], InputStream [
String, 10], InputStream [String, 11], InputStream [String, 12], InputStream [
String, 13], InputStream [String, 14], InputStream [String, 15], InputStream [
String, 16], InputStream [String, 17], InputStream [String, 18], InputStream [
String, 19], InputStream [String, 20], InputStream [String, 21], InputStream [
String, 22], InputStream [String, 23], OutputStream [/tmp/tmp4wt98oxv, 24]}
```

```
>> Streams ["stdout"]
{OutputStream [stdout, 1]}
```

StringToString

WMA link

StringToString[string]
converts a *string* to an open input stream.

```
>> strm = StringToString["abc 123"]
InputStream [String, 25]
```

Word

WMA link

Word
is a data type for Read.

Write

WMA link

Write[channel, expr1, expr2, ...]
writes the expressions to the output channel followed by a newline.

```
>> stream = OpenWrite[]
OutputStream [/tmp/tmppyn8jvk0, 26]
>> Write[stream, 10 x + 15 y ^ 2]
```

```

>> Write[stream, 3 Sin[z]]
>> Close[stream];
>> stream = OpenRead[%];
>> ReadList[stream]
{10x + 15y2, 3Sin[z]}

```

WriteString

WMA link

```
WriteString[stream, $str1, str2, ... ]
writes the strings to the output stream.
```

```

>> stream = OpenWrite[];
>> WriteString[stream, "This is a test 1"]
>> WriteString[stream, "This is also a test 2"]
>> pathname = Close[stream];
>> FilePrint[%]
This is a test 1This is also a test 2
>> stream = OpenWrite[];
>> WriteString[stream, "This is a test 1", "This is also a test 2"]
>> pathname = Close[stream]
/tmp/tmpd3bturtt
>> FilePrint[%]
This is a test 1This is also a test 2

```

If stream is the string “stdout” or “stderr”, writes to the system standard output/ standard error channel:

```
>> WriteString["stdout", "Hola"]
```

17. Filesystem Operations

Contents

\$BaseDirectory	197	DeleteDirectory	201	FileNameSplit	206
\$HomeDirectory	197	DeleteFile	201	FileNameTake	206
\$InitialDirectory	198	Directory	201	FileNames	206
\$InstallationDirectory .	198	DirectoryName	202	FileType	207
\$OperatingSystem	198	DirectoryQ	202	FindFile	207
\$Path	198	DirectoryStack	202	FindList	207
\$PathnameSeparator .	199	ExpandFileName	202	Hash	208
\$RootDirectory	199	File	202	Needs	208
\$TemporaryDirectory .	199	FileBaseName	203	ParentDirectory	208
\$UserBaseDirectory . .	199	FileByteCount	203	RenameDirectory	208
AbsoluteFileName	199	FileDate	203	RenameFile	209
CopyDirectory	200	FileExistsQ	204	ResetDirectory	209
CopyFile	200	FileExtension	204	SetDirectory	209
CreateDirectory	200	FileHash	204	SetFileDate	210
CreateFile	200	FileInfo	205	ToFileName	210
CreateTemporary	200	FileNameDepth	205	URLSave	210
		FileNameJoin	205		

\$BaseDirectory

WMA link

```
$BaseDirectory
    returns the folder where user configurations are stored.
```

```
>> $BaseDirectory
    /src/external-vcs/github/Mathics3/mathics-core/mathics
```

\$HomeDirectory

WMA link

```
$HomeDirectory
    returns the users HOME directory.
```

```
>> $HomeDirectory
    /home/rocky
```

\$InitialDirectory

WMA link

\$InitialDirectory
returns the directory from which *Mathics3* was started.

```
>> $InitialDirectory  
/src/external-vcs/github/Mathics3/mathics-core/mathics
```

\$InstallationDirectory

WMA link

\$InstallationDirectory
returns the top-level directory in which *Mathics3* was installed.

```
>> $InstallationDirectory  
/src/external-vcs/github/Mathics3/mathics-core/mathics
```

\$OperatingSystem

WMA link

\$OperatingSystem
gives the type of operating system running Mathics.

```
>> $OperatingSystem  
Unix
```

\$Path

WMA link

\$Path
returns the list of directories to search when looking for a file.

```
>> $Path  
{., /home/rocky, /src/external-vcs/github/Mathics3/mathics-core/mathics/data, /src/external-vcs/github/Ma
```

\$PathnameSeparator

WMA link

\$PathnameSeparator
returns a string for the separator in paths.

```
>> $PathnameSeparator  
/
```

\$RootDirectory

WMA link

```
$RootDirectory  
returns the system root directory.
```

```
>> $RootDirectory  
/
```

\$TemporaryDirectory

WMA link

```
$TemporaryDirectory  
returns the directory used for temporary files.
```

```
>> $TemporaryDirectory  
/tmp
```

\$UserBaseDirectory

WMA link

```
$UserBaseDirectory  
returns the folder where user configurations are stored.
```

```
>> $RootDirectory  
/
```

AbsoluteFileName

WMA link

```
AbsoluteFileName["name"]  
returns the absolute version of the given filename.
```

```
>> AbsoluteFileName["ExampleData/sunflowers.jpg"]  
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/sunflowers.jpg
```

CopyDirectory

WMA link

```
CopyDirectory["dir1", "dir2"]  
copies directory dir1 to dir2.
```

CopyFile

WMA link

```
CopyFile["file1", "file2"]  
copies file1 to file2.
```

```
>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers.jpg"]  
MathicsSunflowers.jpg  
>> DeleteFile["MathicsSunflowers.jpg"]
```

CreateDirectory

WMA link

```
CreateDirectory["dir"]  
creates a directory called dir.  
CreateDirectory[]  
creates a temporary directory.
```

```
>> dir = CreateDirectory[]  
/tmp/m0lzx9goj
```

CreateFile

WMA link

```
CreateFile[{"filename"}]  
Creates a file named "filename" temporary file, but do not open it.  
CreateFile[]  
Creates a temporary file, but do not open it.
```

CreateTemporary

WMA link

```
CreateTemporary[]  
Creates a temporary file, but do not open it.
```

DeleteDirectory

WMA link

```
DeleteDirectory["dir"]
deletes a directory called dir.
```

```
>> dir = CreateDirectory[]
/tmp/m0dz1k91z

>> DeleteDirectory[dir]

>> DirectoryQ[dir]
False
```

DeleteFile

WMA link

```
Delete["file"]
 deletes file.
Delete[{file1, "file2", ...}]
 deletes a list of files.
```

```
>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers.jpg"];

>> DeleteFile["MathicsSunflowers.jpg"]

>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers1.jpg"];

>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers2.jpg"];

>> DeleteFile[{"MathicsSunflowers1.jpg", "MathicsSunflowers2.jpg"}]
```

Directory

WMA link

```
Directory[]
 returns the current working directory.
```

```
>> Directory[]
 /src/external-vcs/github/Mathics3/mathics-core/mathics
```

DirectoryName

WMA link

```
DirectoryName["name"]
 extracts the directory name from a filename.
```

```
>>DirectoryName["a/b/c"]
a/b
```

```
>> DirectoryName["a/b/c", 2]
a
```

DirectoryQ

WMA link

```
DirectoryQ["name"]
returns True if the directory called name exists and False otherwise.
```

```
>> DirectoryQ["ExampleData/"]
True
>> DirectoryQ["ExampleData/MythicalSubdir/"]
False
```

DirectoryStack

WMA link

```
DirectoryStack[]
returns the directory stack.
```

```
>> DirectoryStack[]
{/src/external-vcs/github/Mathics3/mathics-core/mathics}
```

ExpandFileName

WMA link

```
ExpandFileName["name"]
expands name to an absolute filename for your system.
```

```
>> ExpandFileName["ExampleData/sunflowers.jpg"]
{/src/external-vcs/github/Mathics3/mathics-core/mathics/ExampleData/sunflowers.jpg}
```

File

WMA link

```
File["file"]
is a symbolic representation of an element in the local file system.
```

FileBaseName

WMA link

```
FileBaseName["file"]
gives the base name for the specified file name.
```

```
>> FileBaseName["file.txt"]
file

>> FileBaseName["file.tar.gz"]
file.tar
```

FileByteCount

WMA link

```
FileByteCount[file]
returns the number of bytes in file.
```

```
>> FileByteCount["ExampleData/sunflowers.jpg"]
142286
```

FileDate

WMA link

```
FileDate[file, types]
returns the time and date at which the file was last modified.
```

```
>> FileDate["ExampleData/sunflowers.jpg"]
{2123, 1, 16, 3, 50, 39.9396}

>> FileDate["ExampleData/sunflowers.jpg", "Access"]
{2123, 2, 26, 13, 19, 30.2059}

>> FileDate["ExampleData/sunflowers.jpg", "Creation"]
Missing [NotApplicable]

>> FileDate["ExampleData/sunflowers.jpg", "Change"]
{2123, 1, 16, 3, 50, 39.9396}

>> FileDate["ExampleData/sunflowers.jpg", "Modification"]
{2123, 1, 16, 3, 50, 39.9396}

>> FileDate["ExampleData/sunflowers.jpg", "Rules"]
{Access -> {2123, 2, 26, 13, 19, 30.2059}, Creation -> Missing [
    NotApplicable], Change -> {2123, 1, 16, 3, 50, 39.9396}, Modification
 -> {2123, 1, 16, 3, 50, 39.9396}}
```

FileExistsQ

WMA link

```
FileExistsQ["file"]
  returns True if file exists and False otherwise.
```

```
>> FileExistsQ["ExampleData/sunflowers.jpg"]
True
>> FileExistsQ["ExampleData/sunflowers.png"]
False
```

FileExtension

WMA link

```
FileExtension["file"]
  gives the extension for the specified file name.
```

```
>> FileExtension["file.txt"]
txt
>> FileExtension["file.tar.gz"]
gz
```

FileHash

WMA link

```
FileHash[file]
  returns an integer hash for the given file.
FileHash[file, type]
  returns an integer hash of the specified type for the given file.
  The types supported are "MD5", "Adler32", "CRC32", "SHA", "SHA224", "SHA256",
  "SHA384", and "SHA512".
FileHash[file, type, format]
  gives a hash code in the specified format.
```

```
>> FileHash["ExampleData/sunflowers.jpg"]
109937059621979839952736809235486742106
>> FileHash["ExampleData/sunflowers.jpg", "MD5"]
109937059621979839952736809235486742106
>> FileHash["ExampleData/sunflowers.jpg", "Adler32"]
1607049478
>> FileHash["ExampleData/sunflowers.jpg", "SHA256"]
111619807552579450300684600241129773909359865098672286468229443390003894913065
```

FileInfo

WMA link

```
FileInformation["file"]
  returns information about file.
```

This function is totally undocumented in MMA!

```
>> FileInformation["ExampleData/sunflowers.jpg"]
{File
 -> /src/external-vcs/github/Mathics3/mathics-core/mathics/ExampleData/sunflowers.jpg, FileType
 -> File, ByteCount-> 142286, Date-> 7.0385*^9}
```

FileNameDepth

WMA link

```
FileNameDepth["name"]
  gives the number of path parts in the given filename.
```

```
>> FileNameDepth["a/b/c"]
3
>> FileNameDepth["a/b/c/"]
3
```

FileNameJoin

WMA link

```
FileNameJoin[{ "dir_1", "dir_2", ...}]
  joins the dir_i together into one path.
FileNameJoin[..., OperatingSystem->{"os"}]
  yields a file name in the format for the specified operating system. Possible choices are "Windows", "MacOSX", and "Unix".
```

```
>> FileNameJoin[{"dir1", "dir2", "dir3"}]
dir1/dir2/dir3
>> FileNameJoin[{"dir1", "dir2", "dir3"}, OperatingSystem -> "Unix"]
dir1/dir2/dir3
>> FileNameJoin[{"dir1", "dir2", "dir3"}, OperatingSystem -> "Windows"]
dir1\dir2\dir3
```

FileNameSplit

WMA link

```
FileNameSplit["filenames"]
  splits a filename into a list of parts.
```

```
>> FileNameSplit["example/path/file.txt"]
{example, path, file.txt}
```

FileNameTake

WMA link

```
FileNameTake["file"]
    returns the last path element in the file name name.
FileNameTake["file", n]
    returns the first n path elements in the file name name.
FileNameTake["file", $-n$]
    returns the last n path elements in the file name name.
```

FileNames

WMA link

```
FileNames[]
    Returns a list with the filenames in the current working folder.
FileNames[form]
    Returns a list with the filenames in the current working folder that matches with form.
FileNames[{form_1, form_2, ...}]
    Returns a list with the filenames in the current working folder that matches with one of form_1, form_2, ...
FileNames[{form_1, form_2, ...}, {dir_1, dir_2, ...}]
    Looks into the directories dir_1, dir_2, ...
FileNames[{form_1, form_2, ...}, {dir_1, dir_2, ...}]
    Looks into the directories dir_1, dir_2, ...
FileNames[{forms, dirs, n}
    Look for files up to the level n.
```

```
>> SetDirectory[$InstallationDirectory <> "/autoload"];
>> FileNames["*.m", "formats"]//Length
0
>> FileNames["*.m", "formats", 3]//Length
14
>> FileNames["*.m", "formats", Infinity]//Length
14
```

FileType

WMA link

```
FileType["file"]
    gives the type of a file, a string. This is typically File, Directory or None.
```

```

>> FileType["ExampleData/sunflowers.jpg"]
File

>> FileType["ExampleData"]
Directory

>> FileType["ExampleData/nonexistent"]
None

```

FindFile

WMA link

```

FindFile[name]
  searches $Path for the given filename.

```

```

>> FindFile["ExampleData/sunflowers.jpg"]
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/sunflowers.jpg

>> FindFile["VectorAnalysis`"]
/src/external-vcs/github/Mathics3/mathics-core/mathics/packages/VectorAnalysis/Kernel/init.m

>> FindFile["VectorAnalysis`VectorAnalysis`"]
/src/external-vcs/github/Mathics3/mathics-core/mathics/packages/VectorAnalysis/VectorAnalysis.m

```

FindList

WMA link

```

FindList[file, text]
  returns a list of all lines in file that contain text.
FindList[file, {text1, text2, ...}]
  returns a list of all lines in file that contain any of the specified string.
FindList[{file1, file2, ...}, ...]
  returns a list of all lines in any of the filei that contain the specified strings.

```

```

>> stream = FindList["ExampleData/EinsteinSzilLetter.txt", "uranium"];

>> FindList["ExampleData/EinsteinSzilLetter.txt", "uranium", 1]
{in manuscript, leads me to expect that the element uranium may be turned into}

```

Hash

Hash function (WMA link)

Hash [*expr*]

returns an integer hash for the given *expr*.

Hash [*expr*, *type*]

returns an integer hash of the specified *type* for the given *expr*.

The types supported are "MD5", "Adler32", "CRC32", "SHA", "SHA224", "SHA256", "SHA384", and "SHA512".

Hash [*expr*, *type*, *format*]

Returns the hash in the specified format.

```
> Hash["The Adventures of Huckleberry Finn"] = 213425047836523694663619736686226550816
```

```
> Hash["The Adventures of Huckleberry Finn", "SHA256"] = 950926495945903842880571834086092549189343518116698
```

```
> Hash[1/3] = 56073172797010645108327809727054836008
```

```
> Hash[{a, b, {c, d, e, f}}] = 135682164776235407777080772547528225284
```

```
> Hash[SomeHead[3.1415]] = 58042316473471877315442015469706095084
```

```
>> Hash[{a, b, c}, "xyzstr"]
```

```
Hash [{a, b, c}, xyzstr, Integer]
```

Needs

WMA link

Needs ["context'"]

loads the specified context if not already in \$Packages .

```
>> Needs["VectorAnalysis'"]
```

ParentDirectory

WMA link

ParentDirectory []

returns the parent of the current working directory.

ParentDirectory ["*dir*"]

returns the parent *dir*.

```
>> ParentDirectory[]
```

```
/src/external-vcs/github/Mathics3/mathics-core/mathics
```

RenameDirectory

WMA link

RenameDirectory ["*dir1*'', "*dir2*"]

renames directory *dir1* to *dir2*.

RenameFile

WMA link

```
RenameFile["file1", "file2"]  
renames file1 to file2.
```

```
>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers.jpg"]  
MathicsSunflowers.jpg  
  
>> RenameFile["MathicsSunflowers.jpg", "MathicsSunnyFlowers.jpg"]  
MathicsSunnyFlowers.jpg  
  
>> DeleteFile["MathicsSunnyFlowers.jpg"]
```

ResetDirectory

WMA link

```
ResetDirectory[]  
pops a directory from the directory stack and returns it.
```

```
>> ResetDirectory[]  
/src/external-vcs/github/Mathics3/mathics-core/mathics/autoload
```

SetDirectory

WMA link

```
SetDirectory[dir]  
sets the current working directory to dir.
```

```
>> SetDirectory[]  
/home/rocky
```

SetFileDialog

WMA link

```
SetFileDialog["file"]  
set the file access and modification dates of file to the current date.  
SetFileDialog["file", date]  
set the file access and modification dates of file to the specified date list.  
SetFileDialog["file", date, type]  
set the file date of file to the specified date list. The "type" can be one of "Access", "Creation",  
"Modification", or All.
```

Create a temporary file (for example purposes)

```
>> tmpfilename = $TemporaryDirectory <> "/tmp0";
```

```
>> Close[OpenWrite[tmpfilename]];
>> SetFileDate[tmpfilename, {2002, 1, 1, 0, 0, 0.}, "Access"];
>> FileDate[tmpfilename, "Access"]
{2002,1,1,0,0,0.}
```

ToFileName

WMA link

```
ToFileName[{"dir_1", "dir_2", ...}]
joins the diri together into one path.
```

ToFileName has been superseded by FileNameJoin.

```
>> ToFileName[{"dir1", "dir2"}, "file"]
dir1/dir2/file
>> ToFileName["dir1", "file"]
dir1/file
>> ToFileName[{"dir1", "dir2", "dir3"}]
dir1/dir2/dir3
```

URLSave

WMA link

```
URLSave['url']
Save "url" in a temporary file.
URLSave['url', 'filename']
Save "url" in filename.
```

18. Forms of Input and Output

A *Form* format specifies the way Mathics Expression input is read or output written.
The variable `$OutputForms` of section 18 has a list of Forms defined.
See also WMA link.

Contents

Form variables	211	InputForm	213	TableForm	218
\$OutputForms . . .	211	MakeBoxes	214	TeXForm	218
\$PrintForms	212	MathMLForm . . .	214	TraditionalForm .	218
Forms which appear in		MatrixForm	214	Forms which are not in	
\$OutputForms	212	NumberForm	215	\$OutputForms	218
BaseForm	212	OutputForm	215	FormBaseClass . . .	218
FormBaseClass . . .	212	PythonForm	215	MakeBoxes	219
FullForm	213	RowBox	216	RowBox	219
GridBox	213	StandardForm . . .	216	StringForm	219
		SympyForm	216		

Form variables

Form variables

\$OutputForms

`$OutputForms`

contains the list of all output forms. It is updated automatically when new `OutputForms` are defined by setting format values.

```
>> $OutputForms
```

```
{MathMLForm, TraditionalForm, FullForm, StandardForm, MatrixForm, SympyForm, PythonForm, InputForm, Ta
```

\$PrintForms

`$PrintForms`

contains the list of basic print forms. It is updated automatically when new `PrintForms` are defined by setting format values.

```
>> $PrintForms
```

```
{MathMLForm, TraditionalForm, FullForm, StandardForm, SympyForm, PythonForm, InputForm, TeXForm, MyF
```

Suppose now that we want to add a new format `MyForm`. Initially, it does not belong to `$PrintForms`:

```
>> MemberQ[$PrintForms, MyForm]
```

Now, let's define a format rule:

```
>> Format[MyForm[F[x_]]]:= "F<<" <> ToString[x] <> ">>"  
>> Format[F[x_], MyForm]:= MyForm[F[x]]
```

Now, the new format belongs to the \$PrintForms list

```
>> MemberQ[$PrintForms, MyForm]
```

Forms which appear in \$OutputForms.

Forms which appear in \$OutputForms.

BaseForm

```
BaseForm[expr, n]  
prints numbers in expr in base n.
```

```
>> BaseForm[33, 2]  
>> BaseForm[234, 16]  
>> BaseForm[12.3, 2]  
>> BaseForm[-42, 16]  
>> BaseForm[x, 2]  
x  
>> BaseForm[12, 3] // FullForm  
BaseForm[12,3]
```

Bases must be between 2 and 36:

```
>> BaseForm[12, -3]  
>> BaseForm[12, 100]
```

FormBaseClass

Base class for a Mathics Form.

All Forms should subclass this.

FullForm

WMA link

```
FullForm[expr]  
displays the underlying form of expr.
```

```
>> FullForm[a + b * c]  
Plus[a, Times[b, c]]
```

```

>> FullForm[2/3]
Rational[2,3]

>> FullForm["A string"]
"A string"

```

GridBox

GridBox[{{...}, {...}}]
 is a box construct that represents a sequence of boxes arranged in a grid. # »

```
MathMLForm[TableForm[{{a,b},{c,d}}]] # = ...
```

InputForm

WMA link

InputForm [*expr*]
 displays *expr* in an unambiguous form suitable for input.

```

>> InputForm[a + b * c]
a + b * c

>> InputForm["A string"]
"A string"

>> InputForm[f'[x]]
Derivative[1][f][x]

>> InputForm[Derivative[1, 0][f][x]]
Derivative[1,0][f][x]

```

MakeBoxes

WMA link

MakeBoxes [*expr*]
 is a low-level formatting primitive that converts *expr* to box form, without evaluating it.
 $\backslash(x \backslash^ 2\backslash)$
 directly inputs box objects.

String representation of boxes

```

>> \(\(x \backslash^ 2\)\)
SuperscriptBox[x,2]

>> \(\(x \backslash_ 2\)\)
SubscriptBox[x,2]

>> \(\( a \backslash+ b \backslash% c\)\)
UnderoverscriptBox[a,b,c]

```

```

>> \(\ a \& b \% c\)
UnderoverscriptBox[a,c,b]

>> \(\ x \& y \)
OverscriptBox[x,y]

>> \(\ x \& y \)
UnderscriptBox[x,y]

```

MathMLForm

WMA link

```

MathMLForm[expr]
displays expr as a MathML expression.

```

```

>> MathMLForm[HoldForm[Sqrt[a3]]]
<math display="block"><msqrt> <msup><mi>a</mi>
<mn>3</mn></msup> </msqrt></math>

>> MathMLForm[\[Mu]]
<math display="block"><mi>\mu</mi></math>

# This can causes the TeX to fail # » MathMLForm[Graphics[Text[" $\mu$ "]]] # = ...
= ...

```

MatrixForm

WMA link

```

MatrixForm[m]
displays a matrix m, hiding the underlying list structure.

```

```

>> Array[a, {4,3}]//MatrixForm

$$\begin{pmatrix} a[1,1] & a[1,2] & a[1,3] \\ a[2,1] & a[2,2] & a[2,3] \\ a[3,1] & a[3,2] & a[3,3] \\ a[4,1] & a[4,2] & a[4,3] \end{pmatrix}$$


```

NumberForm

```

NumberForm[expr, n]
prints a real number expr with n-digits of precision.
NumberForm[expr, {n, f}]
prints with n-digits and f digits to the right of the decimal point.

```

```

>> NumberForm[N[Pi], 10]
3.141592654
>> NumberForm[N[Pi], {10, 5}]
3.14159

```

OutputForm

WMA link

```
OutputForm[expr]
    displays expr in a plain-text form.
```

```
>> OutputForm[f'[x]]
f'[x]

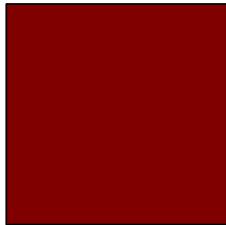
>> OutputForm[Derivative[1, 0][f][x]]
Derivative[1,0][f][x]
```

`OutputForm` is used by default:

```
>> OutputForm[{"A string", a + b}]
{A string, a + b}

>> {"A string", a + b}
{A string, a + b}

>> OutputForm[Graphics[Rectangle[]]]
```



PythonForm

```
PythonForm[expr]
    returns an approximate equivalent of expr in Python, when that is possible. We assume that
    Python has SymPy imported. No explicit import will be included in the result.
```

```
>> PythonForm[Infinity]
math.inf

>> PythonForm[Pi]
sympy.pi

>> E // PythonForm
sympy.E

>> {1, 2, 3} // PythonForm
[1, 2, 3]
```

RowBox

WMA link

```
RowBox[{...}]
```

is a box construct that represents a sequence of boxes arranged in a horizontal row.

StandardForm

WMA link

```
StandardForm[expr]
```

displays *expr* in the default form.

```
>> StandardForm[a + b * c]  
a + bc
```

```
>> StandardForm["A string"]
```

A string

```
>> f'[x]
```

$f'[x]$

SympyForm

```
SympyForm[expr]
```

returns an Sympy *expr* in Python. Sympy is used internally to implement a number of Mathics functions, like Simplify.

```
>> SympyForm[Pi^2]
```

π^{**2}

```
>> E^2 + 3E // SympyForm
```

$\exp(2) + 3^E$

TableForm

WMA link

```
TableForm[expr]
```

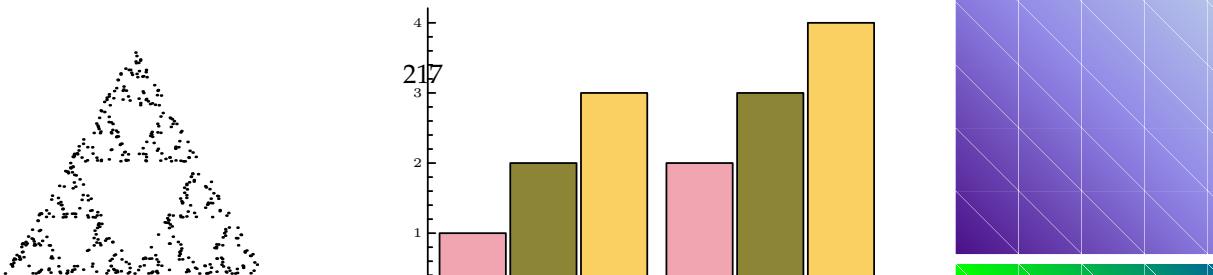
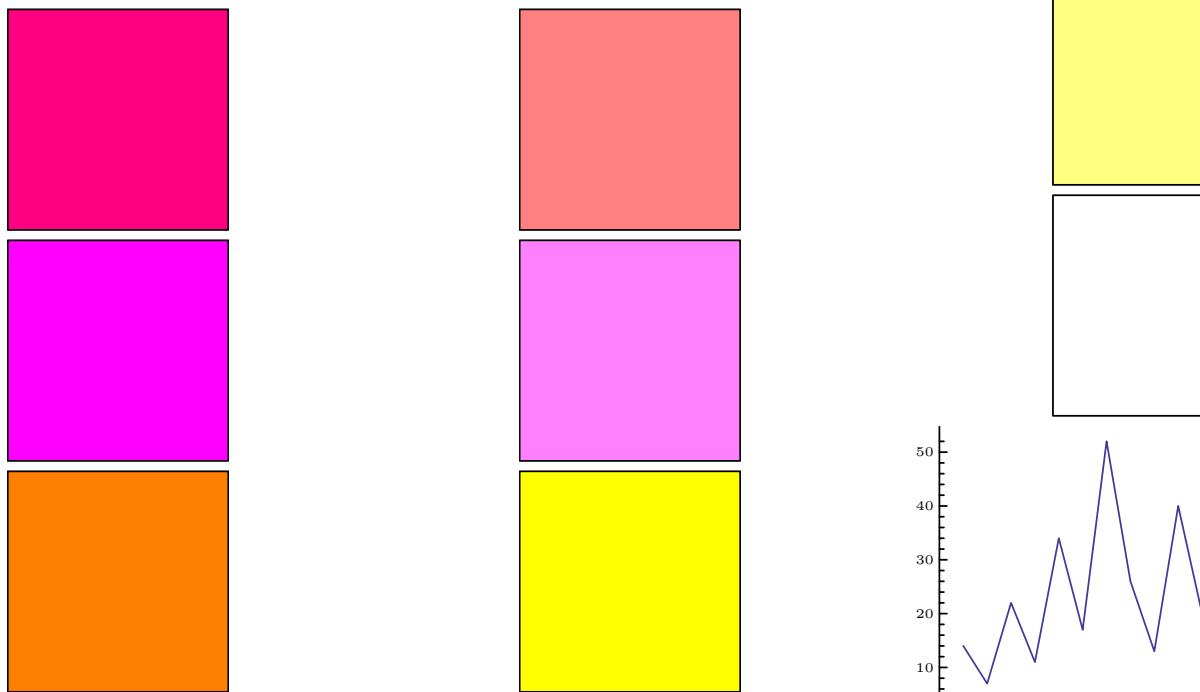
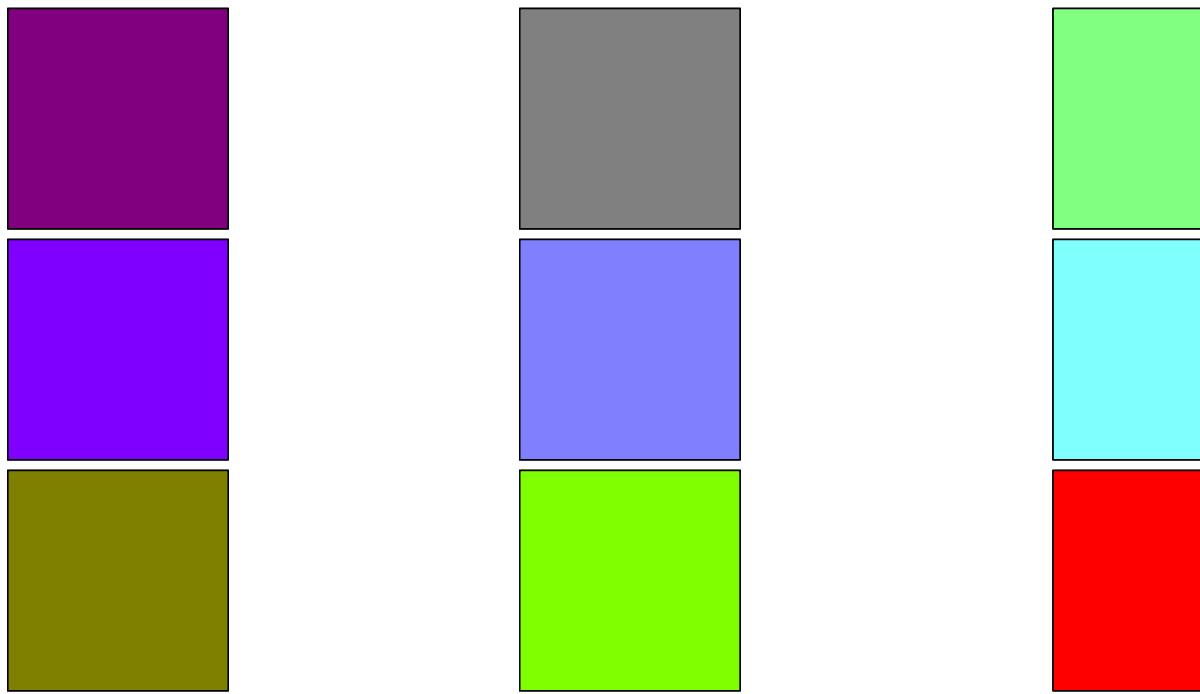
displays *expr* as a table.

```
>> TableForm[Array[a, {3,2}],TableDepth->1]
```

{ $a[1,1], a[1,2]$ }
{ $a[2,1], a[2,2]$ }
{ $a[3,1], a[3,2]$ }

A table of Graphics:

```
>> Table[Style[Graphics[{EdgeForm[{Black}], RGBColor[r,g,b], Rectangle[]}], ImageSizeMultipliers->\{0.2, 1\}], \{r,0,1,1/2\}, \{g,0,1,1/2\}, \{b,0,1,1/2\}] // TableForm
```



TeXForm

WMA link

```
TeXForm[expr]
    displays expr using TeX math mode commands.
```

```
>> TeXForm[HoldForm[Sqrt[a^3]]]
\sqrt{a^3}
```

TraditionalForm

WMA link

```
TraditionalForm[expr]
    displays expr in a format similar to the traditional mathematical notation, where function evaluations are represented by brackets instead of square brackets.
```

Forms which are not in \$OutputForms

Forms which are not in \$OutputForms

FormBaseClass

Base class for a Mathics Form.

All Forms should subclass this.

MakeBoxes

WMA link

```
MakeBoxes[expr]
    is a low-level formatting primitive that converts expr to box form, without evaluating it.
\(\dots\)
    directly inputs box objects.
```

String representation of boxes

```
>> \(\mathbf{x}^2\)
SuperscriptBox[x, 2]

>> \(\mathbf{x}_2\)
SubscriptBox[x, 2]

>> \(\mathbf{a} + \mathbf{b} \% \mathbf{c}\)
UnderoverscriptBox[a, b, c]

>> \(\mathbf{a} \& \mathbf{b} \% \mathbf{c}\)
UnderoverscriptBox[a, c, b]

>> \(\mathbf{x} \& \mathbf{y}\)
OverscriptBox[x, y]
```

```
>> \(\mathbf{x} \mathbf{+} \mathbf{y}\)  
UnderscriptBox [x,y]
```

RowBox

WMA link

RowBox[{...}]
is a box construct that represents a sequence of boxes arranged in a horizontal row.

StringForm

WMA link

StringForm[str, expr1, expr2, ...]
displays the string str, replacing placeholders in str with the corresponding expressions.

```
>> StringForm["`1` bla `2` blub `` bla `2`", a, b, c]  
a bla b blub c bla b
```

19. Function Application

Contents

Function (&)	221	Slot	221	SlotSequence	222
------------------------	-----	----------------	-----	------------------------	-----

Function (&)

```
Function[body]
body &
represents a pure function with parameters #1, #2, etc.
Function[{x1, x2, ...}, body]
represents a pure function with parameters x1, x2, etc.
Function[{x1, x2, ...}, body, attr]
assume that the function has the attributes attr.
```

```
>> f := # ^ 2 &
>> f[3]
9
>> #^3& /@ {1, 2, 3}
{1, 8, 27}
>> #1+#2&[4, 5]
9
```

You can use Function with named parameters:

```
>> Function[{x, y}, x * y][2, 3]
6
```

Parameters are renamed, when necessary, to avoid confusion:

```
>> Function[{x}, Function[{y}, f[x, y]]][y]
Function [{y$},f [y,y$]]
>> Function[{y}, f[x, y]] /. x->y
Function [{y},f [y,y]]
>> Function[y, Function[x, y^x]][x][y]
xy
>> Function[x, Function[y, x^y]][x][y]
xy
```

Slots in inner functions are not affected by outer function application:

```
>> g[#] & [h[#]] & [5]
g[h[5]]
```

In the evaluation process, the attributes associated with an Expression are determined by its Head. If the Head is also a non-atomic Expression, in general, no Attribute is assumed. In particular, it is what happens when the head of the expression has the form:

“Function[*body*]” or: “Function[*vars, body*]”

```
>> h := Function[{x}, Hold[1+x]]
```

```
>> h[1 + 1]
Hold[1 + 2]
```

Notice that *Hold* in the body prevents the evaluation of \$1+x\$, but not the evaluation of \$1+1\$. To avoid that evaluation, of its arguments, the Head should have the attribute *HoldAll*. This behavior can be obtained by using the three arguments form version of this expression:

```
>> h:= Function[{x}, Hold[1+x], HoldAll]
```

```
>> h[1+1]
Hold[1 + (1 + 1)]
```

In this case, the attribute *HoldAll* is assumed, preventing the evaluation of the argument \$1+1\$ before passing it to the function body.

Slot

# <i>n</i>	represents the <i>n</i> th argument to a pure function.
#	is short-hand for #1.
#0	represents the pure function itself.

```
>> #
#1
```

Unused arguments are simply ignored:

```
>> {#1, #2, #3}&[1, 2, 3, 4, 5]
{1, 2, 3}
```

Recursive pure functions can be written using #0:

```
>> If[#1<=1, 1, #1 #0[#1-1]]& [10]
3628800
```

SlotSequence

##	is the sequence of arguments supplied to a pure function.
## <i>n</i>	starts with the <i>n</i> th argument.

```
>> Plus[##]& [1, 2, 3]
6
>> Plus[##2]& [1, 2, 3]
5
>> FullForm[##]
SlotSequence[1]
```

20. Functional Composition and Operator Forms

Functional Composition is a way to combine simple functions to build more complicated ones. Like the usual composition of functions in mathematics, the result of each function is passed as the argument of the next, and the result of the last one is the result of the whole.

The symbolic structure of Mathics3 makes it easy to create “operators” that can be composed and manipulated symbolically-forming “pipelines” of operations-and then applied to arguments.

Some built-in functions also directly support a “curried” form, in which they can immediately be given as symbolic operators.

Contents

Composition	223	Identity	224
-----------------------	-----	--------------------	-----

Composition

WMA link

```
Composition[f, g]
  returns the composition of two functions f and g.
```

```
>> Composition[f, g][x]
f[g[x]]

>> Composition[f, g, h][x, y, z]
f[g[h[x,y,z]]]

>> Composition[]
Identity

>> Composition[] [x]
x

>> Attributes[Composition]
{Flat, OneIdentity, Protected}

>> Composition[f, Composition[g, h]]
Composition[f,g,h]
```

Identity

WMA link

Identity[*x*]

is the identity function, which returns *x* unchanged.

>> **Identity**[*x*]

x

>> **Identity**[*x*, *y*]

Identity [*x, y*]

21. Functional Programming

Functional programming is a programming paradigm where programs are constructed by applying and composing functions.

It is made richer by expressions like $f[x]$ being treated as symbolic data.

This term is often used in contrast to Procedural programming.

Contents

Applying Functions to	Thread 229	Iteratively Applying
Lists 225	Function Application . 229	Functions 232
Apply (@@) 226	Function (&) . . . 230	FixedPoint 232
List 226	Slot 231	FixedPointList . . . 233
Map (/@) 227	SlotSequence . . . 231	Fold 233
MapAt 227	Functional	FoldList 234
MapIndexed 228	Composition and	Nest 234
MapThread 228	Operator Forms . 231	NestList 235
Scan 228	Composition . . . 232	NestWhile 235
	Identity 232	

Applying Functions to Lists

Applying Functions to Lists

Many computations can be conveniently specified in terms of applying functions in parallel to many elements in a list.

Many mathematical functions are automatically taken to be “listable”, so that they are always applied to every element in a list.

Apply (@@)

```
Apply[f, expr]
f @@ expr
    replaces the head of expr with f.
Apply[f, expr, levelspec]
    applies f on the parts specified by levelspec.
```

```
>> f @@ {1, 2, 3}
f[1,2,3]

>> Plus @@ {1, 2, 3}
6
```

The head of *expr* need not be List:

```
>> f @@ (a + b + c)
f[a,b,c]
```

Apply on level 1:

```
>> Apply[f, {a + b, g[c, d, e * f], 3}, {1}]  
{f[a,b],f[c,d,ef],3}
```

The default level is 0:

```
>> Apply[f, {a, b, c}, {0}]  
f[a,b,c]
```

Range of levels, including negative level (counting from bottom):

```
>> Apply[f, {{{{{a}}}}}, {2, -3}]  
{f[f[{a}]]}
```

Convert all operations to lists:

```
>> Apply[List, a + b * c ^ e * f[g], {0, Infinity}]  
{a,{b,{g},{c,e}}}
```

List

WMA link

```
List[e1, e2, ..., ei]  
{e1, e2, ..., ei}  
represents a list containing the elements e1...ei.
```

List is the head of lists:

```
>> Head[{1, 2, 3}]  
List
```

Lists can be nested:

```
>> {{a, b, {c, d}}}  
{a,b,{c,d}}
```

Map (/@)

```
Map[f, expr] or f /@ expr  
applies f to each part on the first level of expr.  
Map[f, expr, levelspec]  
applies f to each level specified by levelspec of expr.
```

```
>> f /@ {1, 2, 3}  
{f[1],f[2],f[3]}  
  
>> #^2& /@ {1, 2, 3, 4}  
{1,4,9,16}
```

Map f on the second level:

```
>> Map[f, {{a, b}, {c, d, e}}, {2}]  
{f[a],f[b],f[c],f[d],f[e]}
```

Include heads:

```
>> Map[f, a + b + c, Heads->True]
f[Plus][f[a], f[b], f[c]]
```

MapAt

<pre>MapAt[f, expr, n]</pre>	applies f to the element at position n in $expr$. If n is negative, the position is counted from the end.
<pre>MapAt[f, expr, {i, j ...}]</pre>	applies f to the part of $expr$ at position $\{i, j, \dots\}$.
<pre>MapAt[f, pos]</pre>	represents an operator form of $MapAt$ that can be applied to an expression.

Map f onto the part at position 2:

```
>> MapAt[f, {a, b, c, d}, 2]
{a, f[b], c, d}
```

Map f onto multiple parts:

```
>> MapAt[f, {a, b, c, d}, {{1}, {4}}]
{f[a], b, c, f[d]}
```

Map f onto the at the end:

```
>> MapAt[f, {a, b, c, d}, -1]
{a, b, c, f[d]}
```

Map f onto an association:

```
>> MapAt[f, <|"a" -> 1, "b" -> 2, "c" -> 3, "d" -> 4, "e" -> 5|>, 3]
{a -> 1, b -> 2, c -> f[3], d -> 4, e -> 5}
```

Use negative position in an association:

```
>> MapAt[f, <|"a" -> 1, "b" -> 2, "c" -> 3, "d" -> 4|>, -3]
{a -> 1, b -> f[2], c -> 3, d -> 4}
```

Use the operator form of $MapAt$:

```
>> MapAt[f, 1][{a, b, c, d}]
{f[a], b, c, d}
```

MapIndexed

<pre>MapIndexed[f, expr]</pre>	applies f to each part on the first level of $expr$, including the part positions in the call to f .
<pre>MapIndexed[f, expr, levelspec]</pre>	applies f to each level specified by $levelspec$ of $expr$.

```
>> MapIndexed[f, {a, b, c}]
{f[a, {1}], f[b, {2}], f[c, {3}]}
```

Include heads (index 0):

```
>> MapIndexed[f, {a, b, c}, Heads->True]
f [List, {0}] [f [a, {1}], f [b, {2}], f [c, {3}]]
```

Map on levels 0 through 1 (outer expression gets index {}):

```
>> MapIndexed[f, a + b + c * d, {0, 1}]
f [f [a, {1}] + f [b, {2}] + f [cd, {3}], {}]
```

Get the positions of atoms in an expression (convert operations to List first to disable Listable functions):

```
>> expr = a + b * f[g] * c ^ e;
>> listified = Apply[List, expr, {0, Infinity}];
>> MapIndexed[#2 &, listified, {-1}]
{{1}, {{2, 1}}, {{2, 2, 1}}, {{2, 3, 1}, {2, 3, 2}}}}
```

Replace the heads with their positions, too:

```
>> MapIndexed[#2 &, listified, {-1}, Heads -> True]
{0} [{1}, {2, 0} [{2, 1}, {2, 2, 0} [{2, 2, 1}], {2, 3, 0} [{2, 3, 1}, {2, 3, 2}]]]
```

The positions are given in the same format as used by Extract. Thus, mapping Extract on the indices given by MapIndexed re-constructs the original expression:

```
>> MapIndexed[Extract[expr, #2] &, listified, {-1}, Heads -> True]
a + bf [g] c^e
```

MapThread

```
'MapThread[f, {{a1, a2, ...}, {b1, b2, ...}, ...}]
  returns {f[a1, b1, ...], f[a2, b2, ...], ...}.
MapThread[f, {expr1, expr2, ...}, n]
  applies f at level n.
```

```
>> MapThread[f, {{a, b, c}, {1, 2, 3}}]
{f [a, 1], f [b, 2], f [c, 3]}

>> MapThread[f, {{{a, b}, {c, d}}, {{e, f}, {g, h}}}, 2]
{{f [a, e], f [b, f]}, {f [c, g], f [d, h]}}
```

Scan

```
Scan[f, expr]
  applies f to each element of expr and returns Null.
'Scan[f, expr, levelspec]
  applies f to each level specified by levelspec of expr.
```

```
>> Scan[Print, {1, 2, 3}]
1
2
3
```

Thread

```
Thread[f[args]]  
  threads f over any lists that appear in args.  
Thread[f[args], h]  
  threads over any parts with head h.
```

```
>> Thread[f[{a, b, c}]]  
{f[a], f[b], f[c]}  
  
>> Thread[f[{a, b, c}, t]]  
{f[a, t], f[b, t], f[c, t]}  
  
>> Thread[f[a + b + c], Plus]  
f[a] + f[b] + f[c]
```

Functions with attribute `Listable` are automatically threaded over lists:

```
>> {a, b, c} + {d, e, f} + g  
{a + d + g, b + e + g, c + f + g}
```

Function Application

Function Application

Function (&)

```
Function[body]  
body &  
  represents a pure function with parameters #1, #2, etc.  
Function[{x1, x2, ...}, body]  
  represents a pure function with parameters x1, x2, etc.  
Function[{x1, x2, ...}, body, attr]  
  assume that the function has the attributes attr.
```

```
>> f := # ^ 2 &  
  
>> f[3]  
9  
  
>> #^3& /@ {1, 2, 3}  
{1, 8, 27}  
  
>> #1+#2&[4, 5]  
9
```

You can use `Function` with named parameters:

```
>> Function[{x, y}, x * y][2, 3]  
6
```

Parameters are renamed, when necessary, to avoid confusion:

```

>> Function[{x}, Function[{y}, f[x, y]]][y]
Function [{y$},f [y,y$]]

>> Function[{y}, f[x, y]] /. x->y
Function [{y},f [y,y]]

>> Function[y, Function[x, y^x]][x][y]
xy

>> Function[x, Function[y, x^y]][x][y]
xy

```

Slots in inner functions are not affected by outer function application:

```

>> g[#] & [h[#]] & [5]
g[h[5]]

```

In the evaluation process, the attributes associated with an Expression are determined by its Head. If the Head is also a non-atomic Expression, in general, no Attribute is assumed. In particular, it is what happens when the head of the expression has the form:

“Function[body]” or: “Function[vars, body]”

```

>> h := Function[{x}, Hold[1+x]]

>> h[1 + 1]
Hold[1 + 2]

```

Notice that *Hold* in the body prevents the evaluation of \$1+x\$, but not the evaluation of \$1+1\$. To avoid that evaluation, of its arguments, the Head should have the attribute *HoldAll*. This behavior can be obtained by using the three arguments form version of this expression:

```

>> h:= Function[{x}, Hold[1+x], HoldAll]

>> h[1+1]
Hold[1 + (1 + 1)]

```

In this case, the attribute *HoldAll* is assumed, preventing the evaluation of the argument \$1+1\$ before passing it to the function body.

Slot

# <i>n</i>	represents the <i>n</i> th argument to a pure function.
#	is short-hand for #1.
#0	represents the pure function itself.

```

>> #
#1

```

Unused arguments are simply ignored:

```

>> {[#1, #2, #3]&[1, 2, 3, 4, 5]
{1,2,3}

```

Recursive pure functions can be written using #0:

```
>> If[#1<=1, 1, #1 #0[#1-1]]& [10]
3628800
```

SlotSequence

```
##  
    is the sequence of arguments supplied to a pure function.  
##n  
    starts with the nth argument.
```

```
>> Plus[##]& [1, 2, 3]
6
>> Plus[##2]& [1, 2, 3]
5
>> FullForm[##]
SlotSequence[1]
```

Functional Composition and Operator Forms

Functional Composition and Operator Forms

Functional Composition is a way to combine simple functions to build more complicated ones. Like the usual composition of functions in mathematics, the result of each function is passed as the argument of the next, and the result of the last one is the result of the whole.

The symbolic structure of Mathics3 makes it easy to create “operators” that can be composed and manipulated symbolically-forming “pipelines” of operations-and then applied to arguments.

Some built-in functions also directly support a “curried” form, in which they can immediately be given as symbolic operators.

Composition

WMA link

```
Composition[f, g]  
    returns the composition of two functions f and g.
```

```
>> Composition[f, g][x]
```

```
f[g[x]]
```

```
>> Composition[f, g, h][x, y, z]
```

```
f[g[h[x,y,z]]]
```

```
>> Composition[]
```

```
Identity
```

```
>> Composition[] [x]
```

```
x
```

```
>> Attributes[Composition]
```

```
{Flat, OneIdentity, Protected}
```

```
>> Composition[f, Composition[g, h]]  
Composition [f,g,h]
```

Identity

WMA link

```
Identity[x]  
is the identity function, which returns  $x$  unchanged.
```

```
>> Identity[x]  
x  
>> Identity[x, y]  
Identity [x,y]
```

Iteratively Applying Functions

Iteratively Applying Functions

Functional iteration is an elegant way to represent repeated operations that is used a lot.

FixedPoint

```
FixedPoint[f, expr]  
starting with  $expr$ , iteratively applies  $f$  until the result no longer changes.  
FixedPoint[f, expr, n]  
performs at most  $n$  iterations. The same that using $MaxIterations->n$
```

```
>> FixedPoint[Cos, 1.0]  
0.739085  
>> FixedPoint[#+1 &, 1, 20]  
21
```

FixedPointList

```
FixedPointList[f, expr]  
starting with  $expr$ , iteratively applies  $f$  until the result no longer changes, and returns a list of  
all intermediate results.  
FixedPointList[f, expr, n]  
performs at most  $n$  iterations.
```

```
>> FixedPointList[Cos, 1.0, 4]  
{1., 0.540302, 0.857553, 0.65429, 0.79348}
```

Observe the convergence of Newton's method for approximating square roots:

```
>> newton[n_] := FixedPointList[.5(# + n/#)&, 1.];
```

```

>> newton[9]
{1., 5., 3.4, 3.02353, 3.00009, 3., 3., 3.}

Plot the "hailstone" sequence of a number:
>> collatz[1] := 1;

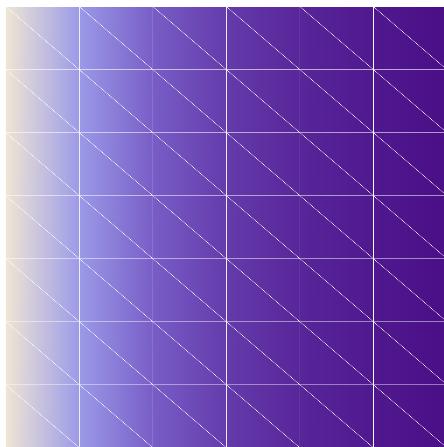
>> collatz[x_ ? EvenQ] := x / 2;

>> collatz[x_] := 3 x + 1;

>> list = FixedPointList[collatz, 14]
{14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 1}

>> ListLinePlot[list]

```



Fold

Fold[*f*, *x*, *list*]
 returns the result of iteratively applying the binary operator *f* to each element of *list*, starting with *x*.
Fold[*f*, *list*]
 is equivalent to **Fold**[*f*, **First**[*list*], **Rest**[*list*]].

```

>> Fold[Plus, 5, {1, 1, 1}]
8
>> Fold[f, 5, {1, 2, 3}]
f [f [f [5,1],2],3]

```

FoldList

FoldList[*f*, *x*, *list*]
 returns a list starting with *x*, where each element is the result of applying the binary operator *f* to the previous result and the next element of *list*.
FoldList[*f*, *list*]
 is equivalent to **FoldList**[*f*, **First**[*list*], **Rest**[*list*]].

```

>> FoldList[f, x, {1, 2, 3}]
{x, f[x, 1], f[f[x, 1], 2], f[f[f[x, 1], 2], 3]}

>> FoldList[Times, {1, 2, 3}]
{1, 2, 6}

```

Nest

Nest[*f*, *expr*, *n*]
starting with *expr*, iteratively applies *f* *n* times and returns the final result.

```

>> Nest[f, x, 3]
f[f[f[x]]]

>> Nest[(1+#+2 &, x, 2]
(1 + (1 + x)2)2

```

NestList

NestList[*f*, *expr*, *n*]
starting with *expr*, iteratively applies *f* *n* times and returns a list of all intermediate results.

```

>> NestList[f, x, 3]
{x, f[x], f[f[x]], f[f[f[x]]]}

>> NestList[2 # &, 1, 8]
{1, 2, 4, 8, 16, 32, 64, 128, 256}

```

Chaos game rendition of the Sierpinski triangle:

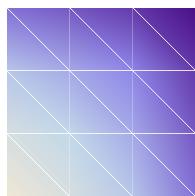
```

>> vertices = {{0,0}, {1,0}, {.5, .5 Sqrt[3]}};

>> points = NestList[.5(vertices[[ RandomInteger[{1,3}] ]] + #)&,
{0.,0.}, 500];

>> Graphics[Point[points], ImageSize->Small]

```



NestWhile

```
NestWhile[f, expr, test]
  applies a function f repeatedly on an expression expr, until applying test on the result no
  longer yields True.
NestWhile[f, expr, test, m]
  supplies the last m results to test (default value: 1).
NestWhile[f, expr, test, All]
  supplies all results gained so far to test.
```

Divide by 2 until the result is no longer an integer:

```
>> NestWhile[#/2&, 10000, IntegerQ]
 625
  —
  2
```

Calculate the sum of third powers of the digits of a number until the same result appears twice:

```
>> NestWhile[Total[IntegerDigits[#]^3] &, 5, UnsameQ, All]
 371
```

Print the intermediate results:

```
>> NestWhile[Total[IntegerDigits[#]^3] &, 5, (Print[{##}]; UnsameQ[##])
  &, All]
{5}
{5, 125}
{5, 125, 134}
{5, 125, 134, 92}
{5, 125, 134, 92, 737}
{5, 125, 134, 92, 737, 713}
{5, 125, 134, 92, 737, 713, 371}
{5, 125, 134, 92, 737, 713, 371, 371}
371
```

22. Functions used in Quantum Mechanics

Contents

Angular Momentum	236	PauliMatrix	237	ThreeJSymbol	238
ClebschGordan	236	SixJSymbol	237		

Angular Momentum

Angular Momentum

Angular momentum in physics is the rotational analog of linear momentum. It is an important quantity in physics because it is a conserved quantity the total angular momentum of a closed system remains constant.

ClebschGordan

Clebsch-Gordan coefficients matrices (SymPy, WMA)

`ClebschGordan[{j1, m1}, {j2, m2}, {j m}]`
returns the Clebsch-Gordan coefficient for the decomposition of $|j, m\rangle$ in terms of $|j_1, m_1\rangle, |j_2, m_2\rangle$.

```
>> ClebschGordan[{3 / 2, 3 / 2}, {1 / 2, -1 / 2}, {1, 1}]  

$$\frac{\sqrt{3}}{2}$$

```

ClebschGordan works with integer and halfinteger arguments:

```
>> ClebschGordan[{1/2, -1/2}, {1/2, -1/2}, {1, -1}]  
1  
>> ClebschGordan[{1/2, -1/2}, {1, 0}, {1/2, -1/2}]  

$$-\frac{\sqrt{3}}{3}$$

```

Compare with WMA example:

```
>> ClebschGordan[{5, 0}, {4, 0}, {1, 0}] == Sqrt[5 / 33]  
True
```

PauliMatrix

Pauli matrices (SymPy, WMA)

`PauliMatrix[k]`
returns the k th Pauli spin matrix).

```

>> Table[PauliMatrix[i], {i, 1, 3}]
{{{0, 1}, {1, 0}}, {{0, -I}, {I, 0}}, {{1, 0}, {0, -1}}}

>> PauliMatrix[1] . PauliMatrix[2] == I PauliMatrix[3]
True

>> MatrixExp[I \[Phi]/2 PauliMatrix[3]]

$$\left\{ \left\{ E^{\frac{I}{2} \text{phi}}, 0 \right\}, \left\{ 0, E^{\left( -\frac{I}{2} \right) \text{phi}} \right\} \right\}$$


>> % /. \[Phi] \[Rule] 2 Pi
{{-1, 0}, {0, -1}}

```

SixJSymbol

6-j symbol (SymPy, WMA)

```

SixJSymbol[{j1, j2, j3}, {j4, j5, j6}]
returns the values of the Wigner 6-j symbol.

```

```

>> SixJSymbol[{1, 2, 3}, {1, 2, 3}]

$$\frac{1}{105}$$


```

SixJSymbol is symmetric under permutations:

```

>> % == SixJSymbol[{3, 2, 1}, {3, 2, 1}]
True

>> SixJSymbol[{1, 2, 3}, {1, 2, 3}] == SixJSymbol[{2, 1, 3}, {2, 1, 3}]
True

```

SixJSymbol works with integer and half-integer arguments:

```

>> SixJSymbol[{1/2, 1/2, 1}, {5/2, 7/2, 3}]

$$-\frac{\sqrt{21}}{21}$$


```

Compare with WMA example:

```

>> SixJSymbol[{1, 2, 3}, {2, 1, 2}] == 1 / (5 Sqrt[21])
True

```

Result 0 returned for unphysical cases:

```

>> SixJSymbol[{1, 2, 3}, {4, 5, 12}]
0

```

Arguments must be integer or half integer values:

```

>> SixJSymbol[{0.5, 0.5, 1.1}, {0.5, 0.5, 1.1}]
SixJSymbol values {0.5, 0.5, 1.1} {0.5, 0.5, 1.1} must be integer or
half integer and fulfill the triangle relation
SixJSymbol [{0.5, 0.5, 1.1}, {0.5, 0.5, 1.1}]

```

ThreeJSymbol

3-j symbol (SymPy, WMA)

```
ThreeJSymbol[{j1, m1}, {j2, m2}, {j3, m3}]  
    returns the values of the Wigner 3-j symbol.
```

Compare with SymPy examples:

```
>> ThreeJSymbol[{2, 0}, {6, 0}, {4, 0}]  

$$\frac{\sqrt{715}}{143}$$

```

ThreeJSymbol is symmetric under permutations:

```
>> % == ThreeJSymbol[{2, 0}, {4, 0}, {6, 0}] == ThreeJSymbol[{4, 0}, {2, 0}, {6, 0}]  
True  
>> ThreeJSymbol[{2, 0}, {6, 0}, {4, 1}]  
0
```

Compare with WMA examples:

```
>> ThreeJSymbol[{6, 0}, {4, 0}, {2, 0}] == Sqrt[5 / 143]  
True  
>> ThreeJSymbol[{2, 1}, {2, 2}, {4, -3}] == -(1 / (3 Sqrt[2]))  
True  
>> ThreeJSymbol[{1/2, -1/2}, {1/2, -1/2}, {1, 1}]  

$$-\frac{\sqrt{3}}{3}$$

```

Result 0 returned for unphysical cases:

```
>> ThreeJSymbol[{1, 2}, {3, 4}, {5, 12}]  
0
```

Arguments must be integer or half integer values:

```
>> ThreeJSymbol[{2.1, 6}, {4, 0}, {0, 0}]  
ThreeJSymbol values {2.1, 6}, {4, 0}, {0, 0} must be integer or half  
integer  
ThreeJSymbol [{2.1, 6}, {4, 0}, {0, 0}]
```

23. Global System Information

Contents

\$CommandLine	239	\$PythonImplementation	241	Environment	242
\$Machine	239	\$ScriptCommandLine	241	GetEnvironment	243
\$MachineName	239	\$SystemID	241	MathicsVersion	243
\$Packages	240	\$SystemMemory	241	MemoryAvailable	243
\$ParentProcessID	240	\$SystemWordLength	241	MemoryInUse	243
\$ProcessID	240	\$UserName	242	Run	243
\$ProcessorType	240	\$Version	242	Share	244
		\$VersionNumber	242		

\$CommandLine

WMA link

`$CommandLine`

is a list of strings passed on the command line to launch the Mathics session.

>> `$CommandLine`

{docpipeline.py,–output,–keep-going,–want-sorting,–load-module,pymathics.graph,pymathics.natlang}

\$Machine

WMA link

`$Machine`

returns a string describing the type of computer system on which the Mathics is being run.

>> `$Machine`

linux

\$MachineName

WMA link

`$MachineName`

is a string that gives the assigned name of the computer on which Mathics is being run, if such a name is defined.

>> `$MachineName`

muffin

\$Packages

WMA link

\$Packages

returns a list of the contexts corresponding to all packages which have been loaded into Mathics.

>> **\$Packages**

```
{ImportExport', XML', Internal', System', Global'}
```

\$ParentProcessID

WMA link

\$ParentProcessID

gives the ID assigned to the process which invokes the *Mathics3* by the operating system under which it is run.

>> **\$ParentProcessID**

```
83042
```

\$ProcessID

WMA link

\$ProcessID

gives the ID assigned to the *Mathics3* process by the operating system under which it is run.

>> **\$ProcessID**

```
83043
```

\$ProcessorType

WMA link

\$ProcessorType

gives a string giving the architecture of the processor on which the *Mathics3* is being run.

>> **\$ProcessorType**

```
x86_64
```

\$PythonImplementation

\$PythonImplementation

gives a string indicating the Python implementation used to run *Mathics3*.

```
>> $PythonImplementation  
CPython 3.10.10.final.0
```

\$ScriptCommandLine

WMA link

```
$ScriptCommandLine  
is a list of string arguments when running the kernel is script mode.
```

```
>> $ScriptCommandLine  
{}
```

\$SystemID

WMA link

```
$SystemID  
is a short string that identifies the type of computer system on which the Mathics3 is being run.
```

```
>> $SystemID  
linux
```

\$SystemMemory

WMA link

```
$SystemMemory  
Returns the total amount of physical memory.
```

```
>> $SystemMemory  
33611120640
```

\$SystemWordLength

WMA link

```
$SystemWordLength  
gives the effective number of bits in raw machine words on the computer system where Mathics3 is running.
```

```
>> $SystemWordLength  
64
```

\$UserName

WMA link

\$UserName

returns the login name, according to the operative system, of the user that started the current *Mathics3* session.

```
>> $UserName  
rocky
```

\$Version

WMA link

\$Version

returns a string with the current Mathics version and the versions of relevant libraries.

```
>> $Version  
Mathics 6.0.0 on CPython 3.10.10 (main, Feb 22 2023, 17:18:27) [GCC 11.3.0]  
using SymPy 1.11.1, mpmath 1.2.1, numpy 1.24.0, cython Not installed
```

\$VersionNumber

WMA link

\$VersionNumber

is a real number which gives the current Wolfram Language version that *Mathics3* tries to be compatible with.

```
>> $VersionNumber  
10.
```

Environment

WMA link

Environment [*var*]

gives the value of an operating system environment variable.

```
>> Environment["HOME"]  
/home/rocky
```

GetEnvironment

WMA link

GetEnvironment ["*var*"]

gives the setting corresponding to the variable "*var*" in the operating system environment.

```
>> GetEnvironment["HOME"]
```

MathicsVersion

MathicsVersion

this string is the version of Mathics we are running.

```
>> MathicsVersion  
6.0.0
```

MemoryAvailable

WMA link

MemoryAvailable

Returns the amount of the available physical memory.

```
>> MemoryAvailable[]  
21087191040
```

The relationship between \$SystemMemory, MemoryAvailable, and MemoryInUse:

```
>> $SystemMemory > MemoryAvailable[] > MemoryInUse[]  
True
```

MemoryInUse

WMA link

MemoryInUse []

Returns the amount of memory used by all of the definitions objects if we can determine that;
-1 otherwise.

```
>> MemoryInUse[]  
48
```

Run

WMA link

Run [command]

runs command as an external operating system command, returning the exit code obtained.

```
>> Run["date"]  
0
```

Share

WMA link

Share []

release memory forcing Python to do garbage collection. If Python package is psutil installed is the amount of released memory is returned. Otherwise returns 0. This function differs from WMA which tries to reduce the amount of memory required to store definitions, by reducing duplicated definitions.

Share[Symbol]

Does the same thing as Share [] ; Note: this function differs from WMA which tries to reduce the amount of memory required to store definitions associated to *Symbol*.

```
>> Share[]
```

```
1904640
```

24. Graphics and Drawing

Showing something visually can be done in a number of ways:

- Starting with complete images and modifying them using the `Image` Built-in function.
- Use pre-defined 2D or 3D objects like `Circle` of section 12 and `Cuboid` of section ?? and place them in a coordinate space.
- Compute the points of the space using a function. This is done using functions like `Plot` of section ?? and `ListPlot` of section ??.

Contents

Plotting Data	245	NumberLinePlot	258	Cylinder	273
BarChart	247	ParametricPlot	259	Graphics	274
ColorData	248	PieChart	261	Graphics3D	275
ColorDataFunction	248	Plot	264	RGBColor	276
DensityPlot	249	Plot3D	266	Sphere	277
DiscretePlot	250	PolarPlot	268	Tube	278
Graphics	252	Splines	268	Uniform Polyhedra	278
Graphics3D	252	BernsteinBasis	268	Dodecahedron	278
Histogram	253	BezierCurve	269	Icosahedron	279
ListLinePlot	254	BezierFunction	270	Octahedron	279
ListLogPlot	255	Three-Dimensional		Tetrahedron	279
ListPlot	257	Graphics	270	UniformPolyhedron	280
LogPlot	257	Cone	271		
		Cuboid	272		

Plotting Data

Plotting Data

Plotting functions take a function as a parameter and data, often a range of points, as another parameter, and plot or show the function applied to the data.

BarChart

WMA link

`BarChart[{b1, b2 ...}]`
makes a bar chart with lengths b_1, b_2, \dots

Draw-

ing options include - Charting:

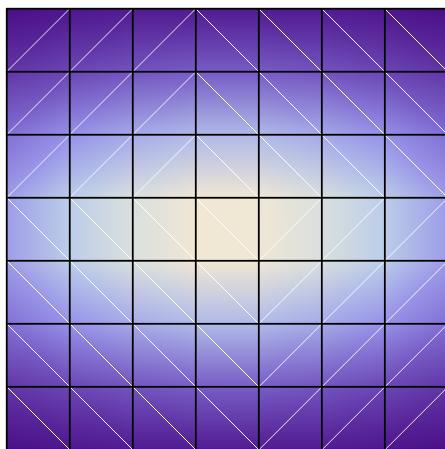
- Mesh
- PlotRange
- ChartLabels
- ChartLegends
- ChartStyle

BarChart specific:

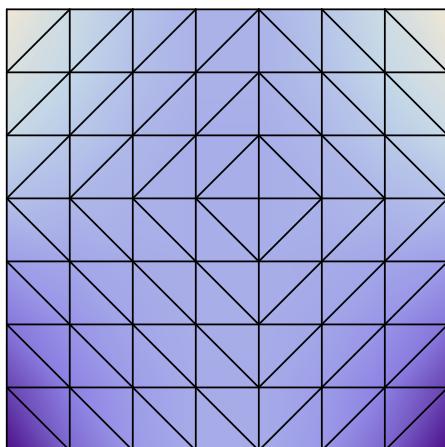
- Axes (default {False, True})
- AspectRatio: (default 1 / GoldenRatio)

A bar chart of a list of heights:

```
>> BarChart[{1, 4, 2}]
```



```
>> BarChart[{1, 4, 2}, ChartStyle -> {Red, Green, Blue}]
```



```
>> BarChart[{{1, 2, 3}, {2, 3, 4}}]
```

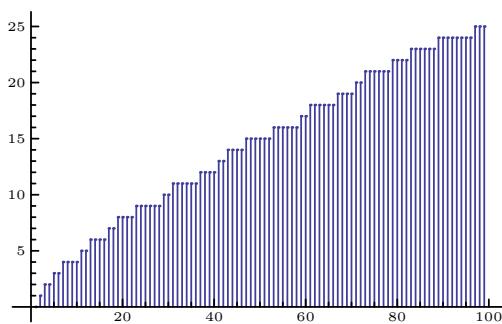
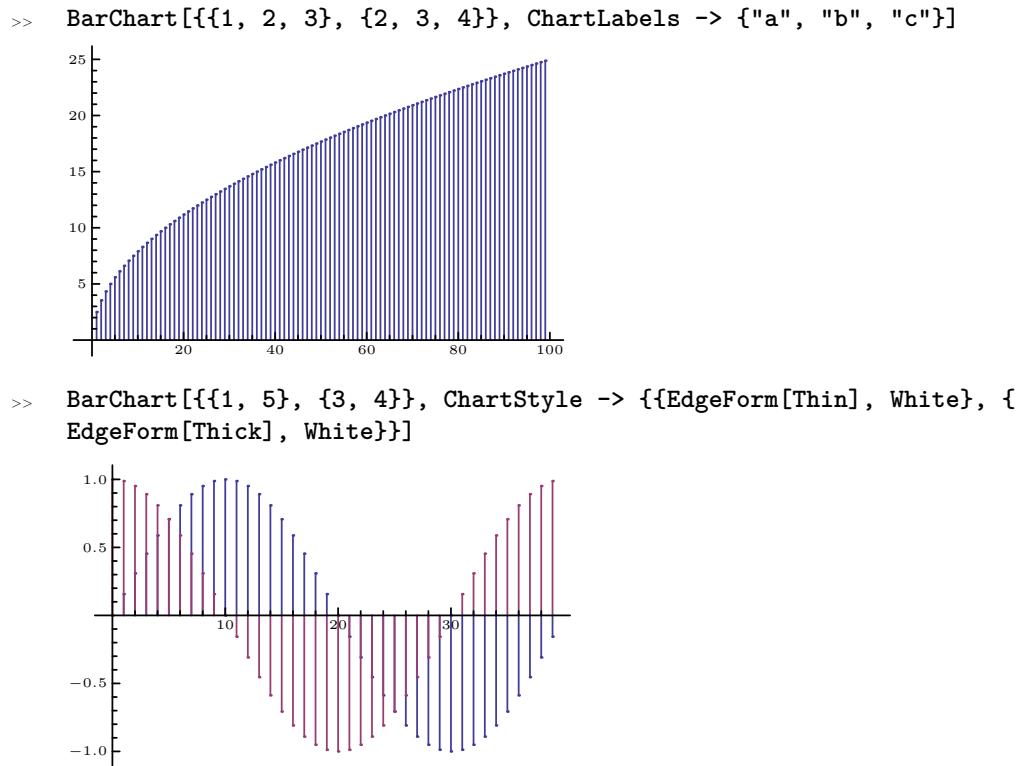


Chart several datasets with categorical labels:



ColorData

[WMA link](#)

```
ColorData["name"]
```

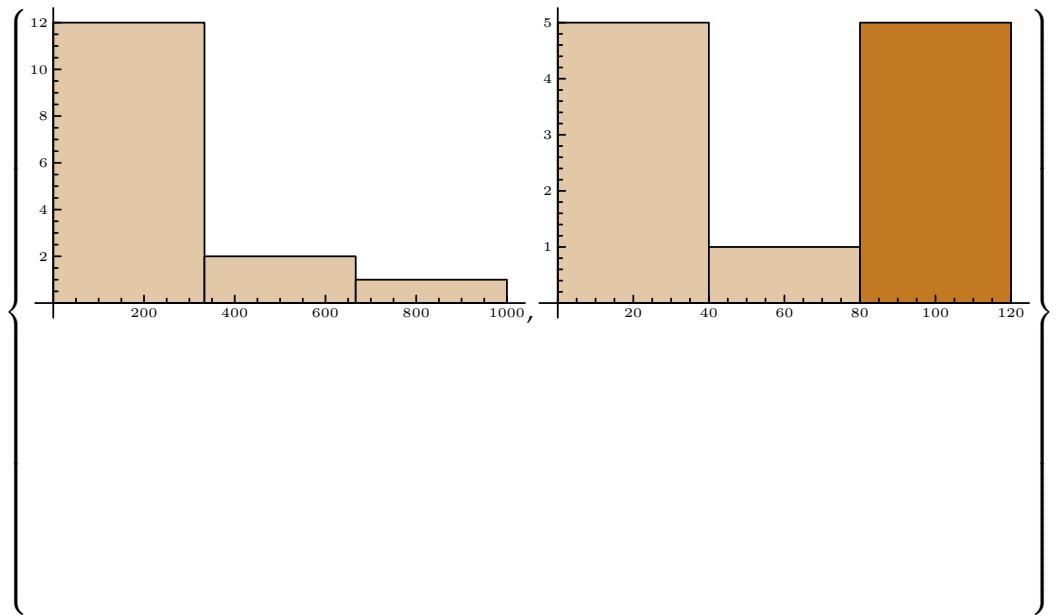
returns a color function with the given *name*.

Define a user-defined color function:

```
>> Unprotect[ColorData]; ColorData["test"] := ColorDataFunction["test",
  "Gradients", {0, 1}, Blend[{Red, Green, Blue}, #1] &]; Protect[
  ColorData]
```

Compare it to the default color function, *LakeColors*:

```
>> {DensityPlot[x + y, {x, -1, 1}, {y, -1, 1}], DensityPlot[x + y, {x, -1, 1}, {y, -1, 1}, ColorFunction->"test"]}
```



ColorDataFunction

WMA link

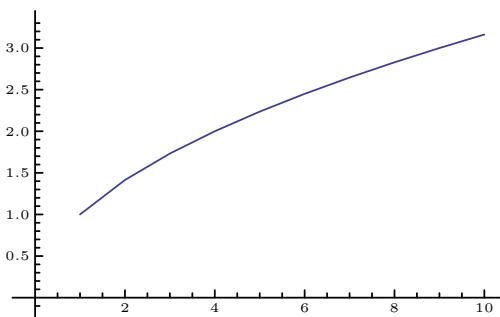
```
ColorDataFunction[range, ...]
is a function that represents a color scheme.
```

DensityPlot

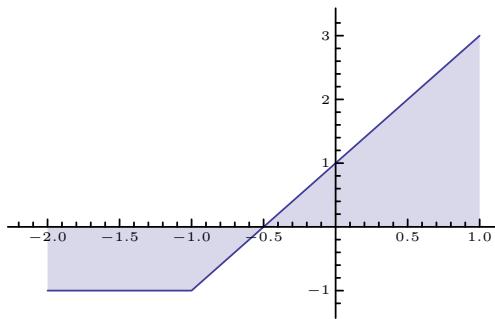
WMA link

```
DensityPlot[f, {x, xmin, xmax}, {y, ymin, ymax}]
plots a density plot of f with x ranging from xmin to xmax and y ranging from ymin to ymax.
```

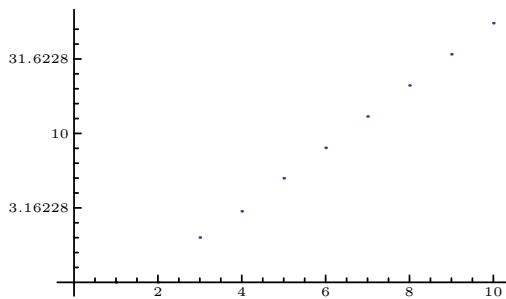
```
>> DensityPlot[x^2 + 1/y, {x, -1, 1}, {y, 1, 4}]
```



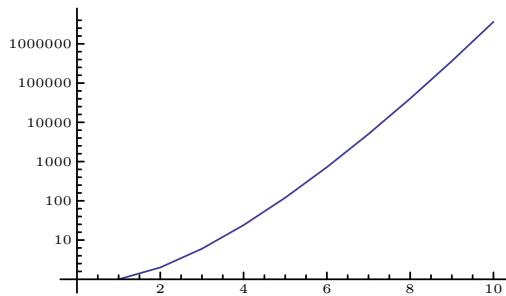
```
>> DensityPlot[1 / x, {x, 0, 1}, {y, 0, 1}]
```



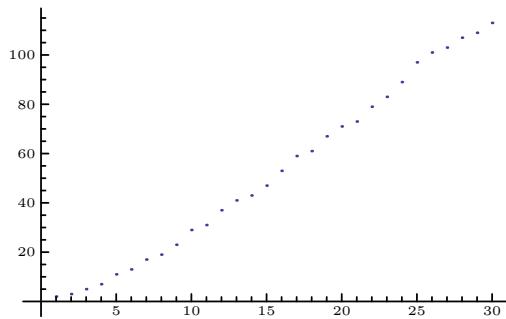
```
>> DensityPlot[Sqrt[x * y], {x, -1, 1}, {y, -1, 1}]
```



```
>> DensityPlot[1/(x^2 + y^2 + 1), {x, -1, 1}, {y, -2, 2}, Mesh->Full]
```



```
>> DensityPlot[x^2 y, {x, -1, 1}, {y, -1, 1}, Mesh->All]
```



DiscretePlot

WMA link

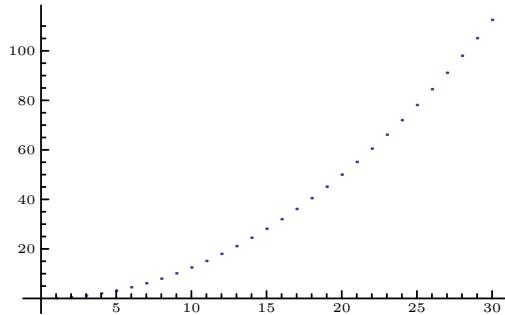
```

DiscretePlot[expr, {x, n_max}]
plots expr with x ranging from 1 to n_max.
DiscretePlot[expr, {x, n_min, n_max}]
plots expr with x ranging from n_min to n_max.
DiscretePlot[expr, {x, n_min, n_max, dn}]
plots expr with x ranging from n_min to n_max using steps dn.
DiscretePlot[{expr1, expr2, ...}, ...]
plots the values of all expri.

```

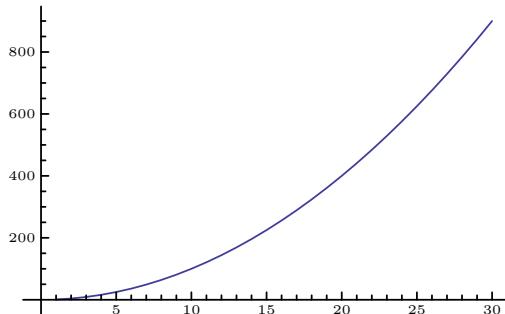
The number of primes for a number k :

```
>> DiscretePlot[PrimePi[k], {k, 1, 100}]
```



is about the same as $\text{Sqrt}[k] * 2.5$:

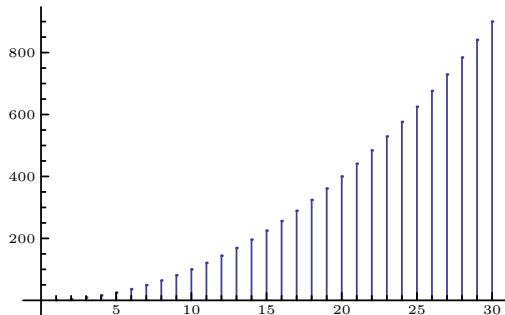
```
>> DiscretePlot[2.5 Sqrt[k], {k, 100}]
```



Notice in the above that when the starting value, n_{min} , is 1, we can omit it.

A plot can contain several functions, using the same parameter, here x :

```
>> DiscretePlot[{Sin[Pi x/20], Cos[Pi x/20]}, {x, 0, 40}]
```



Compare with Plot of section ??.

Graphics

WMA link

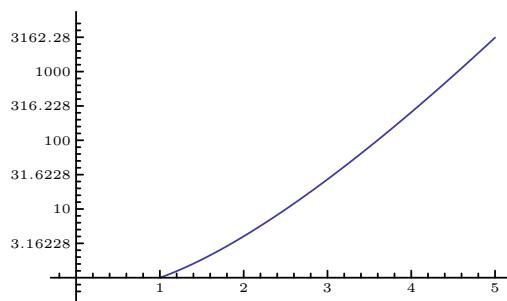
```
Graphics[primitives, options]
```

represents a graphic.

Options include:

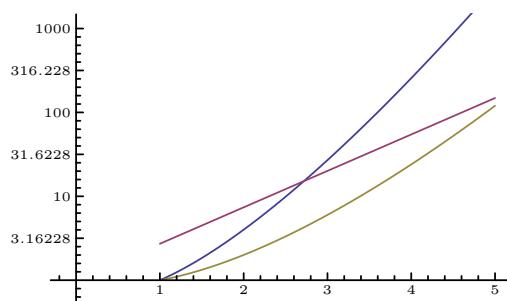
- Axes
- TicksStyle
- AxesStyle
- LabelStyle
- AspectRatio
- PlotRange
- PlotRangePadding
- ImageSize
- Background

```
>> Graphics[{Blue, Line[{{0,0}, {1,1}}]}]
```



Graphics supports PlotRange:

```
>> Graphics[{Rectangle[{1, 1}], Axes -> True, PlotRange -> {{-2, 1.5}, {-1, 1.5}}}]
```



```
>> Graphics[{Rectangle[], Red, Disk[{1,0}]}], PlotRange->{{0,1},{0,1}}]
```



Graphics produces GraphicsBox boxes:

```
>> Graphics[Rectangle[]] // ToBoxes // Head
```

GraphicsBox

In TeXForm, Graphics produces Asymptote figures:

```

>> Graphics[Circle[]] // TeXForm

\begin{asy}
usepackage("amsmath");
size(5.8556cm, 5.8333cm);
draw(ellipse((175,175),175,175), rgb(0, 0, 0)+linewidth(0.66667));
clip(box((-0.33333,0.33333), (350.33,349.67)));
\end{asy}

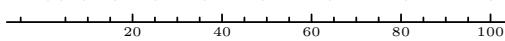
```

Graphics3D

WMA link

Graphics3D [*primitives*, *options*]
represents a three-dimensional graphic.
See also the Section “Plotting” for a list of Plot options.

```
>> Graphics3D[Polygon[{{0,0,0}, {0,1,1}, {1,0,0}}]]
```



In `TeXForm`, `Graphics3D` creates Asymptote figures:

```
>> Graphics3D[Sphere[]] // TeXForm
```

```

\begin{asy}
import three;
import solids;
size(6.6667cm, 6.6667cm);
currentprojection=perspective(2.6,-4.8,4.0);
currentlight=light(rgb(0.5,0.5,1), specular=red, (2,0,2), (2,2,2), (0,2,2));
// Sphere3DBox
draw(surface(sphere((0, 0, 0), 1)), rgb(1,1,1)+opacity(1));
draw((-1,-1,-1)-(1,-1,-1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,-1)-(1,1,-1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,1)-(1,-1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,1)-(1,1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,-1)-(1,1,-1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,-1)-(1,1,-1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,1)-(-1,1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,-1)-(1,1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,1)-(-1,-1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,-1)-(-1,1,1)), rgb(0.4, 0.4, 0.4)+linewidth(1));
\end{asy}

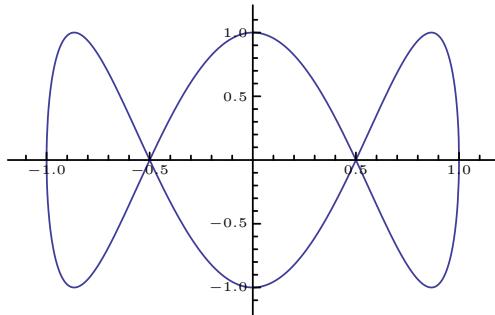
```

Histogram

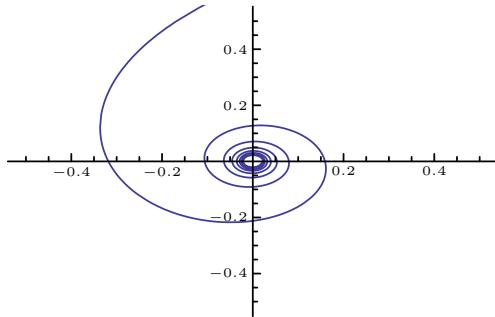
Histogram (WMA link)

```
Histogram[{x1, x2 ...}]  
plots a histogram using the values x1, x2, ....
```

```
>> Histogram[{3, 8, 10, 100, 1000, 500, 300, 200, 10, 20, 200, 100, 200,  
300, 500}]
```



```
>> Histogram[{{1, 2, 10, 5, 50, 20}, {90, 100, 101, 120, 80}}]
```

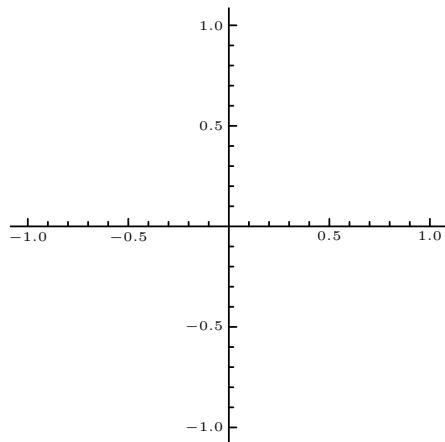


ListLinePlot

WMA link

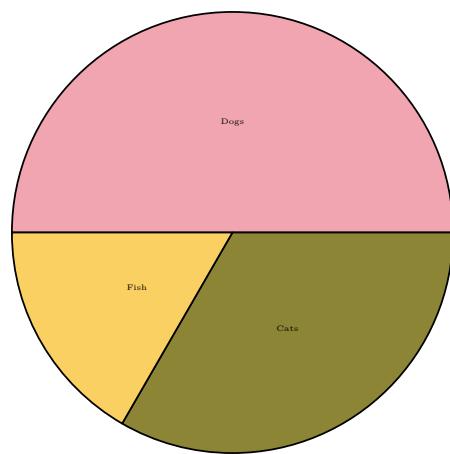
```
ListLinePlot[{y_1, y_2, ...}]  
plots a line through a list of y-values, assuming integer x-values 1, 2, 3, ...  
ListLinePlot[{{x_1, y_1}, {x_2, y_2}, ...}]  
plots a line through a list of x, y pairs.  
ListLinePlot[{list_1, list_2, ...}]  
plots several lines.
```

```
>> ListLinePlot[Table[{n, n ^ 0.5}, {n, 10}]]
```



ListPlot accepts a superset of the Graphics options.

```
>> ListLinePlot[{{{-2, -1}, {-1, -1}, {1, 3}}, Filling->Axis]
```



ListLogPlot

WMA link

```
ListLogPlot[{y_1, y_2, ...}]
```

log plots a list of y-values, assuming integer x-values 1, 2, 3, ...

```
ListLogPlot[{{x_1, y_1}, {x_2, y_2}, ...}]
```

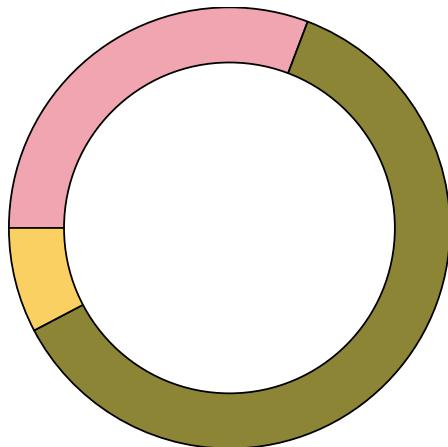
log plots a list of x, y pairs.

```
ListLogPlot[{list_1, list_2, ...}]
```

log plots several lists of points.

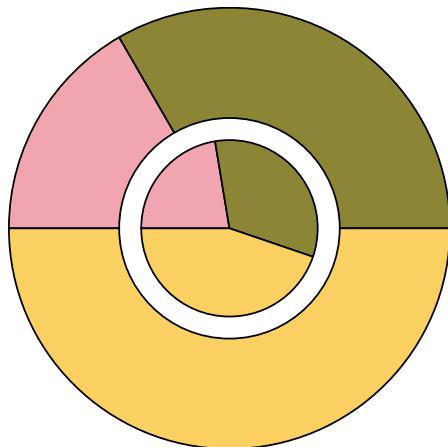
Plotting table of Fibonacci numbers:

```
>> ListLogPlot[Table[Fibonacci[n], {n, 10}]]
```



we see that Fibonacci numbers grow exponentially. So when plotted using on a log scale the result fits points of a sloped line.

```
>> ListLogPlot[Table[n!, {n, 10}], Joined -> True]
```



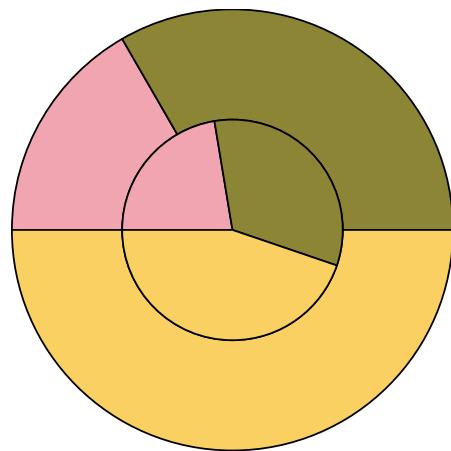
ListPlot

WMA link

```
ListPlot[{y_1, y_2, ...}]  
plots a list of y-values, assuming integer x-values 1, 2, 3, ...  
ListPlot[{{x_1, y_1}, {x_2, y_2}, ...}]  
plots a list of x, y pairs.  
ListPlot[{list_1, list_2, ...}]  
plots several lists of points.
```

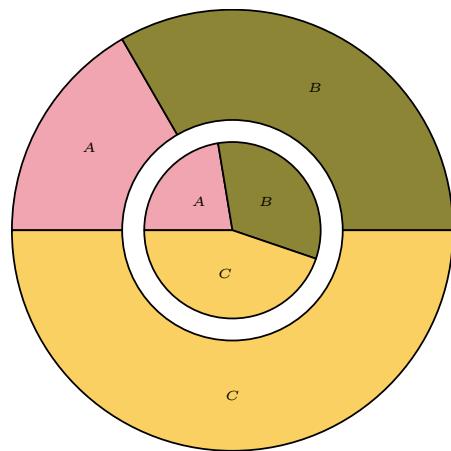
The frequency of Primes:

```
>> ListPlot[Prime[Range[30]]]
```



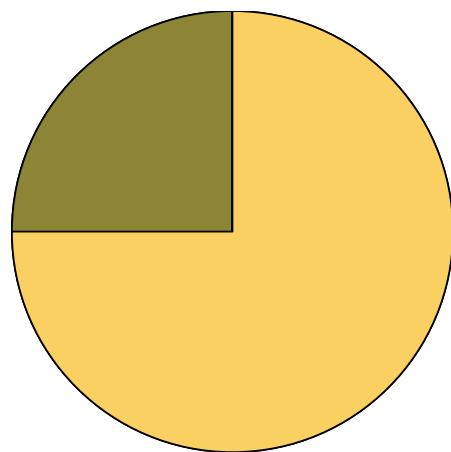
seems very roughly to fit a table of quadratic numbers:

```
>> ListPlot[Table[n^2/8, {n, 30}]]
```



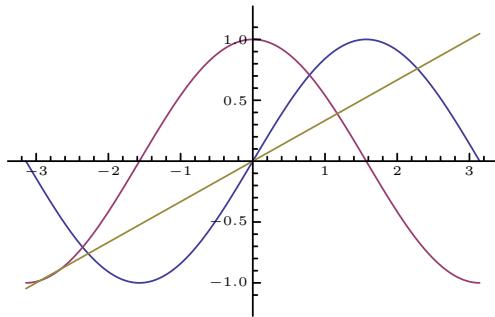
ListPlot accepts some Graphics options:

```
>> ListPlot[Table[n^2, {n, 30}], Joined->True]
```



Compare with Plot of section ??.

```
>> ListPlot[Table[n ^ 2, {n, 30}], Filling->Axis]
```



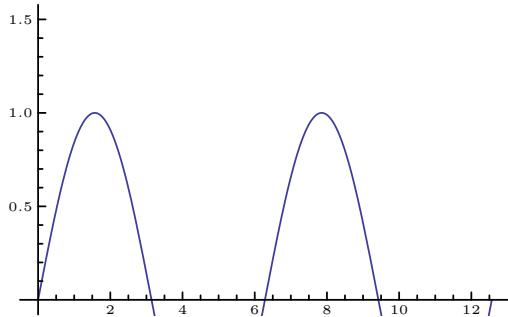
Compare with Plot of section ??.

LogPlot

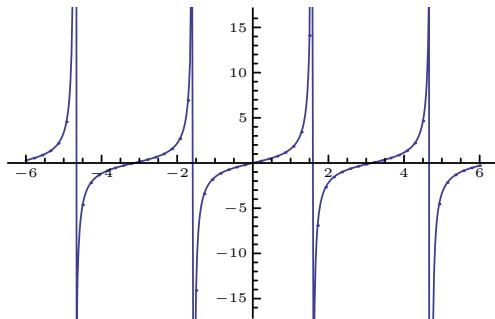
Semi-log plot (WMA link)

```
LogPlot[f, {x, xmin, xmax}]  
    log plots f with x ranging from xmin to xmax.  
Plot[{f1, f2, ...}, {x, xmin, xmax}]  
    log plots several functions f1, f2, ...
```

```
>> LogPlot[x^x, {x, 1, 5}]
```



```
>> LogPlot[{x^x, Exp[x], x!}, {x, 1, 5}]
```

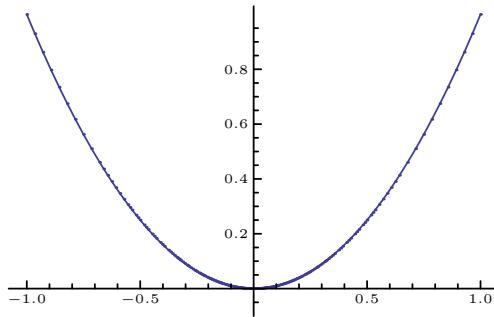


NumberLinePlot

WMA link

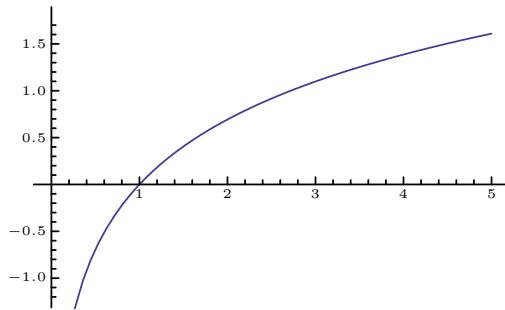
```
NumberLinePlot[{v_1, v_2, ...}]  
    plots a list of values along a line.
```

```
>> NumberLinePlot[Prime[Range[10]]]
```



Compare with:

```
>> NumberLinePlot[Table[x^2, {x, 10}]]
```

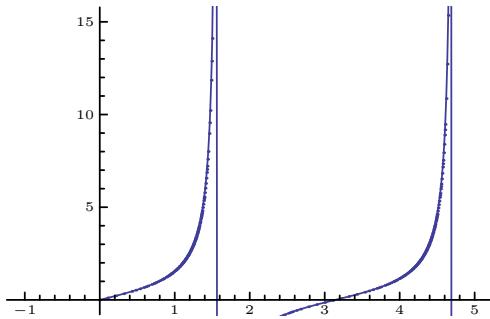


ParametricPlot

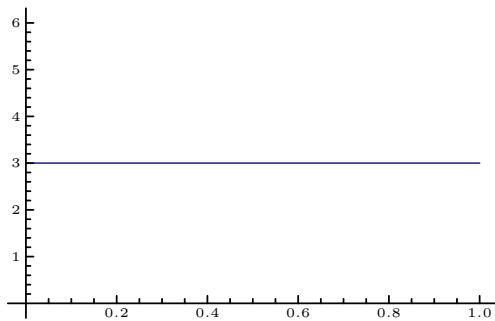
WMA link

```
ParametricPlot[{f_x, f_y}, {u, umin, umax}]
plots a parametric function  $f$  with the parameter  $u$  ranging from  $umin$  to  $umax$ .
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, ...}, {u, umin, umax}]
plots several parametric functions  $f, g, \dots$ 
ParametricPlot[{f_x, f_y}, {u, umin, umax}, {v, vmin, vmax}]
plots a parametric area.
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, ...}, {u, umin, umax}, {v, vmin, vmax}]
plots several parametric areas.
```

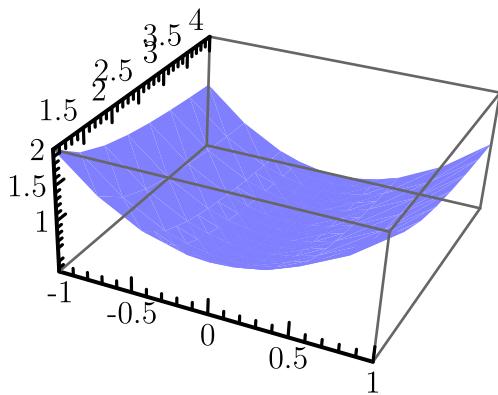
```
>> ParametricPlot[{Sin[u], Cos[3 u]}, {u, 0, 2 Pi}]
```



```
>> ParametricPlot[{Cos[u] / u, Sin[u] / u}, {u, 0, 50}, PlotRange->0.5]
```



```
>> ParametricPlot[{{Sin[u], Cos[u]}, {0.6 Sin[u], 0.6 Cos[u]}, {0.2 Sin[u], 0.2 Cos[u]}}, {u, 0, 2 Pi}, PlotRange->1, AspectRatio->1]
```



PieChart

Pie Chart (WMA link)

```
PieChart[{a1, a2 ...}]  
draws a pie chart with sector angles proportional to a1, a2, ....
```

Drawing options include - Charting:

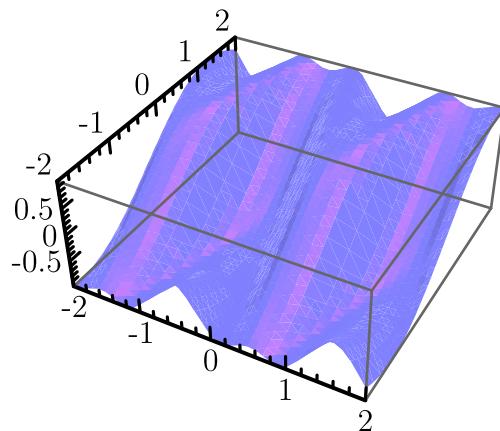
- Mesh
- PlotRange
- ChartLabels
- ChartLegends
- ChartStyle

PieChart specific:

- Axes (default: False, False)
- AspectRatio (default 1)
- SectorOrigin: (default {Automatic, 0})
- SectorSpacing" (default Automatic)

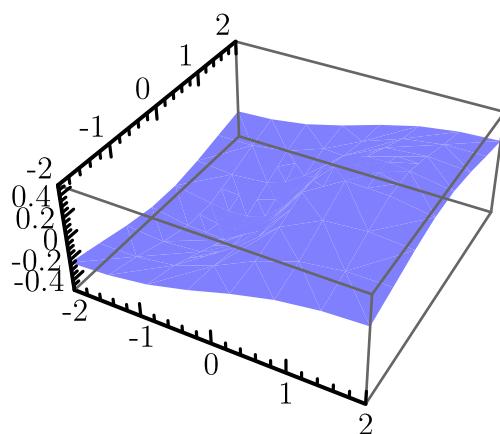
A hypothetical comparison between types of pets owned:

```
>> PieChart[{30, 20, 10}, ChartLabels -> {Dogs, Cats, Fish}]
```



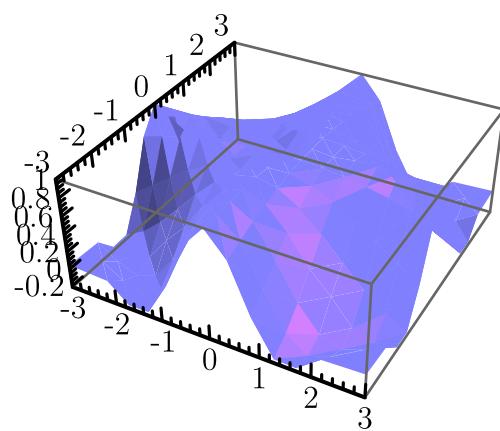
A doughnut chart for a list of values:

```
>> PieChart[{8, 16, 2}, SectorOrigin -> {Automatic, 1.5}]
```



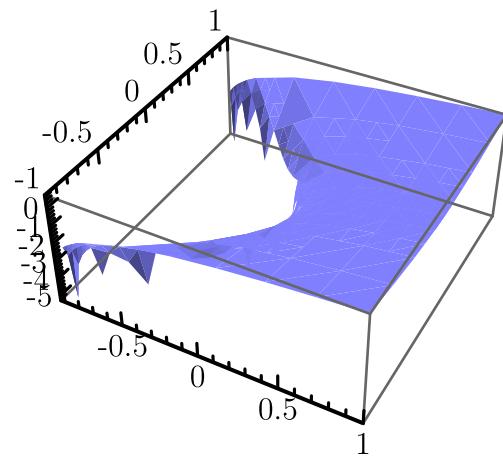
A Pie chart with multiple datasets:

```
>> PieChart[{{10, 20, 30}, {15, 22, 30}}]
```



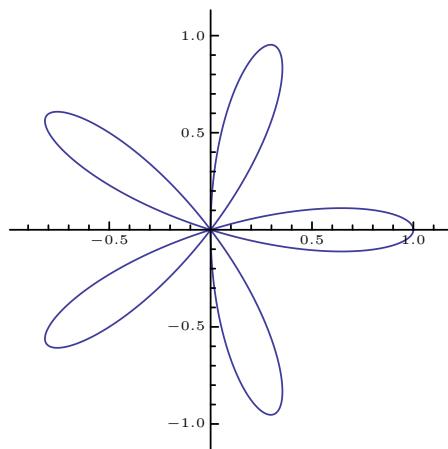
Same as the above, but without gaps between the groups of data:

```
>> PieChart[{{10, 20, 30}, {15, 22, 30}}, SectorSpacing -> None]
```



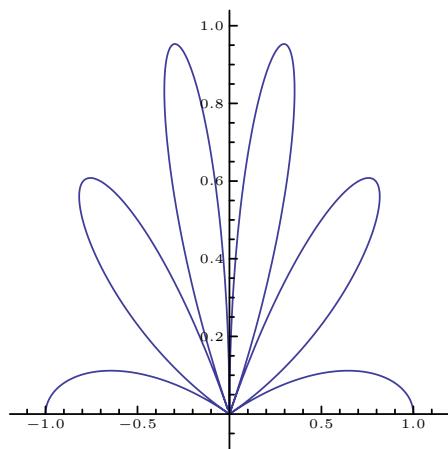
The doughnut chart above with labels on each of the 3 pieces:

```
>> PieChart[{{10, 20, 30}, {15, 22, 30}}, ChartLabels -> {A, B, C}]
```



Negative values are removed, the data below is the same as {1, 3}:

```
>> PieChart[{1, -1, 3}]
```



Plot

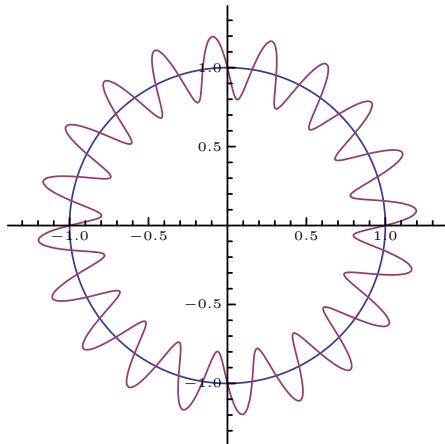
WMA link

```

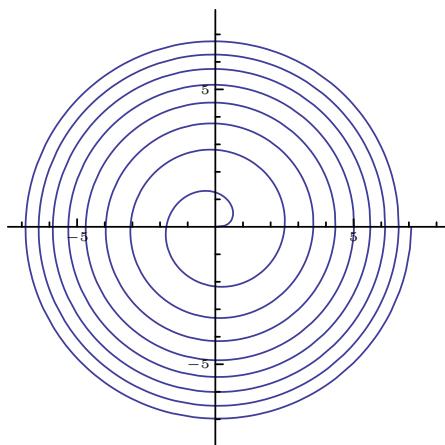
Plot[f, {x, xmin, xmax}]
plots f with x ranging from xmin to xmax.
Plot[{f1, f2, ...}, {x, xmin, xmax}]
plots several functions f1, f2, ...

```

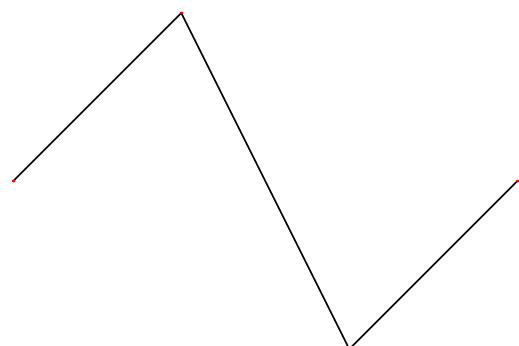
```
>> Plot[{Sin[x], Cos[x], x / 3}, {x, -Pi, Pi}]
```



```
>> Plot[Sin[x], {x, 0, 4 Pi}, PlotRange->{{0, 4 Pi}, {0, 1.5}}]
```



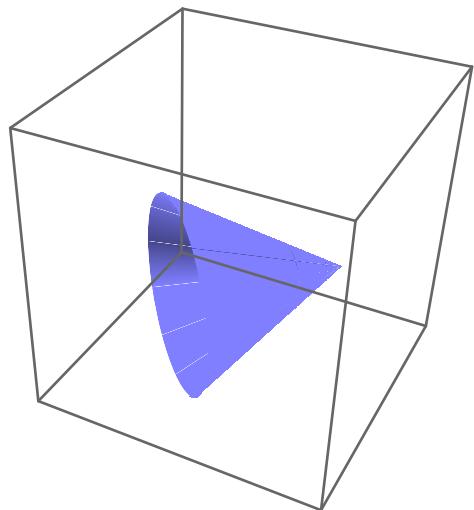
```
>> Plot[Tan[x], {x, -6, 6}, Mesh->Full]
```



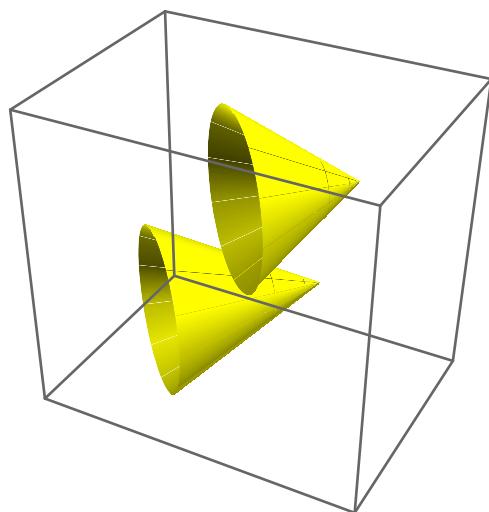
```
>> Plot[x^2, {x, -1, 1}, MaxRecursion->5, Mesh->All]
```



```
>> Plot[Log[x], {x, 0, 5}, MaxRecursion->0]
```

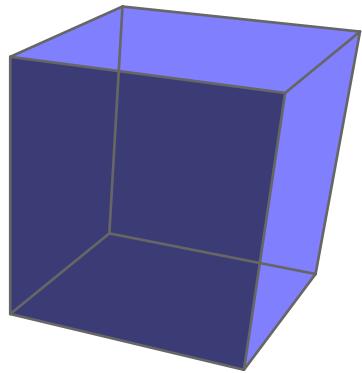


```
>> Plot[Tan[x], {x, 0, 6}, Mesh->All, PlotRange->{{-1, 5}, {0, 15}},  
MaxRecursion->10]
```



A constant function:

```
>> Plot[3, {x, 0, 1}]
```



Plot3D

WMA link

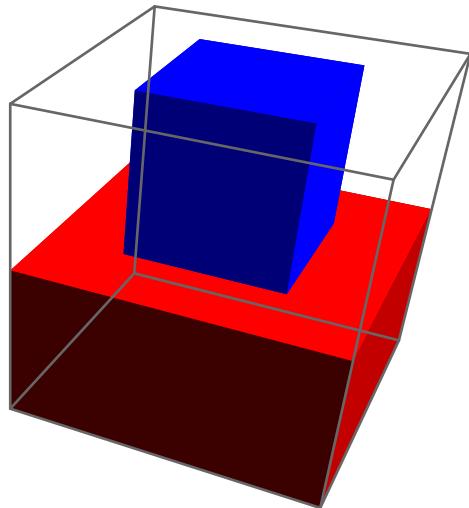
```
Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}]
```

creates a three-dimensional plot of f with x ranging from $xmin$ to $xmax$ and y ranging from $ymin$ to $ymax$.

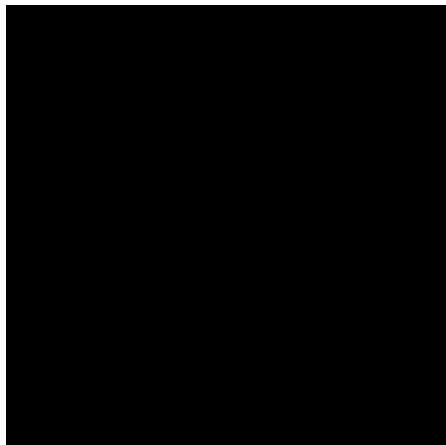
Plot3D has the same options as Graphics3D, in particular:

- Mesh
- PlotPoints
- MaxRecursion

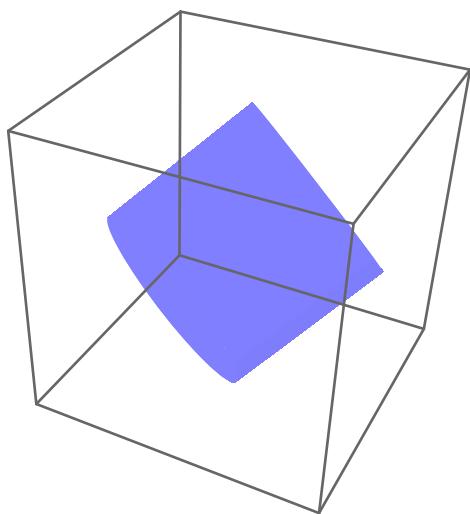
```
>> Plot3D[x^2 + 1/y, {x, -1, 1}, {y, 1, 4}]
```



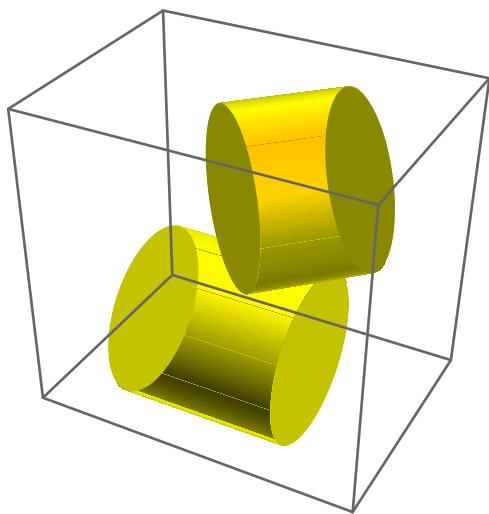
```
>> Plot3D[Sin[y + Sin[3 x]], {x, -2, 2}, {y, -2, 2}, PlotPoints->20]
```



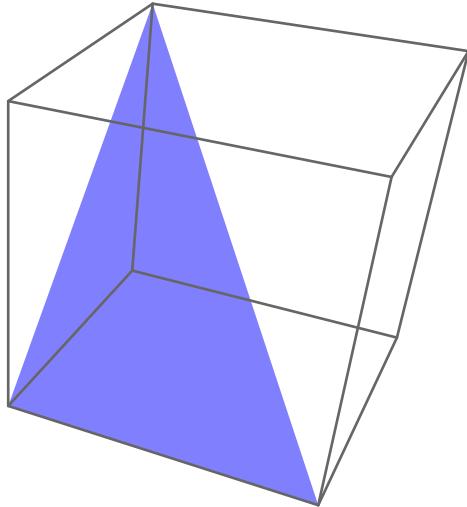
```
>> Plot3D[x / (x ^ 2 + y ^ 2 + 1), {x, -2, 2}, {y, -2, 2}, Mesh->None]
```



```
>> Plot3D[Sin[x y] /(x y), {x, -3, 3}, {y, -3, 3}, Mesh->All]
```



```
>> Plot3D[Log[x + y^2], {x, -1, 1}, {y, -1, 1}]
```



PolarPlot

WMA link

```
PolarPlot[r, {t, t_min, t_max}]
```

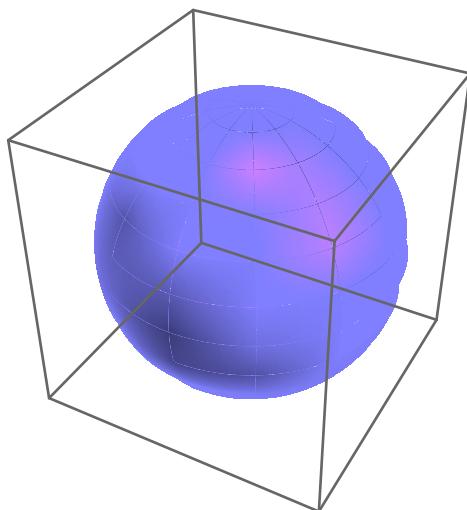
creates a polar plot of curve with radius r as a function of angle t ranging from t_{min} to t_{max} .

In a Polar Plot, a polar coordinate system is used.

A polar coordinate system is a two-dimensional coordinate system in which each point on a plane is determined by a distance from a reference point and an angle from a reference direction.

Here is a 5-blade propeller, or maybe a flower, using PolarPlot:

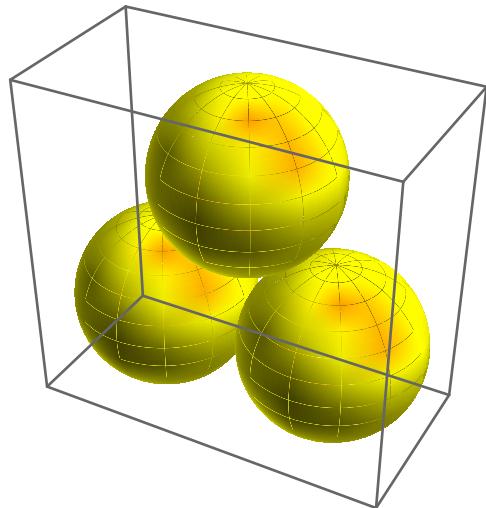
```
>> PolarPlot[Cos[5t], {t, 0, Pi}]
```



The number of blades can be changed by adjusting the t multiplier.

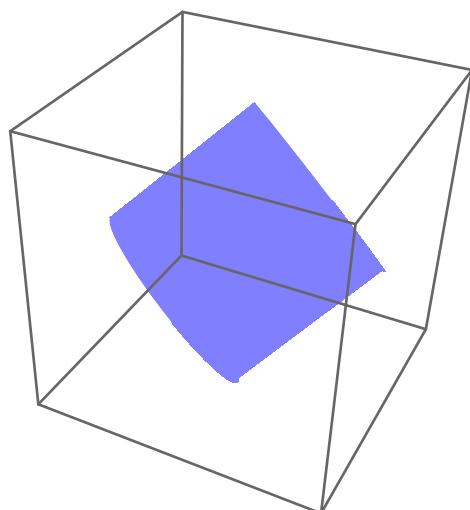
A slight change adding Abs turns this a clump of grass:

```
>> PolarPlot[Abs[Cos[5t]], {t, 0, Pi}]
```



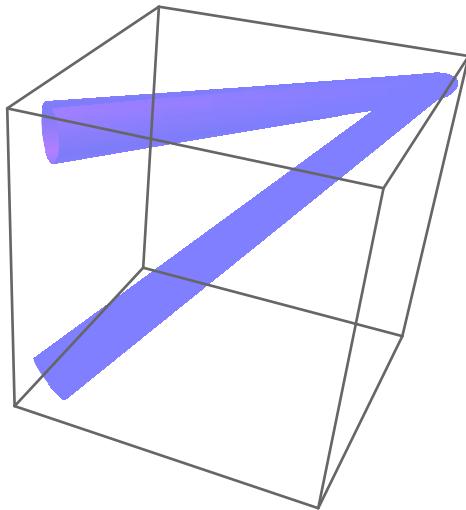
Coils around a ring:

```
>> PolarPlot[{1, 1 + Sin[20 t] / 5}, {t, 0, 2 Pi}]
```



A spring having 16 turns:

```
>> PolarPlot[Sqrt[t], {t, 0, 16 Pi}]
```



Splines

Splines

A Spline is a mathematical function used for interpolation or smoothing. Splines are used both in graphics and computation

BernsteinBasis

Bernstein polynomial basis (SciPy :WMA:

A Bernstein is a polynomial that is a linear combination of Bernstein basis polynomials. With the advent of computer graphics, Bernstein polynomials, restricted to the interval [0, 1], became important in the form of Bézier curves. `BernsteinBasis[d,n,x]` equals $\text{Binomial}[d, n] x^n (1-x)^{(d-n)}$ in the interval [0, 1] and zero elsewhere.

```
BernsteinBasis[d,n,x]  
returns the nth Bernstein basis of degree d at x.
```

```
>> BernsteinBasis[4, 3, 0.5]  
0.25
```

BezierCurve

WMA link

```
BezierCurve[{pt_1, pt_2 ...}]  
represents a Bézier curve with control points pi.  
The result is a curve by combining the Bézier curves when points are taken triples at a time.
```

Option:

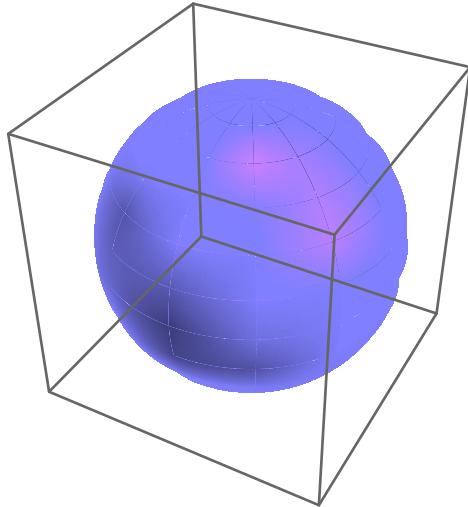
- `SplineDegree->d` specifies that the underlying polynomial basis should have maximal degree d.

Set up some points to form a simple zig-zag...

```
>> pts = {{0, 0}, {1, 1}, {2, -1}, {3, 0}};
```

=

```
>> Graphics[{Line[pts], Red, Point[pts]}]
```



A composite Bézier curve, shown in blue, smooths the zig zag. Control points are shown in red:

```
>> Graphics[{BezierCurve[pts], Blue, Line[pts], Red, Point[pts]}]
```

Extend points...

```
>> pts = {{0, 0}, {1, 1}, {2, -1}, {3, 0}, {5, 2}, {6, -1}, {7, 3}};
```

=

A longer composite Bézier curve and its control points:

```
>> Graphics[{BezierCurve[pts], Blue, Line[pts], Red, Point[pts]}]
```

Notice how the curve from the first to third point is not changed by any points outside the interval. The same is true for points three to five, and so on.

BezierFunction

WMA link

```
BezierFunction[{pt_1, pt_2, ...}]
```

returns a Bézier function for the curve defined by points pt_i . The embedding dimension for the curve represented by $\text{BezierFunction}[pt_1, pt_2, \dots]$ is given by the length of the lists pt_i .

```
>> f = BezierFunction[{{0, 0}, {1, 1}, {2, 0}, {3, 2}}];
```

=

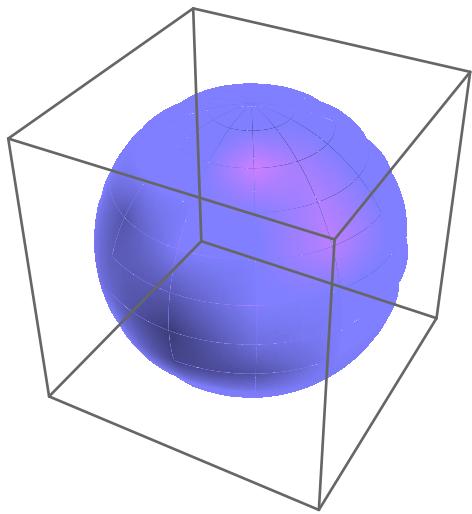
```
>> f[.5]
```

```
{1.5, 0.625}
```

=

Plotting the Bézier Function accoss a Bézier curve:

```
>> Module[{p = {{0, 0}, {1, 1}, {2, -1}, {4, 0}}}, Graphics[BezierCurve[p], Red, Point[Table[BezierFunction[p][x], {x, 0, 1, 0.1}]]]]]
```



Three-Dimensional Graphics

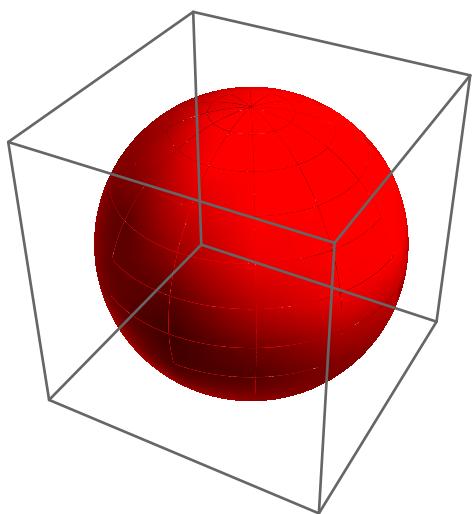
Three-Dimensional Graphics
Functions for working with 3D graphics.

Cone

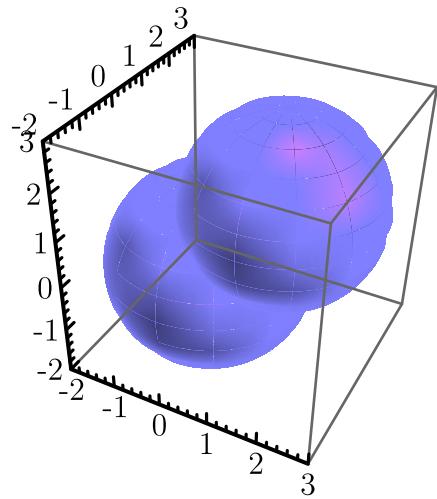
WMA link

```
Cone[{{x1, y1, z1}, {x2, y2, z2}}]
represents a cone of radius 1.
Cone[{{x1, y1, z1}, {x2, y2, z2}}, r]
is a cone of radius r starting at (x1, y1, z1) and ending at (x2, y2, z2).
Cone[{{x1, y1, z1}, {x2, y2, z2}, ... }, r]
is a collection cones of radius r.
```

```
>> Graphics3D[Cone[{{0, 0, 0}, {1, 1, 1}}, 1]]
```



```
>> Graphics3D[{Yellow, Cone[{{-1, 0, 0}, {1, 0, 0}, {0, 0, Sqrt[3]}}, {1, 1, Sqrt[3]}]}, 1]]
```



Cuboid

[WMA link](#)

Cuboid also known as interval, rectangle, square, cube, rectangular parallelepiped, tesseract, orthotope, and box.

`Cuboid[p_min]`

is a unit cube/square with its lower corner at point `p_min`.

`'Cuboid[p_min, p_max]`

is a 2d square with with lower corner `p_min` and upper corner `p_max`.

`Cuboid[{p_min, p_max}]`

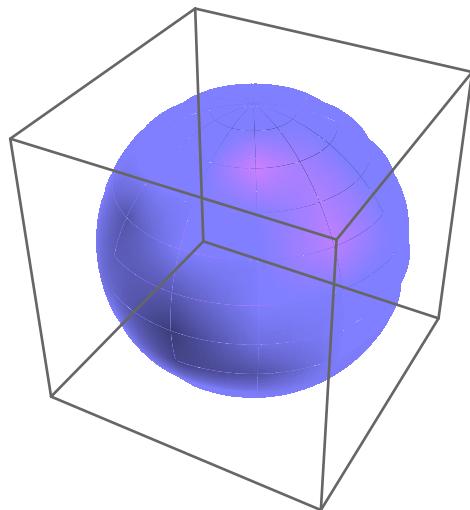
is a cuboid with lower corner `p_min` and upper corner `p_max`.

`Cuboid[{p1_min, p1_max, ...}]`

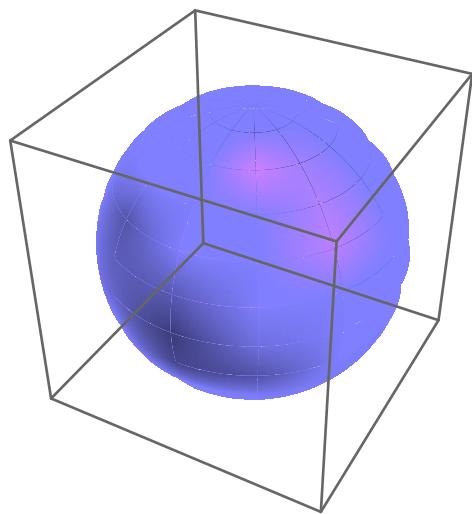
is a collection of cuboids.

`Cuboid[]` is equivalent to `Cuboid[{0,0,0}]`.

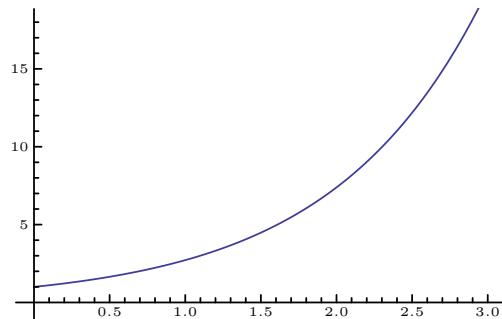
```
>> Graphics3D[Cuboid[{0, 0, 1}]]
```



```
>> Graphics3D[{Red, Cuboid[{{0, 0, 0}, {1, 1, 0.5}}], Blue, Cuboid[{{0.25, 0.25, 0.5}, {0.75, 0.75, 1}}]}]
```



```
>> Graphics[Cuboid[{0, 0}]]
```

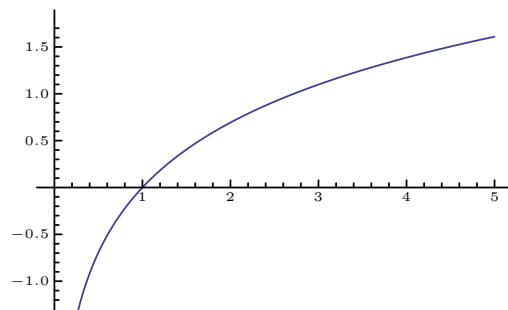


Cylinder

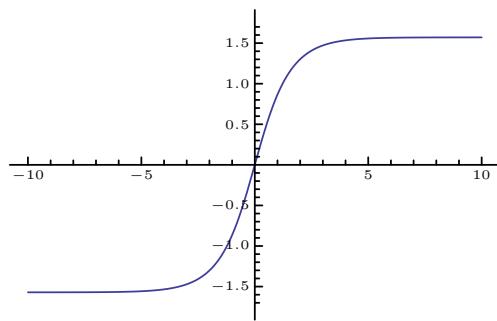
WMA link

```
Cylinder[{{x1, y1, z1}, {x2, y2, z2}}]
represents a cylinder of radius 1.
Cylinder[{{x1, y1, z1}, {x2, y2, z2}}, r]
is a cylinder of radius r starting at (x1, y1, z1) and ending at (x2, y2, z2).
Cylinder[{{x1, y1, z1}, {x2, y2, z2}, ...}, r]
is a collection cylinders of radius r.
```

```
>> Graphics3D[Cylinder[{{0, 0, 0}, {1, 1, 1}}, 1]]
```



```
>> Graphics3D[{Yellow, Cylinder[{{-1, 0, 0}, {1, 0, 0}, {0, 0, Sqrt[3]}, {1, 1, Sqrt[3]}}], 1}]]
```



Graphics

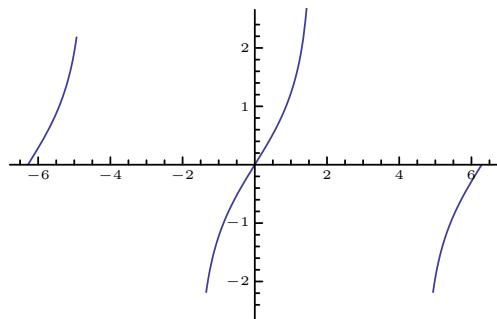
WMA link

`Graphics[primitives, options]`
represents a graphic.

Options include:

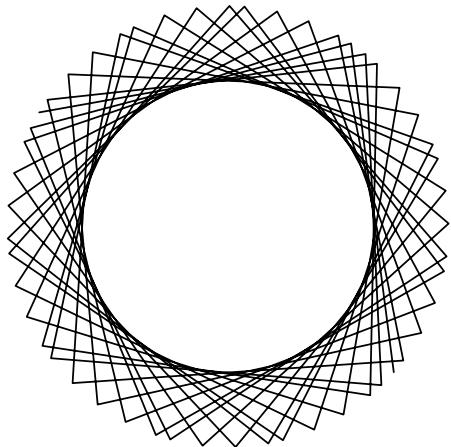
- Axes
- TicksStyle
- AxesStyle
- LabelStyle
- AspectRatio
- PlotRange
- PlotRangePadding
- ImageSize
- Background

```
>> Graphics[{Blue, Line[{{0,0}, {1,1}}]}]
```

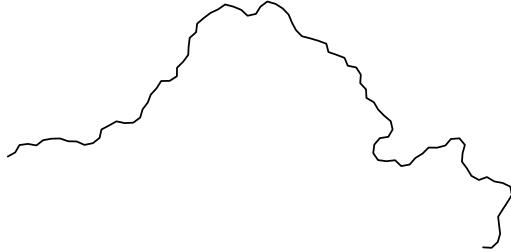


`Graphics` supports `PlotRange`:

```
>> Graphics[{Rectangle[{1, 1}]}], Axes -> True, PlotRange -> {{-2, 1.5}, {-1, 1.5}}]
```



```
>> Graphics[{Rectangle[], Red, Disk[{1, 0}]}], PlotRange -> {{0, 1}, {0, 1}}]
```



Graphics produces GraphicsBox boxes:

```
>> Graphics[Rectangle[]] // ToBoxes // Head
GraphicsBox
```

In TeXForm, Graphics produces Asymptote figures:

```
>> Graphics[Circle[]] // TeXForm
\begin{asy}
usepackage("amsmath");
size(5.8556cm, 5.8333cm);
draw(ellipse((175,175),175,175), rgb(0, 0, 0)+linewidth(0.66667));
clip(box((-0.33333,0.33333), (350.33,349.67)));
\end{asy}
```

Graphics3D

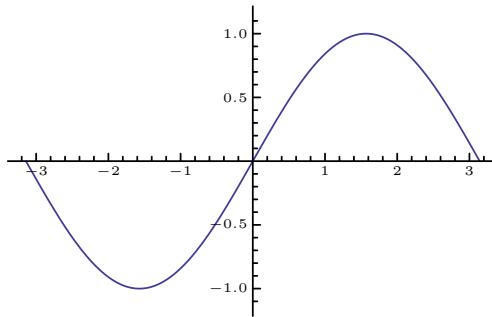
[WMA link](#)

Graphics3D [*primitives*, *options*]

represents a three-dimensional graphic.

See also the Section “Plotting” for a list of Plot options.

```
>> Graphics3D[Polygon[{{0,0,0}, {0,1,1}, {1,0,0}}]]
```



In `TeXForm`, `Graphics3D` creates Asymptote figures:

```
>> Graphics3D[Sphere[]] // TeXForm
```

```
\begin{asy}
import three;
import solids;
size(6.6667cm, 6.6667cm);
currentprojection=perspective(2.6,-4.8,4.0);
currentlight=light(rgb(0.5,0.5,1), specular=red, (2,0,2), (2,2,2), (0,2,2));
// Sphere3DBox
draw(surface(sphere((0, 0, 0), 1)), rgb(1,1,1)+opacity(1));
draw((-1,-1,-1)-(1,-1,-1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,-1)-(1,1,-1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,1)-(1,-1,1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,1,1)-(1,1,1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,-1)-(1,-1,-1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((1,-1,-1)-(1,1,-1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,1)-(1,-1,1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((-1,-1,-1)-(1,1,-1), rgb(0.4, 0.4, 0.4)+linewidth(1));
draw((1,-1,-1)-(1,-1,1), rgb(0.4, 0.4, 0.4)+linewidth(1));
\end{asy}
```

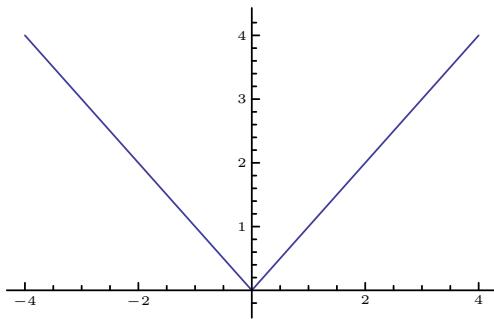
RGBColor

WMA link

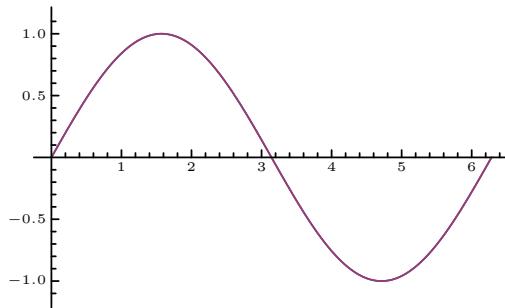
`RGBColor[r, g, b]`

represents a color with the specified red, green and blue components.

```
>> Graphics[MapIndexed[{RGBColor @@ #1, Disk[2*#2 ~Join~ {0}]} &,
IdentityMatrix[3]], ImageSize->Small]
```



```
>> RGBColor[0, 1, 0]
```

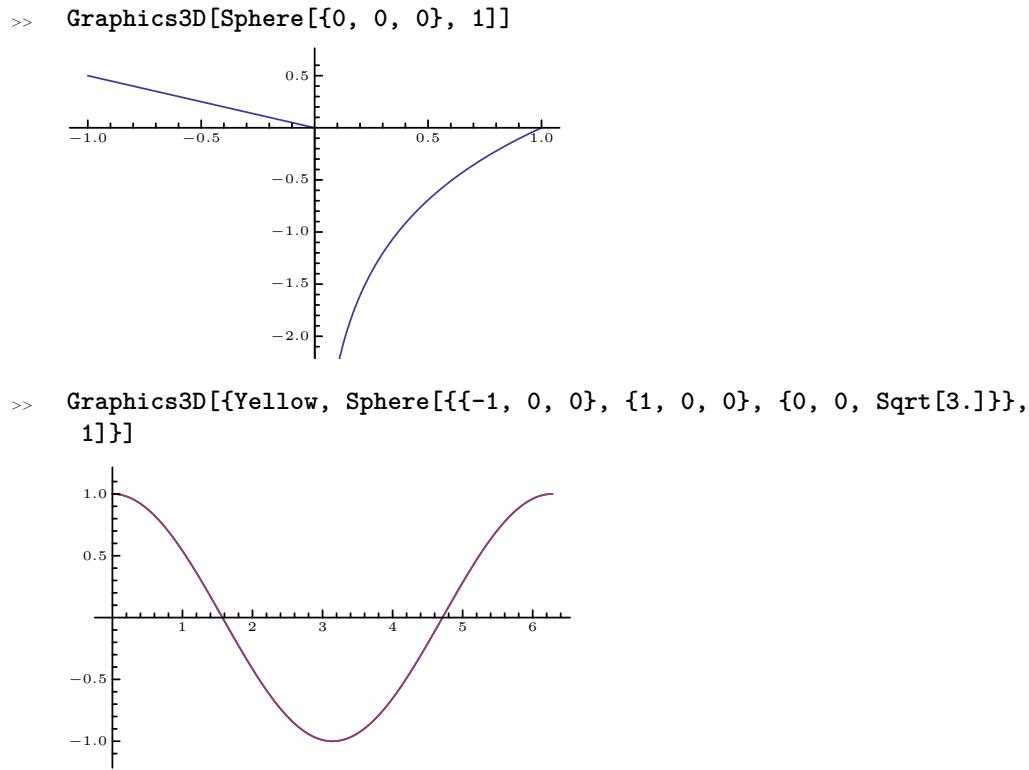


```
>> RGBColor[0, 1, 0] // ToBoxes
StyleBox[GraphicsBox[{EdgeForm[RGBColor[0, 0, 0]], RGBColor[
0, 1, 0], RectangleBox[{0, 0}], AspectRatio -> Automatic, Axes
-> False, AxesStyle -> {}, Background -> Automatic, ImageSize
-> 16, LabelStyle -> {}, PlotRange -> Automatic, PlotRangePadding
-> Automatic, TicksStyle -> {}}, ImageSizeMultipliers
-> {1, 1}, ShowStringCharacters -> True]]
```

Sphere

WMA link

```
Sphere[{x, y, z}]
  is a sphere of radius 1 centered at the point {x, y, z}.
Sphere[{x, y, z}, r]
  is a sphere of radius r centered at the point {x, y, z}.
Sphere[{{x1, y1, z1}, {x2, y2, z2}, ...}, r]
  is a collection spheres of radius r centered at the points {x1, y2, z2}, {x2, y2, z2}, ...]
```



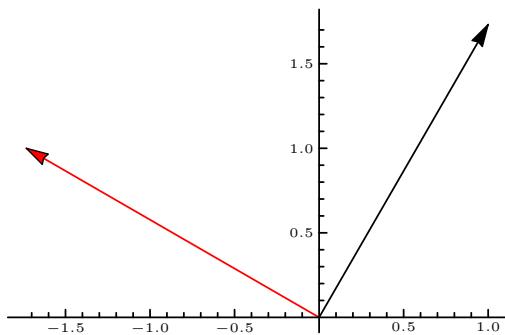
Tube

[WMA link](#)

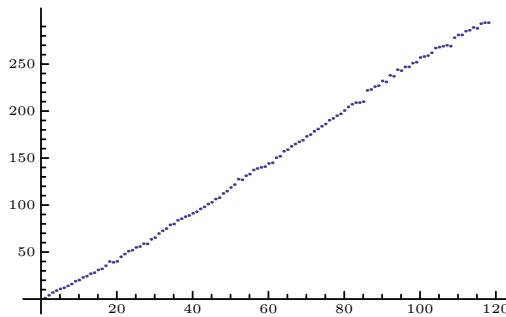
```
Tube[{p1, p2, ...}]
represents a tube passing through  $p_1, p_2, \dots$  with radius 1.
```

```
Tube[{p1, p2, ...}, r]
represents a tube with radius  $r$ .
```

```
>> Graphics3D[Tube[{{0,0,0}, {1,1,1}}]]
```



```
>> Graphics3D[Tube[{{0,0,0}, {1,1,1}, {0, 0, 1}}, 0.1]]
```



Uniform Polyhedra

Uniform Polyhedra

Uniform polyhedra is the grouping of platonic solids, Archimedean solids, and regular star polyhedra.

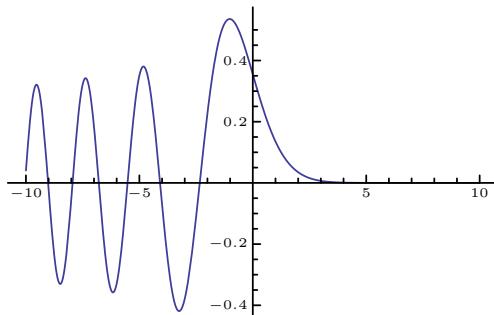
Dodecahedron

WMA link

```
Dodecahedron[]
```

a regular dodecahedron centered at the origin with unit edge length.

```
>> Graphics3D[Dodecahedron[]]
```



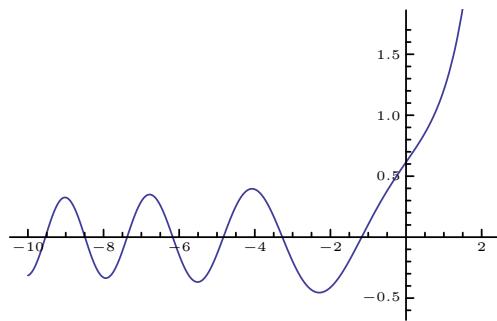
Icosahedron

WMA link

```
Icosahedron[]
```

a regular Icosahedron centered at the origin with unit edge length.

```
>> Graphics3D[Icosahedron[]]
```



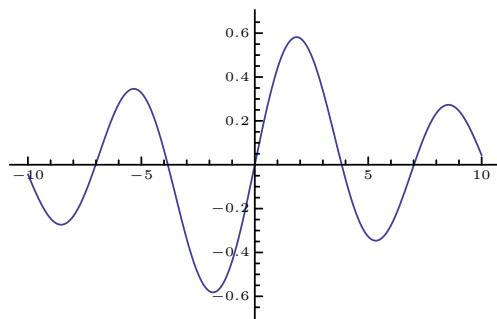
Octahedron

WMA link

```
Octahedron[]
```

a regular octahedron centered at the origin with unit edge length.

```
>> Graphics3D[{Red, Octahedron[]}]
```



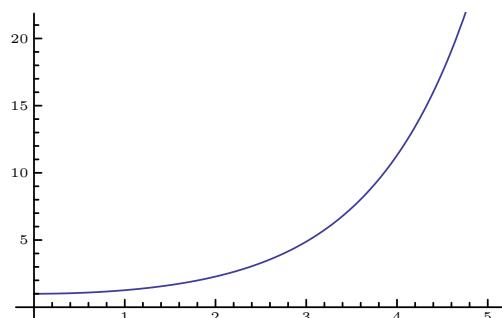
Tetrahedron

WMA link

```
Tetrahedron[]
```

a regular tetrahedron centered at the origin with unit edge length.

```
>> Graphics3D[Tetrahedron[{{0,0,0}, {1,1,1}}, 2], Axes->True]
```

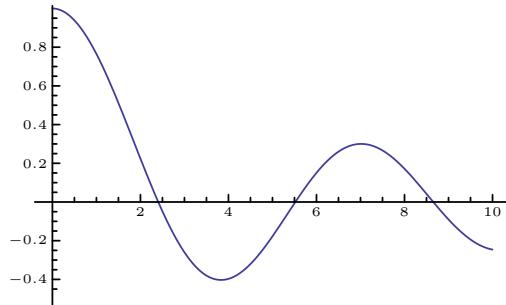


UniformPolyhedron

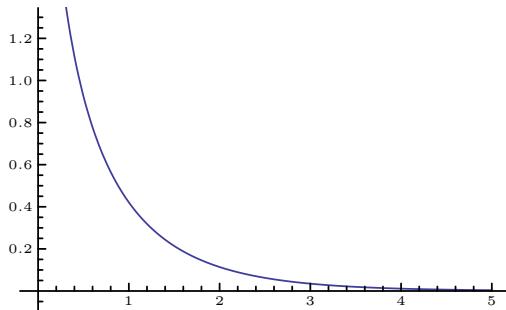
WMA link

```
UniformPolyhedron[{"name"}]
return a uniform polyhedron with the given name.
Names are "tetrahedron", "octahedron", "dodecahedron", or "icosahedron".
```

```
>> Graphics3D[UniformPolyhedron["octahedron"]]
```



```
>> Graphics3D[UniformPolyhedron["dodecahedron"]]
```



```
>> Graphics3D[{"Brown", UniformPolyhedron["tetrahedron"]}]
```

25. Image Manipulation

For the full compliment of functions, you need to have scikit-image installed.

Contents

Basic Image Processing	281	ImageMultiply . . . 295	Miscellaneous
Blur	282	ImageSubtract . . . 295	image-related
ImageAdjust . . .	283	WordCloud 296	functions 303
ImagePartition . . .	283	Image Filters 296	EdgeDetect 305
Sharpen	284	GaussianFilter . . . 297	RandomImage 305
Threshold	285	ImageConvolve . . . 299	TextRecognize 306
Geometric Operations	285	MaxFilter 299	Morphological Image
ImageReflect . . .	286	MedianFilter 300	Processing 306
ImageResize . . .	287	MinFilter 301	Closing 306
ImageRotate . . .	289	Image Properties 301	Dilation 307
Image Colors	289	ImageAspectRatio . . . 301	Erosion 307
Binarize	291	ImageChannels 302	Morphological-
ColorCombine . . .	291	ImageData 302	Components 308
ColorQuantize . . .	292	ImageDimensions . . . 302	Opening 308
ColorSeparate . . .	292	ImageType 303	Operations on Image
Colorize	293	Image testing 303	Structure 308
ImageColorSpace .	293	BinaryImageQ 303	ImageTake 309
Image Compositions .	293	ImageQ 303	Pixel Operations 309
ImageAdd	294		PixelValue 309
			PixelValuePositions 310

Basic Image Processing

Basic Image Processing

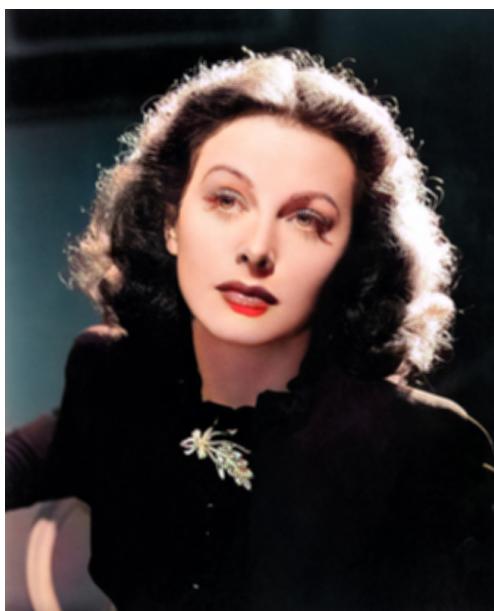
Blur

WMA link

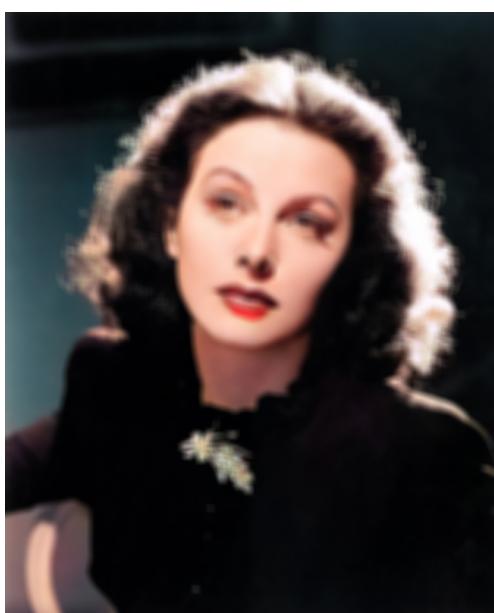
```
Blur[image]
gives a blurred version of image.
Blur[image, r]
blurs image with a kernel of size r.
```

```
>> hedy = Import["ExampleData/hedy.tif"];
```

>> Blur [hedy]



>> Blur [hedy, 5]



ImageAdjust

WMA link

```

ImageAdjust[image]
    adjusts the levels in image.
ImageAdjust[image, c]
    adjusts the contrast in image by c.
ImageAdjust[image, {c, b}]
    adjusts the contrast c, and brightness b in image.
ImageAdjust[image, {c, b, g}]
    adjusts the contrast c, brightness b, and gamma g in image.

```

```

>> hedy = Import["ExampleData/hedy.tif"];
>> ImageAdjust[hedy]

```



ImagePartition

WMA link

```

ImagePartition[image, s]
    Partitions an image into an array of s x s pixel subimages.
ImagePartition[image, {w, h}]
    Partitions an image into an array of w x h pixel subimages.

```

```

>> hedy = Import["ExampleData/hedy.tif"];
>> ImageDimensions[hedy]
{646, 800}
>> ImagePartition[hedy, 256]
>> ImagePartition[hedy, {512, 128}]

```

Sharpen

WMA link

Sharpen[*image*]

gives a sharpened version of *image*.

Sharpen[*image*, *r*]

sharpens *image* with a kernel of size *r*.

```
>> hedy = Import["ExampleData/hedy.tif"];
```

```
>> Sharpen[hedy]
```



```
>> Sharpen[hedy, 5]
```



Threshold

WMA link

`Threshold[image]`

gives a value suitable for binarizing *image*.

The option "Method" may be "Cluster" (use Otsu's threshold), "Median", or "Mean".

```
>> img = Import["ExampleData/hedy.tif"];  
  
>> Threshold[img]  
0.408203  
  
>> Binarize[img, %]
```



```
>> Threshold[img, Method -> "Mean"]  
0.22086  
  
>> Threshold[img, Method -> "Median"]  
0.0593961
```

Geometric Operations

Geometric Operation

ImageReflect

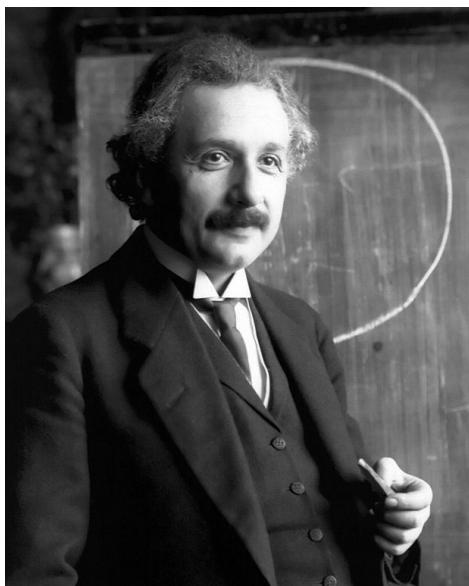
WMA link

```
ImageReflect [image]
  Flips image top to bottom.
ImageReflect [image, side]
  Flips image so that side is interchanged with its opposite.
ImageReflect [image, side_1 -> side_2]
  Flips image so that side_1 is interchanged with side_2.
```

```
>> ein = Import["ExampleData/Einstein.jpg"];
>> ImageReflect[ein]
```



```
>> ImageReflect[ein, Left]
```



```
>> ImageReflect[ein, Left -> Top]
```

ImageResize

WMA link

```
ImageResize[image, width]
ImageResize[image, {width, height}]
```

The Resampling option can be used to specify how to resample the image. Options are:

- Automatic
- Bicubic
- Bilinear
- Box
- Hamming
- Lanczos
- Nearest

See Pillow Filters for a description of these.

```
>> alice = Import["ExampleData/MadTeaParty.gif"]
```



```
>> shape = ImageDimensions[alice]
{640, 487}
>> ImageResize[alice, shape / 2]
```



The default sampling method is “Bicubic” which has pretty good upscaling and downscaling quality. However “Box” is the fastest:

```
>> ImageResize[alice, shape / 2, Resampling -> "Box"]
```



ImageRotate

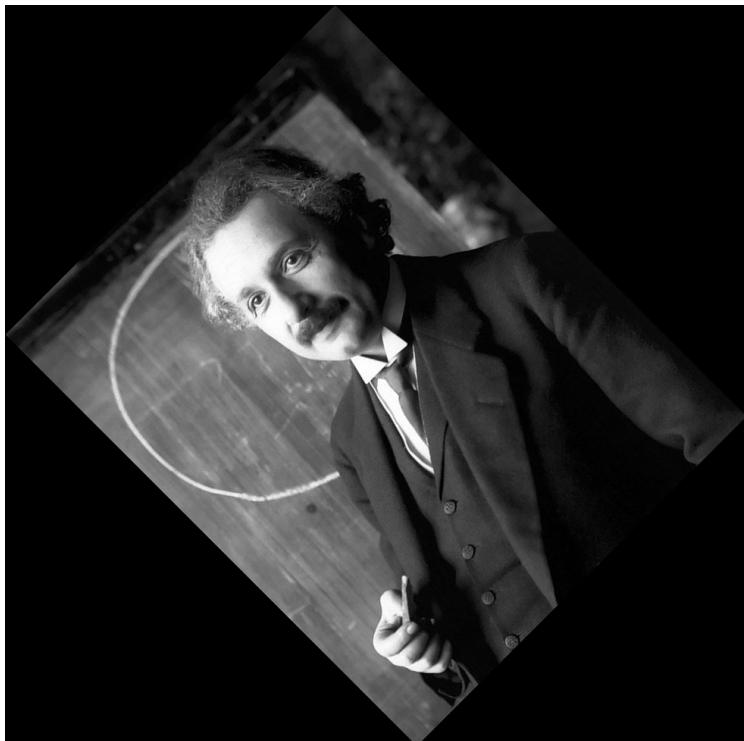
WMA link

```
ImageRotate[image]
  Rotates image 90 degrees counterclockwise.
ImageRotate[image, theta]
  Rotates image by a given angle theta
```

```
>> ein = Import["ExampleData/Einstein.jpg"];
>> ImageRotate[ein]
```



```
>> ImageRotate[ein, 45 Degree]
```



```
>> ImageRotate[ein, Pi / 4]
```

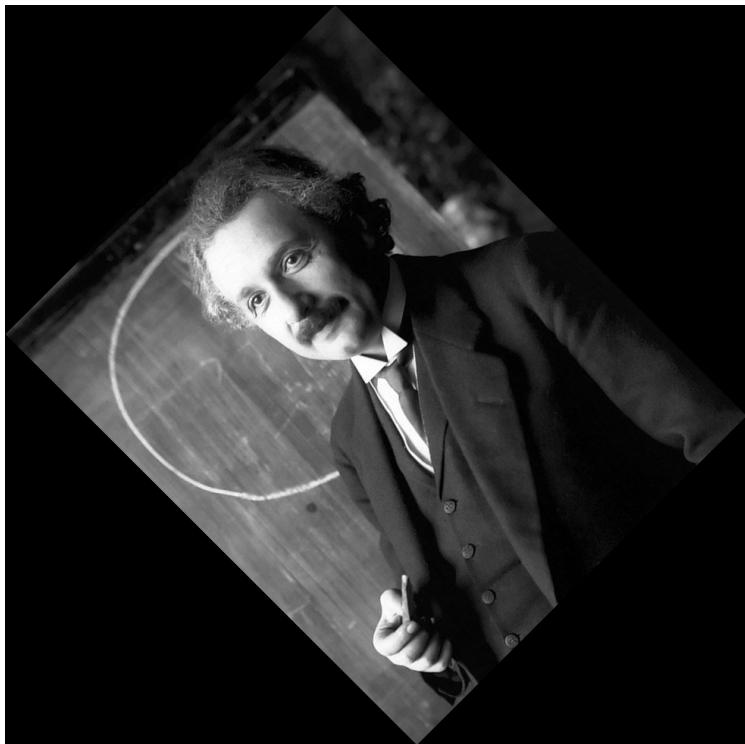


Image Colors

Image Color

Binarize

WMA link

`Binarize[image]`

gives a binarized version of *image*, in which each pixel is either 0 or 1.

`Binarize[image, t]`

map values $x > t$ to 1, and values $x \leq t$ to 0.

`Binarize[image, {t1, t2}]`

map $t1 < x < t2$ to 1, and all other values to 0.

```
>> hedy = Import["ExampleData/hedy.tif"];
```

>> Binarize[hedy]



>> Binarize[hedy, 0.7]



```
>> Binarize[hedy, {0.2, 0.6}]
```



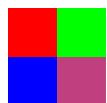
ColorCombine

WMA link

```
ColorCombine[channels, colorspace]
```

Gives an image with *colorspace* and the respective components described by the given channels.

```
>> ColorCombine[{{{1, 0}, {0, 0.75}}, {{0, 1}, {0, 0.25}}, {{0, 0}, {1, 0.5}}}, "RGB"]
```



ColorQuantize

WMA link

```
ColorQuantize[image, n]
```

gives a version of *image* using only *n* colors.

```
>> img = Import["ExampleData/hedy.tif"];
```

```
>> ColorQuantize[img, 6]
```



ColorSeparate

WMA link

```
ColorSeparate[image]
```

Gives each channel of *image* as a separate grayscale image.

```
>> img = Import["ExampleData/hedy.tif"];
```

```
>> ColorSeparate[img]
```

Colorize

WMA link

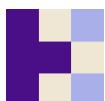
```
Colorize[values]
```

returns an image where each number in the rectangular matrix *values* is a pixel and each occurrence of the same number is displayed in the same unique color, which is different from the colors of all non-identical numbers.

```
Colorize[image]
```

gives a colorized version of *image*.

```
>> Colorize[{{1.3, 2.1, 1.5}, {1.3, 1.3, 2.1}, {1.3, 2.1, 1.5}}]
```



```
>> Colorize[{{1, 2}, {2, 2}, {2, 3}}, ColorFunction -> (Blend[{White, Blue}, #]&)]
```



ImageColorSpace

WMA link

```
ImageColorSpace[image]  
gives image's color space, e.g. "RGB" or "CMYK".
```

```
>> img = Import["ExampleData/MadTeaParty.gif"];  
  
>> ImageColorSpace[img]  
  
>> img = Import["ExampleData/sunflowers.jpg"];  
  
>> ImageColorSpace[img]
```

Image Compositions

Image Composition

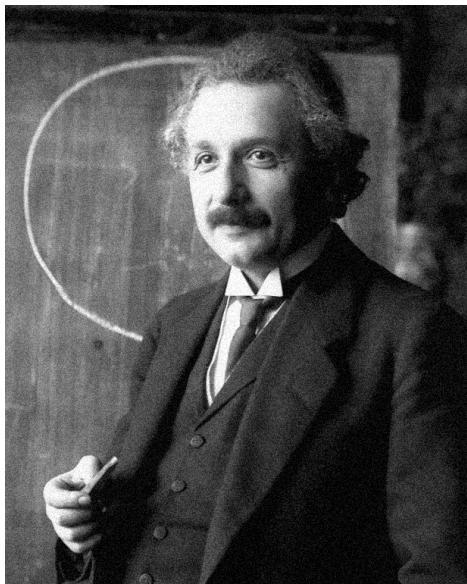
ImageAdd

WMA link

```
ImageAdd[image, expr_1, expr_2, ...]  
adds all expri to image where each expri must be an image or a real number.
```

```
>> i = Image[{{0, 0.5, 0.2, 0.1, 0.9}, {1.0, 0.1, 0.3, 0.8, 0.6}}];  
  
>> ImageAdd[i, 0.5]  
  
  
>> ImageAdd[i, i]  
  
  
>> ein = Import["ExampleData/Einstein.jpg"];  
  
>> noise = RandomImage[{-0.1, 0.1}, ImageDimensions[ein]];
```

```
>> ImageAdd[noise, ein]
```



```
>> hedy = Import["ExampleData/hedy.tif"];
```

```
>> noise = RandomImage[{-0.2, 0.2}, ImageDimensions[hedy], ColorSpace -> "RGB"];
```

```
>> ImageAdd[noise, hedy]
```



ImageMultiply

[WMA link](#)

```
ImageMultiply[image, expr_1, expr_2, ...]
```

multiplies all $expr_i$ with $image$ where each $expr_i$ must be an image or a real number.

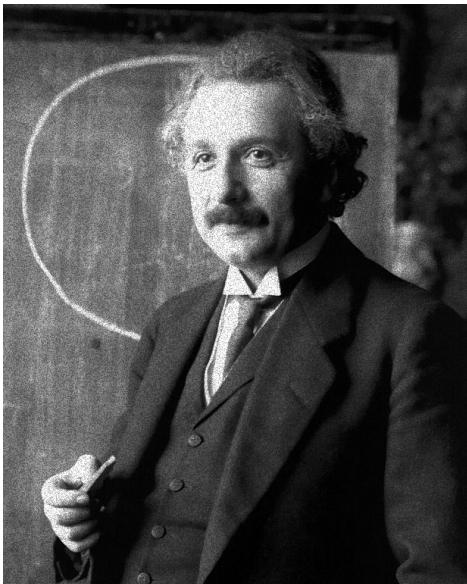
```

>> i = Image[{{0, 0.5, 0.2, 0.1, 0.9}, {1.0, 0.1, 0.3, 0.8, 0.6}}];
>> ImageMultiply[i, 0.2]

>> ImageMultiply[i, i]

>> ein = Import["ExampleData/Einstein.jpg"];
>> noise = RandomImage[{0.7, 1.3}, ImageDimensions[ein]];
>> ImageMultiply[noise, ein]

```



ImageSubtract

[WMA link](#)

```
ImageSubtract[image, expr_1, expr_2, ...]
subtracts all expr_i from image where each expr_i must be an image or a real number.
```

```

>> i = Image[{{0, 0.5, 0.2, 0.1, 0.9}, {1.0, 0.1, 0.3, 0.8, 0.6}}];
>> ImageSubtract[i, 0.2]

>> ImageSubtract[i, i]


```

WordCloud

[WMA link](#)

```
WordCloud[{word1, word2, ...}]
```

Gives a word cloud with the given list of words.

```
WordCloud[{weight1 -> word1, weight2 -> word2, ...}]
```

Gives a word cloud with the words weighted using the given weights.

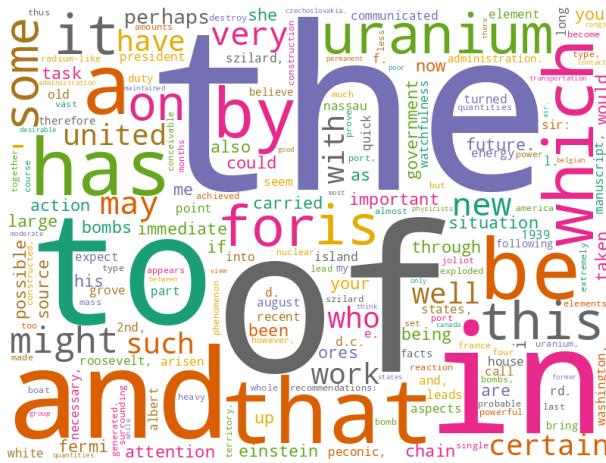
`WordCloud[{weight1, weight2, ...} -> {word1, word2, ...}]`

Also gives a word cloud with the words weighted using the given weights.

```
WordCloud[{{word1, weight1}, {word2, weight2}, ...}]
```

Gives a word cloud with the words weighted using the given weights.

```
>>> WordCloud[StringSplit[Import["ExampleData/EinsteinSzilLetter.txt",  
CharacterEncoding->"UTF8"]]]
```



```
>> WordCloud[Range[50] -> ToString /@ Range[50]]
```



Image Filters

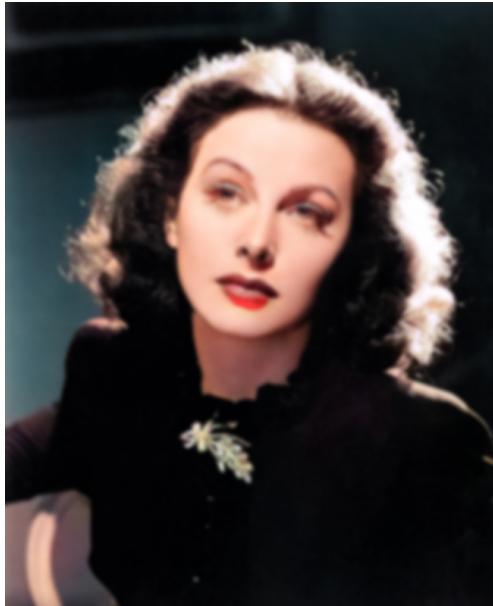
Image Filter

GaussianFilter

WMA link

```
GaussianFilter[image, r]  
blurs image using a Gaussian blur filter of radius r.
```

```
>> hedy = Import["ExampleData/hedy.tif"];  
>> GaussianFilter[hedy, 2.5]
```



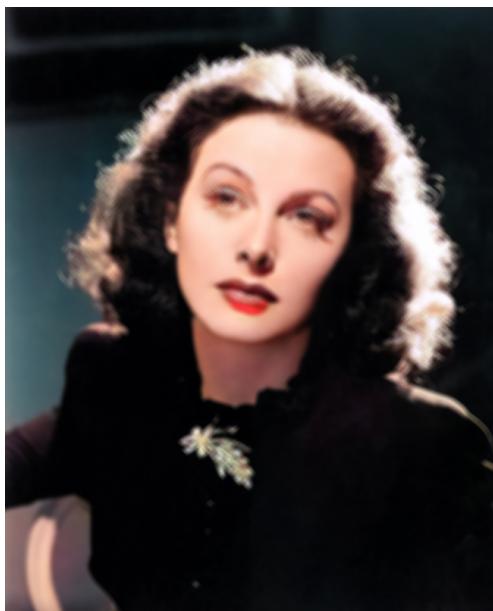
ImageConvolve

WMA link

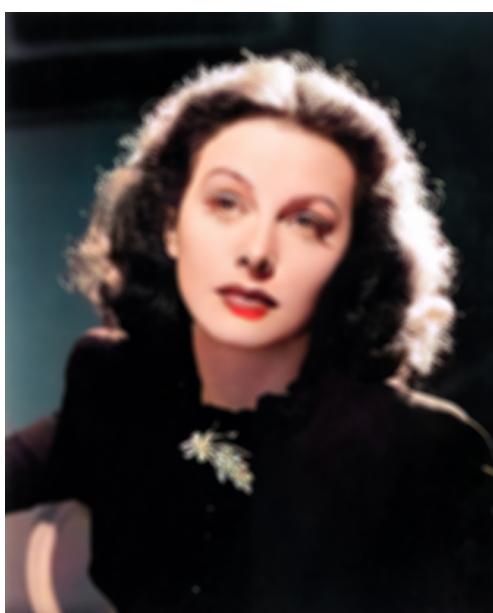
```
ImageConvolve[image, kernel]  
Computes the convolution of image using kernel.
```

```
>> hedy = Import["ExampleData/hedy.tif"];
```

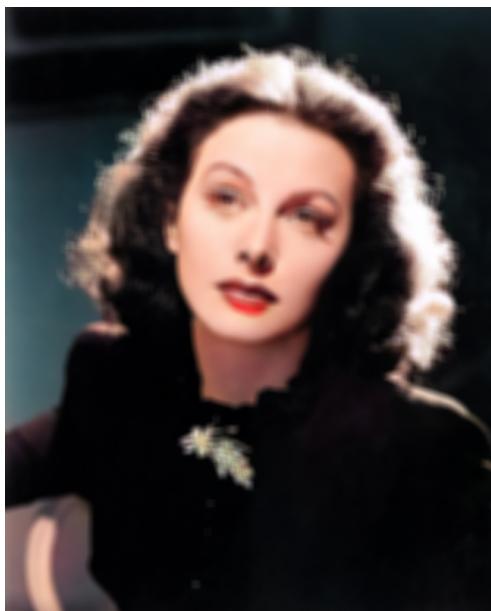
```
>> ImageConvolve[hedy, DiamondMatrix[5] / 61]
```



```
>> ImageConvolve[hedy, DiskMatrix[5] / 97]
```



```
>> ImageConvolve[hedy, BoxMatrix[5] / 121]
```



MaxFilter

WMA link

```
MaxFilter[image, r]
```

gives *image* with a maximum filter of radius *r* applied on it. This always picks the largest value in the filter's area.

```
>> hedy = Import["ExampleData/hedy.tif"];  
>> MaxFilter[hedy, 5]
```



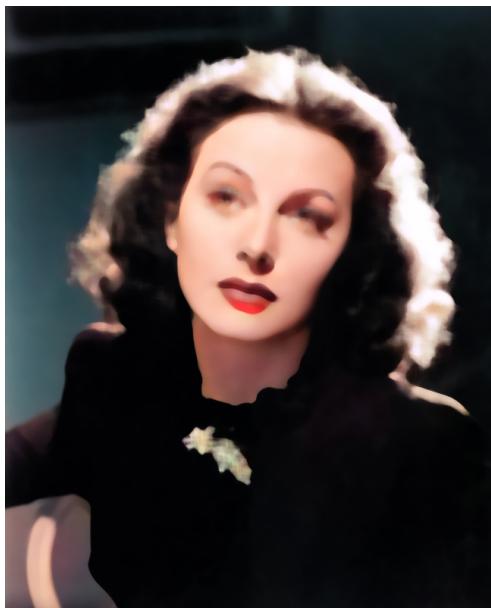
MedianFilter

WMA link

```
MedianFilter[image, r]
```

gives *image* with a median filter of radius *r* applied on it. This always picks the median value in the filter's area.

```
>> hedy = Import["ExampleData/hedy.tif"];  
>> MedianFilter[hedy, 5]
```



MinFilter

WMA link

```
MinFilter[image, r]
```

gives *image* with a minimum filter of radius *r* applied on it. This always picks the smallest value in the filter's area.

```
>> hedy = Import["ExampleData/hedy.tif"];
```

```
>> MinFilter[hedy, 5]
```

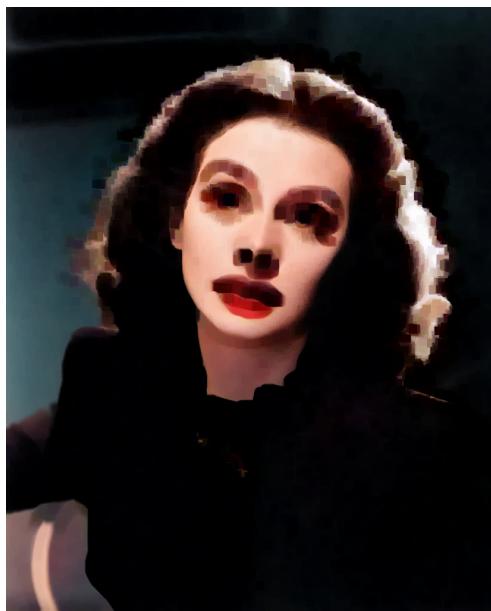


Image Properties

Image Properties

ImageAspectRatio

WMA link

```
ImageAspectRatio[image]  
gives the aspect ratio of image.
```

```
>> img = Import["ExampleData/hedy.tif"];  
>> ImageAspectRatio[img]  
400  
323  
>> ImageAspectRatio[Image[{{0, 1}, {1, 0}, {1, 1}}]]  
3  
2
```

ImageChannels

WMA link

```
ImageChannels[image]  
gives the number of channels in image.
```

```
>> ImageChannels[Image[{{0, 1}, {1, 0}}]]  
1
```

```
>> img = Import["ExampleData/hedy.tif"];
>> ImageChannels[img]
3
```

ImageData

WMA link

ImageData[*image*]
gives a list of all color values of *image* as a matrix.
ImageData[*image*, *stype*]
gives a list of color values in type *stype*.

```
>> img = Image[{{0.2, 0.4}, {0.9, 0.6}, {0.5, 0.8}}];
>> ImageData[img]
{{0.2,0.4},{0.9,0.6},{0.5,0.8}}
>> ImageData[img, "Byte"]
{{51,102},{229,153},{127,204}}
>> ImageData[Image[{{0, 1}, {1, 0}, {1, 1}}], "Bit"]
{{0,1},{1,0},{1,1}}
```

ImageDimensions

WMA link

ImageDimensions[*image*]
Returns the dimensions {*width*, *height*} of *image* in pixels.

```
>> hedy = Import["ExampleData/hedy.tif"];
>> ImageDimensions[hedy]
{646,800}
>> ImageDimensions[RandomImage[1, {50, 70}]]
{50,70}
```

ImageType

WMA link

ImageType[*image*]
gives the interval storage type of *image*, e.g. "Real", "Bit32", or "Bit".

```
>> img = Import["ExampleData/hedy.tif"];
>> ImageType[img]
Byte
```

```
>> ImageType[Image[{{0, 1}, {1, 0}}]]  
Real  
>> ImageType[Binarize[img]]  
Bit
```

Image testing

Image testing

BinaryImageQ

WMA link

```
BinaryImageQ[$image]  
returns True if the pixels of $image are binary bit values, and False otherwise.
```

```
>> img = Import["ExampleData/hedy.tif"];  
>> BinaryImageQ[img]  
False  
>> BinaryImageQ[Binarize[img]]  
True
```

ImageQ

WMA link

```
ImageQ[Image[$pixels]]  
returns True if $pixels has dimensions from which an Image can be constructed, and False otherwise.
```

```
>> ImageQ[Image[{{0, 1}, {1, 0}}]]  
True  
>> ImageQ[Image[{{{0, 0, 0}, {0, 1, 0}}, {{0, 1, 0}, {0, 1, 1}}}] ]  
True  
>> ImageQ[Image[{{{0, 0, 0}, {0, 1}}, {{0, 1, 0}, {0, 1, 1}}}] ]  
False  
>> ImageQ[Image[{1, 0, 1}]]  
False  
>> ImageQ["abc"]  
False
```

Miscellaneous image-related functions

Miscellaneous image-related function

EdgeDetect

WMA link

`EdgeDetect [image]`

returns an image showing the edges in *image*.

```
>> hedy = Import["ExampleData/hedy.tif"];  
>> EdgeDetect [hedy]
```



```
>> EdgeDetect [hedy, 5]
```



```
>> EdgeDetect[hedy, 4, 0.5]
```

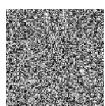


RandomImage

WMA link

```
RandomImage[max]  
    creates an image of random pixels with values 0 to max.  
RandomImage[{min, max}]  
    creates an image of random pixels with values min to max.  
RandomImage[..., size]  
    creates an image of the given size.
```

```
>> RandomImage[1, {100, 100}]
```



TextRecognize

WMA link

```
TextRecognize[image]  
    Recognizes text in image and returns it as a String.
```

```
>> textimage = Import["ExampleData/TextRecognize.png"]
```

```
TextRecognize[image]  
    Recognizes text in image and returns it as a String.
```

```
>> TextRecognize[textimage]
TextRecognize[ image]
```

Recognizes text in image and returns it as a String.

Morphological Image Processing

Morphological Image Processing

Closing

WMA link

```
Closing[image, ker]
```

Gives the morphological closing of *image* with respect to structuring element *ker*.

```
>> ein = Import["ExampleData/Einstein.jpg"];
>> Closing[ein, 2.5]
```



Dilation

WMA link

```
Dilation[image, ker]
```

Gives the morphological dilation of *image* with respect to structuring element *ker*.

```
>> ein = Import["ExampleData/Einstein.jpg"];
```

```
>> Dilation[ein, 2.5]
```



Erosion

WMA link

```
Erosion[image, ker]
```

Gives the morphological erosion of *image* with respect to structuring element *ker*.

```
>> ein = Import["ExampleData/Einstein.jpg"];  
>> Erosion[ein, 2.5]
```



MorphologicalComponents

WMA link

MorphologicalComponents[*image*]

Builds a 2-D array in which each pixel of *image* is replaced by an integer index representing the connected foreground image component in which the pixel lies.

MorphologicalComponents[*image*, *threshold*]

consider any pixel with a value above *threshold* as the foreground.

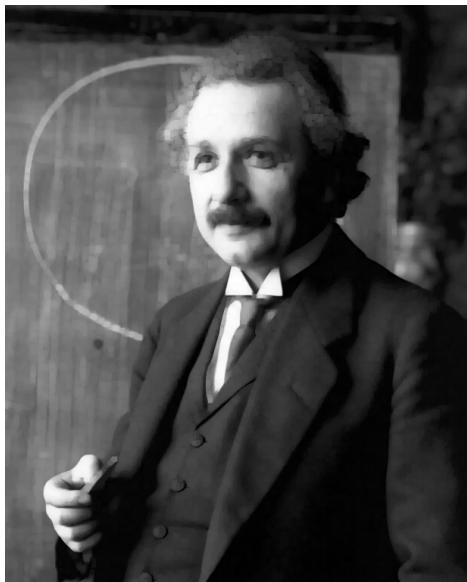
Opening

WMA link

Opening[*image*, *ker*]

Gives the morphological opening of *image* with respect to structuring element *ker*.

```
>> ein = Import["ExampleData/Einstein.jpg"];  
>> Opening[ein, 2.5]
```



Operations on Image Structure

Operations on Image Structure

ImageTake

Extract Image parts WMA link

ImageTake[*image*, *n*]

gives the first *n* rows of *image*.

ImageTake[*image*, -*n*]

gives the last *n* rows of *image*.

ImageTake[*image*, {*r*₁, *r*₂}]

gives rows *r*₁, ..., *r*₂ of *image*.

ImageTake[*image*, {*r*₁, *r*₂}, {*c*₁, *c*₂}]

gives a cropped version of *image*.

Crop to the include only the upper half (244 rows) of an image:

```
>> alice = Import["ExampleData/MadTeaParty.gif"]; ImageTake[alice, 244]
```



Now crop to the include the lower half of that image:

```
>> ImageTake[alice, -244]
```



Just the text around the hat:

```
>> ImageTake[alice, {40, 150}, {500, 600}]
```



Pixel Operations

Pixel Operation

PixelValue

WMA link

```
PixelValue[image, {x, y}]
```

gives the value of the pixel at position $\{x, y\}$ in *image*.

```
>> hedy = Import["ExampleData/hedy.tif"];
```

```
>> PixelValue[hedy, {1, 1}]
{0.439216, 0.356863, 0.337255}
```

PixelValuePositions

WMA link

```
PixelValuePositions[image, val]
```

gives the positions of all pixels in *image* that have value *val*.

```
>> PixelValuePositions[Image[{{0, 1}, {1, 0}, {1, 1}}], 1]
{{1, 1}, {1, 2}, {2, 1}, {2, 3}}
```

```
>> PixelValuePositions[Image[{{0.2, 0.4}, {0.9, 0.6}, {0.3, 0.8}}], 0.5,  
0.15]  
{ {2,2}, {2,3} }  
  
>> hedy = Import["ExampleData/hedy.tif"];  
  
>> PixelValuePositions[hedy, 1, 0][[1]]  
{101,491,1}  
  
>> PixelValue[hedy, {180, 192}]  
{0.00784314,0.00784314,0.0156863}
```

26. Importing and Exporting

Many kinds data formats can be read into *Mathics3*. Variable \$ExportFormats of section 26 contains a list of file formats that are supported by Export of section 26, while \$InputFormats of section ?? does the corresponding thing for Import of section 26.

Contents

\$ExportFormats	311	B64Encode	312	ImportString	314
\$ExtensionMappings	311	Export	313	RegisterExport	315
\$FormatMappings	312	ExportString	313	RegisterImport	316
\$ImportFormats	312	FileFormat	313	URLFetch	317
B64Decode	312	Import	314		

\$ExportFormats

WMA link

```
$ExportFormats
    returns a list of file formats supported by Export.
```

```
>> $ExportFormats
{BMP, Base64, CSV, GIF, JPEG, JPEG2000, PBM, PCX, PGM, PNG, PPM, SVG, TIFF, Text, asy}
```

\$ExtensionMappings

```
System`ConvertersDump`$ExtensionMappings
    Returns a list of associations between file extensions and file types.
```

The format associated to the extension ".jpg"

```
>> ".jpg"/. System`ConvertersDump`$ExtensionMappings
JPEG
```

\$FormatMappings

```
System`ConverterDump`$FormatMappings
    Returns a list of associations between file extensions and file types.
```

The list of MIME types associated to the extension JPEG:

```
>> Select[System`ConvertersDump`$FormatMappings, (#1[[2]]=="JPEG")&] [[All
, 1]]
{APPLICATION/JPG, APPLICATION/X-JPG, IMAGE/JPEG, IMAGE/JPG, IMAGE/PJPEG, JPEG, JPG}
```

\$ImportFormats

WMA link

```
$ImportFormats
returns a list of file formats supported by Import.
```

```
>> $ImportFormats
{BMP, Base64, CSV, GIF, HTML, ICO, JPEG, JPEG2000, JSON, PBM, PCX, PGM, PNG, PPM, Package, TGA, TIFF, Te
```

B64Decode

WMA link

```
System`Convert`B64Dump`B64Decode[string]
Decode string in Base64 coding to an expression.
```

```
>> System`Convert`B64Dump`B64Decode["R!="]
String "R!=" is not a valid b64 encoded string.
$Failed
```

B64Encode

WMA link

```
System`Convert`B64Dump`B64Encode[expr]
Encodes expr in Base64 coding
```

```
>> System`Convert`B64Dump`B64Encode["Hello world"]
SGVsbG8gd29ybGQ=

>> System`Convert`B64Dump`B64Decode[%]
Hello world

>> System`Convert`B64Dump`B64Encode[Integrate[f[x], {x, 0, 2}]]
SW50ZWdyYXRlW2ZbeF0sIHt4LCAwLCAyfV0=

>> System`Convert`B64Dump`B64Decode[%]
Integrate[f[x], {x, 0, 2}]
```

Export

WMA link

```
Export["file.ext", expr]
    exports expr to a file, using the extension ext to determine the format.
Export["file", expr, "format"]
    exports expr to a file in the specified format.
Export["file", exprs, elems]
    exports exprs to a file as elements specified by elems.
```

ExportString

WMA link

```
ExportString[expr, form]
    exports expr to a string, in the format form.
Export["file", exprs, elems]
    exports exprs to a string as elements specified by elems.
```

```
>> ExportString[{{1,2,3,4},{3},{2},{4}}, "CSV"]
1,2,3,4
 3,
 2,
 4,
>> ExportString[{1,2,3,4}, "CSV"]
1,
 2,
 3,
 4,
>> ExportString[Integrate[f[x],{x,0,2}], "SVG"]//Head
String
```

FileFormat

WMA link

```
FileFormat["name"]
attempts to determine what format Import should use to import specified file.
```

```
>> FileFormat["ExampleData/sunflowers.jpg"]
JPEG
>> FileFormat["ExampleData/EinsteinSzilLetter.txt"]
Text
>> FileFormat["ExampleData/hedy.tif"]
TIFF
```

Import

WMA link

```
Import["file"]
  imports data from a file.
Import["file", elements]
  imports the specified elements from a file.
Import["http://url", ...] and Import["ftp://url", ...]
  imports from a URL.

>> Import["ExampleData/ExampleData.txt", "Elements"]
{Data, Lines, Plaintext, String, Words}

>> Import["ExampleData/ExampleData.txt", "Lines"]
{Example File Format, Created by Angus, 0.629452
 0.586355, 0.711009 0.687453, 0.246540 0.433973, 0.926871
 0.887255, 0.825141 0.940900, 0.847035 0.127464, 0.054348
 0.296494, 0.838545 0.247025, 0.838697 0.436220, 0.309496 0.833591}

>> Import["ExampleData/colors.json"]
{colorsArray -> {{"colorName" -> black, "rgbValue" -> (0, 0,
 0), "hexValue" -> "#000000"}, {"colorName" -> red, "rgbValue" -> (255, 0,
0), "hexValue" -> "#FF0000"}, {"colorName" -> green, "rgbValue" -> (0, 255,
0), "hexValue" -> "#00FF00"}, {"colorName" -> blue, "rgbValue" -> (0, 0,
255), "hexValue" -> "#0000FF"}, {"colorName" -> yellow, "rgbValue" -> (255, 255,
0), "hexValue" -> "#FFFF00"}, {"colorName" -> cyan, "rgbValue" -> (0, 255,
255), "hexValue" -> "#00FFFF"}, {"colorName" -> magenta, "rgbValue" -> (255, 0,
255), "hexValue" -> "#FF00FF"}, {"colorName" -> white, "rgbValue" -> (255,
255, 255), "hexValue" -> "#FFFFFF"}}}
```

ImportString

WMA link

```
ImportString["data", "format"]
  imports data in the specified format from a string.
ImportString["file", elements]
  imports the specified elements from a string.
ImportString["data"]
  attempts to determine the format of the string from its content.
```

```
>> str = "Hello!\n This is a testing text\n";
>> ImportString[str, "Elements"]
{Data, Lines, Plaintext, String, Words}
>> ImportString[str, "Lines"]
{Hello!, This is a testing text}
```

RegisterExport

```
RegisterExport["format", func]
register func as the default function used when exporting from a file of type "format".
```

Simple text exporter

```
>> ExampleExporter1[filename_, data_, opts___] := Module[{strm =
  OpenWrite[filename], char = data}, WriteString[strm, char]; Close[
  strm]]

>> ImportExport`RegisterExport["ExampleFormat1", ExampleExporter1]

>> Export["sample.txt", "Encode this string!", "ExampleFormat1"];

>> FilePrint["sample.txt"]
Encode this string!
```

Very basic encrypted text exporter

```
>> ExampleExporter2[filename_, data_, opts___] := Module[{strm =
  OpenWrite[filename], char}, (* TODO: Check data *)char =
  FromCharacterCode[Mod[ToCharacterCode[data] - 84, 26] + 97];
  WriteString[strm, char]; Close[strm]]

>> ImportExport`RegisterExport["ExampleFormat2", ExampleExporter2]

>> Export["sample.txt", "encodethisstring", "ExampleFormat2"];

>> FilePrint["sample.txt"]
rapbqrqruvffgevat
```

RegisterImport

```
RegisterImport["format", defaultFunction]
register defaultFunction as the default function used when importing from a file of type "format".
RegisterImport["format", {"elem1" :> conditionalFunction1, "elem2" :> conditionalFunction2, ..., defaultFunction}]
registers multiple elements (elem1, ...) and their corresponding converter functions (conditionalFunction1, ...) in addition to the defaultFunction.
RegisterImport["format", {"conditionalFunctions", defaultFunction, "elem3" :> postFunction3, "elem4" :> postFunction4, ...}]
also registers additional elements (elem3, ...) whose converters (postFunction3, ...) act on output from the low-level functions.
```

First, define the default function used to import the data.

```
>> ExampleFormat1Import[filename_String] := Module[{stream, head, data},
  stream = OpenRead[filename]; head = ReadList[stream, String, 2];
  data = Partition[ReadList[stream, Number], 2]; Close[stream]; {""
  Header" -> head, "Data" -> data}]
```

RegisterImport is then used to register the above function to a new data format.

```
>> ImportExport`RegisterImport["ExampleFormat1", ExampleFormat1Import]
```

```

>> FilePrint["ExampleData/ExampleData.txt"]
Example File Format
Created by Angus
0.629452 0.586355
0.711009 0.687453
0.246540 0.433973
0.926871 0.887255
0.825141 0.940900
0.847035 0.127464
0.054348 0.296494
0.838545 0.247025
0.838697 0.436220
0.309496 0.833591

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat1", "Elements"}]
{Data, Header}

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat1", "Header"}]
{Example File Format, Created by Angus}

Conditional Importer:
>> ExampleFormat2DefaultImport[filename_String] := Module[{stream, head},
  stream = OpenRead[filename]; head = ReadList[stream, String, 2];
  Close[stream]; {"Header" -> head}]

>> ExampleFormat2DataImport[filename_String] := Module[{stream, data},
  stream = OpenRead[filename]; Skip[stream, String, 2]; data =
  Partition[ReadList[stream, Number], 2]; Close[stream]; {"Data" ->
  data}]

>> ImportExport`RegisterImport["ExampleFormat2", {"Data" :>
  ExampleFormat2DataImport, ExampleFormat2DefaultImport}]

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Elements"}]
{Data, Header}

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Header"}]
{Example File Format, Created by Angus}

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Data"}] //
Grid

 0.629452 0.586355
 0.711009 0.687453
 0.246540 0.433973
 0.926871 0.887255
 0.825141 0.9409
 0.847035 0.127464
 0.054348 0.296494
 0.838545 0.247025
 0.838697 0.43622
 0.309496 0.833591

```

URLFetch

WMA link

`URLFetch[URL]`

Returns the content of *URL* as a string.

=

...

27. Input and Output

Contents

\$Echo	318	Print	318
------------------	-----	-----------------	-----

\$Echo

WMA link

\$Echo

gives a list of files and pipes to which all input is echoed.

Print

WMA link

Print[*expr*, . . .]

prints each *expr* in string form.

```
>> Print["Hello world!"]
Hello world!

>> Print["The answer is ", 7 * 6, ".]
The answer is 42.
```

28. Input/Output, Files, and Filesystem

Contents

File and Stream	
Operations	319
Character	320
Close	320
EndOfFile	320
Expression	320
FilePrint	320
Find	321
Get (<>)	321
\$InputFileName	322
InputStream	322
\$Input	322
Number	322
OpenAppend	322
OpenRead	323
OpenWrite	323
OutputStream	323
Put (>>)	324
PutAppend (>>>)	324
Read	326
ReadList	326
Record	326
SetStreamPosition	327
Skip	327
StreamPosition	327
Streams	328
StringToStream	328
Word	328
Write	329
WriteString	329
Filesystem Operations	329
AbsoluteFileName	330
\$BaseDirectory	330
CopyDirectory	330
CopyFile	330
CreateDirectory	331
CreateFile	331
CreateTemporary	331
DeleteDirectory	331
DeleteFile	331
Directory	332
DirectoryName	332
DirectoryQ	332
DirectoryStack	332
ExpandFileName	333
File	333
FileBaseName	333
FileByteCount	333
FileDate	334
FileExistsQ	334
FileExtension	334
FileHash	335
FileInformation	335
FileNameDepth	335
FileNameJoin	335
FileNameSplit	336
FileNameTake	336
FileNames	336
FileType	337
FindFile	337
FindList	337
Hash	338
\$HomeDirectory	338
\$InitialDirectory	338
\$InstallationDirectory	338
Needs	339
\$OperatingSystem	339
ParentDirectory	339
\$Path	339
\$PathnameSeparator	339
RenameDirectory	340
RenameFile	340
ResetDirectory	340
\$RootDirectory	340
SetDirectory	341
SetFileDate	341
\$TemporaryDirectory	341
ToFileName	341
URLSave	342
\$UserBaseDirectory	342
Importing and Exporting	342
B64Decode	342
B64Encode	342
\$ExtensionMappings	343
\$FormatMappings	343
Export	343
\$ExportFormats	343
ExportString	344
FileFormat	344
Import	345
\$ImportFormats	345
ImportString	345
RegisterExport	346
RegisterImport	347
URLFetch	347

File and Stream Operations

File and Stream Operation

Character

WMA link

Character
is a data type for Read.

Close

WMA link

Close [*stream*]
closes an input or output stream.

```
>> Close[StringToStream["123abc"]]  
String  
>> file=Close[OpenWrite[]]  
/tmp/tmpkj3wsqyo
```

Closing a file doesn't delete it from the filesystem

```
>> DeleteFile[file];
```

EndOfFile

WMA link

EndOfFile
is returned by Read when the end of an input stream is reached.

Expression

WMA link

Expression
is a data type for Read.

For information about underlying data structure Expression (a kind of M-expression) that is central in evaluation, see: AST, M-Expression, General List same thing.

FilePrint

WMA link

FilePrint [*file*]
prints the raw contents of *file*.

Find

WMA link

Find[*stream*, *text*]

find the first line in *stream* that contains *text*.

```
>> stream = OpenRead["ExampleData/EinsteinSzilLetter.txt",
   CharacterEncoding->"UTF8"];

>> Find[stream, "uranium"]

>> Find[stream, "uranium"]

>> Close[stream]

>> stream = OpenRead["ExampleData/EinsteinSzilLetter.txt",
   CharacterEncoding->"UTF8"];

>> Find[stream, {"energy", "power"}]

>> Find[stream, {"energy", "power"}]

>> Close[stream]
```

Get (<<)

WMA link

<<name

reads a file and evaluates each expression, returning only the last one.

Get[*name*, Trace->True]

Runs Get tracing each line before it is evaluated.

```
>> filename = $TemporaryDirectory <> "/example_file";

>> Put[x + y, filename]

>> Get[filename]

>> filename = $TemporaryDirectory <> "/example_file";

>> Put[x + y, 2x^2 + 4z!, Cos[x] + I Sin[x], filename]

>> Get[filename]

>> DeleteFile[filename]
```

\$InputFileName

WMA link

\$InputFileName

is the name of the file from which input is currently being read.

While in interactive mode, \$InputFileName is "".

```
>> $InputFileName
```

InputStream

WMA link

```
InputStream[name, n]
represents an input stream for functions such as Read or Find.
```

StringToStream opens an input stream:

```
>> stream = StringToStream["Mathics is cool!"]
InputStream [String, 27]

>> Close[stream]
String
```

\$Input

WMA link

```
$Input
is the name of the stream from which input is currently being read.
```

```
>> $Input
```

Number

WMA link

```
Number
is a data type for Read.
```

OpenAppend

WMA link

```
OpenAppend['file']
opens a file and returns an OutputStream to which writes are appended.
```

```
>> OpenAppend[]
OutputStream [/tmp/tmpcnh6fney, 27]
```

OpenRead

WMA link

```
OpenRead['file']
opens a file and returns an InputStream.
```

```
>> OpenRead["ExampleData/EinsteinSzilLetter.txt", CharacterEncoding -> "UTF8"]
InputStream [ExampleData/EinsteinSzilLetter.txt, 28]
```

```
>> Close[OpenRead["https://raw.githubusercontent.com/Mathics3/mathics-
core/master/README.rst"]];
```

OpenWrite

WMA link

```
OpenWrite[“file”]
opens a file and returns an OutputStream.
```

```
>> OpenWrite[]
OutputStream [/tmp/tmpcqq_7ips,29]
```

OutputStream

WMA link

```
OutputStream[name, n]
represents an output stream.
```

By default, the list of Streams normally OutputStream entries for stderr and stdout

```
>> Streams []
{InputStream [stdin, 0], OutputStream [stdout, 1], OutputStream [
stderr, 2], OutputStream [/tmp/tmp0hi3hmgy, 3], InputStream [
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 4], OutputStream [
/tmp/tmp46rnxgfg, 5], InputStream [String, 6], InputStream [
String, 7], InputStream [String, 8], InputStream [String, 9], InputStream [
String, 10], InputStream [String, 11], InputStream [String, 12], InputStream [
String, 13], InputStream [String, 14], InputStream [String, 15], InputStream [
String, 16], InputStream [String, 17], InputStream [String, 18], InputStream [
String, 19], InputStream [String, 20], InputStream [String, 21], InputStream [
String, 22], InputStream [String, 23], OutputStream [
/tmp/tmp4wt98oxv, 24], InputStream [String, 25], InputStream [
/tmp/tmppyn8jk0, 26], OutputStream [/tmp/tmpcnh6fney, 27], InputStream [
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 28], OutputStream [
/tmp/tmpcqq_7ips, 29]}
```

Put (>>)

WMA link

```
expr >> filename
write expr to a file.
Put [expr1, expr2, ..., filename]
write a sequence of expressions to a file.
```

```
>> Put[40!, fortyfactorial]
fortyfactorial is not string, InputStream[], or OutputStream[]
815915283247897734345611269596115894272000000000>fortyfactorial
```

```

>> filename = $TemporaryDirectory <> "/fortyfactorial";
>> Put[40!, filename]
>> FilePrint[filename]
>> Get[filename]
815915283247897734345611269596115894272000000000
>> DeleteFile[filename]

>> filename = $TemporaryDirectory <> "/fiftyfactorial";
>> Put[10!, 20!, 30!, filename]
>> FilePrint[filename]
>> DeleteFile[filename]

=
>> filename = $TemporaryDirectory <> "/example_file";
>> Put[x + y, 2x^2 + 4z!, Cos[x] + I Sin[x], filename]
>> FilePrint[filename]
>> DeleteFile[filename]

```

PutAppend (>>>)

WMA link

```

expr >>> filename
append expr to a file.
PutAppend[expr1, expr2, ..., $``filename'`]
write a sequence of expressions to a file.

```

```

>> Put[50!, "factorials"]

>> FilePrint["factorials"]

>> PutAppend[10!, 20!, 30!, "factorials"]

>> FilePrint["factorials"]

>> 60! >>> "factorials"

>> FilePrint["factorials"]

>> "string" >>> factorials

>> FilePrint["factorials"]

```

Read

WMA link

```
Read[stream]
  reads the input stream and returns one expression.
Read[stream, type]
  reads the input stream and returns an object of the given type.
Read[stream, type]
  reads the input stream and returns an object of the given type. type
Read[stream, Hold[Expression]]
  reads the input stream for an Expression and puts it inside Hold.
```

is one of:

- Byte
- Character
- Expression
- HoldExpression
- Number
- Real
- Record
- String
- Word

```
>> stream = StringToStream["abc123"];

>> Read[stream, String]
abc123

>> stream = StringToStream["abc 123"];

>> Read[stream, Word]

>> Read[stream, Word]

>> stream = StringToStream["123, 4"];

>> Read[stream, Number]

>> Read[stream, Number]

>> stream = StringToStream["2+2\n2+3"];
```

Read with a Hold[Expression] returns the expression it reads unevaluated so it can be later inspected and evaluated:

```
>> Read[stream, Hold[Expression]]

>> Read[stream, Expression]
5

>> Close[stream];
```

Reading a comment however will return the empty list:

```
>> stream = StringToStream["(* ::Package:: *)"];

>> Read[stream, Hold[Expression]]
```

```

>> Close[stream];
>> stream = StringToStream["123 abc"];
>> Read[stream, {Number, Word}]
{123,abc}

```

Multiple lines:

```

>> stream = StringToStream["\"Tengo una\nvaca lechera.\""]; Read[stream]
Tengo una
vaca lechera.

```

ReadList

WMA link

<pre>ReadList["file"]</pre>	Reads all the expressions until the end of file.
<pre>ReadList["file", type]</pre>	Reads objects of a specified type until the end of file.
<pre>ReadList["file", {type1, type2, ...}]</pre>	Reads a sequence of specified types until the end of file.

```

>> ReadList[StringToStream["a 1 b 2"], {Word, Number}]
{{a,1},{b,2}}
>> stream = StringToStream["\"abc123\"]";
>> ReadList[stream]
{abc123}
>> InputForm[%]
{"abc123"}

```

Record

WMA link

<pre>Record</pre>	is a data type for Read.
-------------------	--------------------------

SetStreamPosition

WMA link

<pre>SetStreamPosition[stream, n]</pre>	sets the current position in a stream.
---	--

```

>> stream = StringToStream["Mathics is cool!"]
InputStream [String,43]

```

```
>> SetStreamPosition[stream, 8]
8
>> Read[stream, Word]
is
>> SetStreamPosition[stream, Infinity]
16
```

Skip

WMA link

```
Skip[stream, type]
    skips ahead in an input stream by one object of the specified type.
Skip[stream, type, n]
    skips ahead in an input stream by n objects of the specified type.
```

```
>> stream = StringToStream["a b c d"];
>> Read[stream, Word]
>> Skip[stream, Word]
>> Read[stream, Word]
>> stream = StringToStream["a b c d"];
>> Read[stream, Word]
>> Skip[stream, Word, 2]
>> Read[stream, Word]
```

StreamPosition

WMA link

```
StreamPosition[stream]
    returns the current position in a stream as an integer.
```

```
>> stream = StringToStream["Mathics is cool!"]
InputStream [String, 47]
>> Read[stream, Word]
Mathics
>> StreamPosition[stream]
7
```

Streams

WMA link

Streams []

returns a list of all open streams.

```
>> Streams []
{InputStream [stdin, 0], OutputStream [stdout, 1], OutputStream [
stderr, 2], OutputStream [/tmp/tmp0hi3hmgy, 3], InputStream [
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 4], OutputStream [
/tmp/tmp46rnxgfg, 5], InputStream [String, 6], InputStream [
String, 7], InputStream [String, 8], InputStream [String, 9], InputStream [
String, 10], InputStream [String, 11], InputStream [String, 12], InputStream [
String, 13], InputStream [String, 14], InputStream [String, 15], InputStream [
String, 16], InputStream [String, 17], InputStream [String, 18], InputStream [
String, 19], InputStream [String, 20], InputStream [String, 21], InputStream [
String, 22], InputStream [String, 23], OutputStream [
/tmp/tmp4wt98oxv, 24], InputStream [String, 25], InputStream [
/tmp/tmpbyn8jvk0, 26], OutputStream [/tmp/tmpcnh6fney, 27], InputStream [
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/EinsteinSzilLetter.txt, 28], OutputStream [
/tmp/tmpcq7ips, 29], InputStream [String, 30], InputStream [
String, 31], InputStream [String, 32], InputStream [String, 33], InputStream [
String, 34], InputStream [String, 35], InputStream [String, 36], InputStream [
String, 37], InputStream [String, 38], InputStream [String, 39], InputStream [
String, 40], InputStream [String, 41], InputStream [String, 42], InputStream [
String, 43], InputStream [String, 44], InputStream [String, 45], InputStream [
String, 46], InputStream [String, 47], OutputStream [/tmp/tmpo9yseybx, 48]}
```

```
>> Streams ["stdout"]
{OutputStream [stdout, 1]}
```

StringToStream

WMA link

StringToStream [string]

converts a *string* to an open input stream.

```
>> strm = StringToStream["abc 123"]
InputStream [String, 49]
```

Word

WMA link

Word

is a data type for Read.

Write

WMA link

```
Write[channel, expr1, expr2, ...]  
writes the expressions to the output channel followed by a newline.
```

```
>> stream = OpenWrite[]  
OutputStream [/tmp/tmplg8xzw2o, 50]  
>> Write[stream, 10 x + 15 y ^ 2]  
>> Write[stream, 3 Sin[z]]  
>> Close[stream];  
>> stream = OpenRead[%];  
>> ReadList[stream]  
{10x + 15y2, 3Sin[z]}
```

WriteString

WMA link

```
WriteString[stream, $str1, str2, ... ]  
writes the strings to the output stream.
```

```
>> stream = OpenWrite[];  
>> WriteString[stream, "This is a test 1"]  
>> WriteString[stream, "This is also a test 2"]  
>> pathname = Close[stream];  
>> FilePrint[%]  
>> stream = OpenWrite[];  
>> WriteString[stream, "This is a test 1", "This is also a test 2"]  
>> pathname = Close[stream]  
/tmp/tmpkhjswj9z  
>> FilePrint[%]
```

If stream is the string “stdout” or “stderr”, writes to the system standard output/ standard error channel:

```
>> WriteString["stdout", "Hola"]
```

Filesystem Operations

Filesystem Operation

AbsoluteFileName

WMA link

```
AbsoluteFileName["name"]
    returns the absolute version of the given filename.
```

```
>> AbsoluteFileName["ExampleData/sunflowers.jpg"]
    /src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/sunflowers.jpg
```

\$BaseDirectory

WMA link

```
$BaseDirectory
    returns the folder where user configurations are stored.
```

```
>> $BaseDirectory
    /src/external-vcs/github/Mathics3/mathics-core/mathics
```

CopyDirectory

WMA link

```
CopyDirectory["dir1", "dir2"]
    copies directory dir1 to dir2.
```

CopyFile

WMA link

```
CopyFile["file1", "file2"]
    copies file1 to file2.
```

```
>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers.jpg"]
    MathicsSunflowers.jpg
>> DeleteFile["MathicsSunflowers.jpg"]
```

CreateDirectory

WMA link

```
CreateDirectory["dir"]
    creates a directory called dir.
CreateDirectory[]
    creates a temporary directory.
```

```
>> dir = CreateDirectory[]
    /tmp/mjh79wa2y
```

CreateFile

WMA link

```
CreateFile[“filename”]  
Creates a file named “filename” temporary file, but do not open it.  
CreateFile[]  
Creates a temporary file, but do not open it.
```

CreateTemporary

WMA link

```
CreateTemporary[]  
Creates a temporary file, but do not open it.
```

DeleteDirectory

WMA link

```
DeleteDirectory[“dir”]  
deletes a directory called dir.
```

```
>> dir = CreateDirectory[]  
/tmp/mfvzv56n0t  
>> DeleteDirectory[dir]  
>> DirectoryQ[dir]  
False
```

DeleteFile

WMA link

```
Delete[“file”]  
deletes file.  
Delete[{“file1”, “file2”, ...}]  
deletes a list of files.
```

```
>> CopyFile[“ExampleData/sunflowers.jpg”, “MathicsSunflowers.jpg”];  
>> DeleteFile[“MathicsSunflowers.jpg”]  
>> CopyFile[“ExampleData/sunflowers.jpg”, “MathicsSunflowers1.jpg”];  
>> CopyFile[“ExampleData/sunflowers.jpg”, “MathicsSunflowers2.jpg”];  
>> DeleteFile[{"MathicsSunflowers1.jpg", "MathicsSunflowers2.jpg"}]
```

Directory

WMA link

```
Directory[]  
    returns the current working directory.
```

```
>> Directory[]  
/home/rocky
```

DirectoryName

WMA link

```
DirectoryName["name"]  
    extracts the directory name from a filename.
```

```
>>DirectoryName["a/b/c"]  
a/b  
>>DirectoryName["a/b/c", 2]  
a
```

DirectoryQ

WMA link

```
DirectoryQ["name"]  
    returns True if the directory called name exists and False otherwise.
```

```
>>DirectoryQ["ExampleData/"]  
True  
>>DirectoryQ["ExampleData/MythicalSubdir/"]  
False
```

DirectoryStack

WMA link

```
DirectoryStack[]  
    returns the directory stack.
```

```
>>DirectoryStack[]  
{/src/external-vcs/github/Mathics3/mathics-core/mathics, /home/rocky, /home/rocky}
```

ExpandFileName

WMA link

```
ExpandFileName["name"]  
    expands name to an absolute filename for your system.
```

```
>> ExpandFileName["ExampleData/sunflowers.jpg"]
/home/rocky/ExampleData/sunflowers.jpg
```

File

WMA link

`File["file"]`
is a symbolic representation of an element in the local file system.

FileBaseName

WMA link

`FileBaseName["file"]`
gives the base name for the specified file name.

```
>> FileBaseName["file.txt"]
file
>> FileBaseName["file.tar.gz"]
file.tar
```

FileByteCount

WMA link

`FileByteCount[file]`
returns the number of bytes in *file*.

```
>> FileByteCount["ExampleData/sunflowers.jpg"]
142286
```

FileDate

WMA link

`FileDate[file, types]`
returns the time and date at which the file was last modified.

```
>> FileDate["ExampleData/sunflowers.jpg"]
{2123, 1, 16, 3, 50, 39.9396}
>> FileDate["ExampleData/sunflowers.jpg", "Access"]
{2123, 2, 26, 13, 19, 30.2059}
>> FileDate["ExampleData/sunflowers.jpg", "Creation"]
Missing [NotApplicable]
>> FileDate["ExampleData/sunflowers.jpg", "Change"]
{2123, 1, 16, 3, 50, 39.9396}
```

```

>> FileDate["ExampleData/sunflowers.jpg", "Modification"]
{2123, 1, 16, 3, 50, 39.9396}

>> FileDate["ExampleData/sunflowers.jpg", "Rules"]
{Access -> {2123, 2, 26, 13, 19, 30.2059}, Creation -> Missing [
  NotApplicable], Change -> {2123, 1, 16, 3, 50, 39.9396}, Modification
 -> {2123, 1, 16, 3, 50, 39.9396}}

```

FileExistsQ

WMA link

```
FileExistsQ["file"]
  returns True if file exists and False otherwise.
```

```

>> FileExistsQ["ExampleData/sunflowers.jpg"]
True

>> FileExistsQ["ExampleData/sunflowers.png"]
False

```

FileExtension

WMA link

```
FileExtension["file"]
  gives the extension for the specified file name.
```

```

>> FileExtension["file.txt"]
txt

>> FileExtension["file.tar.gz"]
gz

```

FileHash

WMA link

```
FileHash[file]
  returns an integer hash for the given file.
FileHash[file, type]
  returns an integer hash of the specified type for the given file.
  The types supported are "MD5", "Adler32", "CRC32", "SHA", "SHA224", "SHA256",
  "SHA384", and "SHA512".
FileHash[file, type, format]
  gives a hash code in the specified format.
```

```

>> FileHash["ExampleData/sunflowers.jpg"]
109937059621979839952736809235486742106

>> FileHash["ExampleData/sunflowers.jpg", "MD5"]
109937059621979839952736809235486742106

```

```

>> FileHash["ExampleData/sunflowers.jpg", "Adler32"]
1607049478
>> FileHash["ExampleData/sunflowers.jpg", "SHA256"]
111619807552579450300684600241129773909359865098672286468229443390003894913065

```

FileInformation

WMA link

```
FileInformation["file"]
  returns information about file.
```

This function is totally undocumented in MMA!

```

>> FileInformation["ExampleData/sunflowers.jpg"]
{File->/home/rocky/ExampleData/sunflowers.jpg, FileType
 -> File, ByteCount->142286, Date->7.0385*^9}

```

FileNameDepth

WMA link

```
FileNameDepth["name"]
  gives the number of path parts in the given filename.
```

```

>> FileNameDepth["a/b/c"]
3
>> FileNameDepth["a/b/c/"]
3

```

FileNameJoin

WMA link

```
FileNameJoin[{"dir_1", "dir_2", ...}]
  joins the dir_i together into one path.
FileNameJoin[..., OperatingSystem->{"os"}]
  yields a file name in the format for the specified operating system. Possible choices are "Windows", "MacOSX", and "Unix".
```

```

>> FileNameJoin[{"dir1", "dir2", "dir3"}]
dir1/dir2/dir3
>> FileNameJoin[{"dir1", "dir2", "dir3"}, OperatingSystem -> "Unix"]
dir1/dir2/dir3
>> FileNameJoin[{"dir1", "dir2", "dir3"}, OperatingSystem -> "Windows"]
dir1\dir2\dir3

```

FileNameSplit

WMA link

```
FileNameSplit["filenames"]
splits a filename into a list of parts.
```

```
>> FileNameSplit["example/path/file.txt"]
{example, path, file.txt}
```

FileNameTake

WMA link

```
FileNameTake["file"]
returns the last path element in the file name name.
FileNameTake["file", n]
returns the first n path elements in the file name name.
FileNameTake["file", $-n$]
returns the last n path elements in the file name name.
```

FileNames

WMA link

```
FileNames[]
Returns a list with the filenames in the current working folder.
FileNames[form]
Returns a list with the filenames in the current working folder that matches with form.
FileNames[{form_1, form_2, ...}]
Returns a list with the filenames in the current working folder that matches with one of form_1, form_2, ...
FileNames[{form_1, form_2, ...}, {dir_1, dir_2, ...}]
Looks into the directories dir_1, dir_2, ...
FileNames[{form_1, form_2, ...}, {dir_1, dir_2, ...}]
Looks into the directories dir_1, dir_2, ...
FileNames[{forms, dirs, n}
Look for files up to the level n.
```

```
>> SetDirectory[$InstallationDirectory <> "/autoload"];
>> FileNames["*.m", "formats"]//Length
0
>> FileNames["*.m", "formats", 3]//Length
14
>> FileNames["*.m", "formats", Infinity]//Length
14
```

FileType

WMA link

```
FileType["file"]  
gives the type of a file, a string. This is typically File, Directory or None.
```

```
>> FileType["ExampleData/sunflowers.jpg"]  
File  
>> FileType["ExampleData"]  
Directory  
>> FileType["ExampleData/nonexistent"]  
None
```

FindFile

WMA link

```
FindFile[name]  
searches $Path for the given filename.
```

```
>> FindFile["ExampleData/sunflowers.jpg"]  
/src/external-vcs/github/Mathics3/mathics-core/mathics/data/ExampleData/sunflowers.jpg  
>> FindFile["VectorAnalysis`"]  
/src/external-vcs/github/Mathics3/mathics-core/mathics/packages/VectorAnalysis/Kernel/init.m  
>> FindFile["VectorAnalysis`VectorAnalysis`"]  
/src/external-vcs/github/Mathics3/mathics-core/mathics/packages/VectorAnalysis/VectorAnalysis.m
```

FindList

WMA link

```
FindList[file, text]  
returns a list of all lines in file that contain text.  
FindList[file, {text1, text2, ...}]  
returns a list of all lines in file that contain any of the specified string.  
FindList[{file1, file2, ...}, ...]  
returns a list of all lines in any of the filei that contain the specified strings.
```

```
>> stream = FindList["ExampleData/EinsteinSzilLetter.txt", "uranium"];  
>> FindList["ExampleData/EinsteinSzilLetter.txt", "uranium", 1]  
{in manuscript, leads me to expect that the element uranium may be turned into}
```

Hash

Hash function (WMA link)

Hash [expr]

returns an integer hash for the given *expr*.

Hash [expr, type]

returns an integer hash of the specified *type* for the given *expr*.

The types supported are "MD5", "Adler32", "CRC32", "SHA", "SHA224", "SHA256", "SHA384", and "SHA512".

Hash [expr, type, format]

Returns the hash in the specified format.

```
> Hash["The Adventures of Huckleberry Finn"] = 213425047836523694663619736686226550816
```

```
> Hash["The Adventures of Huckleberry Finn", "SHA256"] = 950926495945903842880571834086092549189343518116698
```

```
> Hash[1/3] = 56073172797010645108327809727054836008
```

```
> Hash[{a, b, {c, {d, e, f}}}] = 135682164776235407777080772547528225284
```

```
> Hash[SomeHead[3.1415]] = 58042316473471877315442015469706095084
```

```
>> Hash[{a, b, c}, "xyzstr"]
```

```
Hash [{a, b, c}, xyzstr, Integer]
```

\$HomeDirectory

WMA link

\$HomeDirectory

returns the users HOME directory.

```
>> $HomeDirectory
```

```
/home/rocky
```

\$InitialDirectory

WMA link

\$InitialDirectory

returns the directory from which *Mathics3* was started.

```
>> $InitialDirectory
```

```
/src/external-vcs/github/Mathics3/mathics-core/mathics
```

\$InstallationDirectory

WMA link

\$InstallationDirectory

returns the top-level directory in which *Mathics3* was installed.

```
>> $InstallationDirectory
```

```
/src/external-vcs/github/Mathics3/mathics-core/mathics
```

Needs

WMA link

```
Needs["context"]
  loads the specified context if not already in $Packages.
```

```
>> Needs["VectorAnalysis"]
```

\$OperatingSystem

WMA link

```
$OperatingSystem
  gives the type of operating system running Mathics.
```

```
>> $OperatingSystem
  Unix
```

ParentDirectory

WMA link

```
ParentDirectory[]
  returns the parent of the current working directory.
ParentDirectory["dir"]
  returns the parent dir.
```

```
>> ParentDirectory[]
  /src/external-vcs/github/Mathics3/mathics-core/mathics
```

\$Path

WMA link

```
$Path
  returns the list of directories to search when looking for a file.
```

```
>> $Path
  {., /home/rocky, /src/external-vcs/github/Mathics3/mathics-core/mathics/data, /src/external-vcs/github/Ma}
```

\$PathnameSeparator

WMA link

```
$PathnameSeparator
  returns a string for the separator in paths.
```

```
>> $PathnameSeparator
  /
```

RenameDirectory

WMA link

```
RenameDirectory["dir1` `,"dir2"]
renames directory dir1 to dir2.
```

RenameFile

WMA link

```
RenameFile["file1` `,"file2"]
renames file1 to file2.
```

```
>> CopyFile["ExampleData/sunflowers.jpg", "MathicsSunflowers.jpg"]
MathicsSunflowers.jpg

>> RenameFile["MathicsSunflowers.jpg", "MathicsSunnyFlowers.jpg"]
MathicsSunnyFlowers.jpg

>> DeleteFile["MathicsSunnyFlowers.jpg"]
```

ResetDirectory

WMA link

```
ResetDirectory[]
pops a directory from the directory stack and returns it.
```

```
>> ResetDirectory[]
/src/external-vcs/github/Mathics3/mathics-core/mathics/autoload
```

\$RootDirectory

WMA link

```
$RootDirectory
returns the system root directory.
```

```
>> $RootDirectory
/
```

SetDirectory

WMA link

```
SetDirectory[dir]
sets the current working directory to dir.
```

```
>> SetDirectory[]
/home/rocky
```

SetFileName

WMA link

```
SetFileName["file"]
    set the file access and modification dates of file to the current date.
SetFileName["file", date]
    set the file access and modification dates of file to the specified date list.
SetFileName["file", date, "type"]
    set the file date of file to the specified date list. The "type" can be one of "Access", "Creation",
    "Modification", or All.
```

Create a temporary file (for example purposes)

```
>> tmpfilename = $TemporaryDirectory <> "/tmp0";
>> Close[OpenWrite[tmpfilename]];
>> SetFileName[tmpfilename, {2002, 1, 1, 0, 0, 0.}, "Access"];
>> FileDate[tmpfilename, "Access"]
{2002, 1, 1, 0, 0, 0.}
```

\$TemporaryDirectory

WMA link

```
$TemporaryDirectory
    returns the directory used for temporary files.
```

```
>> $TemporaryDirectory
/tmp
```

ToFileName

WMA link

```
ToFileName[{"dir_1", "dir_2", ...}]
    joins the diri together into one path.
```

ToFileName has been superseded by FileNameJoin.

```
>> ToFileName[{"dir1", "dir2"}, "file"]
dir1/dir2/file

>> ToFileName["dir1", "file"]
dir1/file

>> ToFileName[{"dir1", "dir2", "dir3"}]
dir1/dir2/dir3
```

URLSave

WMA link

```
URLSave[{"url"}]
Save "url" in a temporary file.
URLSave[{"url", "filename"}]
Save "url" in filename.
```

\$UserBaseDirectory

WMA link

```
$UserBaseDirectory
returns the folder where user configurations are stored.
```

```
>> $RootDirectory
/
```

Importing and Exporting

Importing and Exporting

Many kinds data formats can be read into *Mathics3*. Variable \$ExportFormats of section 26 contains a list of file formats that are supported by Export of section 26, while \$InputFormats of section ?? does the corresponding thing for Import of section 26.

B64Decode

WMA link

```
System`Convert`B64Dump`B64Decode[string]
Decode string in Base64 coding to an expression.
```

```
>> System`Convert`B64Dump`B64Decode["R!="]
String "R!=" is not a valid b64 encoded string.
$Failed
```

B64Encode

WMA link

```
System`Convert`B64Dump`B64Encode[expr]
Encodes expr in Base64 coding
```

```
>> System`Convert`B64Dump`B64Encode["Hello world"]
SGVsbG8gd29ybGQ=
>> System`Convert`B64Dump`B64Decode[%]
>> System`Convert`B64Dump`B64Encode[Integrate[f[x],{x,0,2}]]
SW50ZWdyYXRlW2ZbeF0sIHt4LCAwLCAYfV0=
>> System`Convert`B64Dump`B64Decode[%]
```

\$ExtensionMappings

```
System`ConvertersDump`$ExtensionMappings
    Returns a list of associations between file extensions and file types.
```

The format associated to the extension ".jpg"

```
>> ".jpg"/. System`ConvertersDump`$ExtensionMappings
    JPEG
```

\$FormatMappings

```
System`ConverterDump$FormatMappings
    Returns a list of associations between file extensions and file types.
```

The list of MIME types associated to the extension JPEG:

```
>> Select[System`ConvertersDump`$FormatMappings, (#1[[2]]=="JPEG")&] [[All
    , 1]]
    {APPLICATION/JPG, APPLICATION/X-JPG, IMAGE/JPEG, IMAGE/JPG, IMAGE/PJPEG, JPEG, JPG}
```

Export

WMA link

```
Export["file.ext", expr]
    exports expr to a file, using the extension ext to determine the format.
Export["file", expr, "format"]
    exports expr to a file in the specified format.
Export["file", exprs, elems]
    exports exprs to a file as elements specified by elems.
```

\$ExportFormats

WMA link

```
$ExportFormats
    returns a list of file formats supported by Export.
```

```
>> $ExportFormats
    {BMP, Base64, CSV, ExampleFormat1, ExampleFormat2, GIF, JPEG, JPEG2000, PBM, PCX, PGM, PNG, PPM, SVG, T}
```

ExportString

WMA link

```
ExportString[expr, form]
    exports expr to a string, in the format form.
Export["file", exprs, elems]
    exports exprs to a string as elements specified by elems.
```

```

>> ExportString[{{1,2,3,4},{3},{2},{4}}, "CSV"]
1,2,3,4
3,
2,
4,
>> ExportString[{1,2,3,4}, "CSV"]
1,
2,
3,
4,
>> ExportString[Integrate[f[x],{x,0,2}], "SVG"]//Head
String

```

FileFormat

WMA link

```

FileFormat["name"]
attempts to determine what format Import should use to import specified file.

```

```

>> FileFormat["ExampleData/sunflowers.jpg"]
JPEG

>> FileFormat["ExampleData/EinsteinSzilLetter.txt"]
Text

>> FileFormat["ExampleData/hedy.tif"]
TIFF

```

Import

WMA link

```

Import["file"]
imports data from a file.
Import["file", elements]
imports the specified elements from a file.
Import["http://url", ...] and Import["ftp://url", ...]
imports from a URL.

```

```

>> Import["ExampleData/ExampleData.txt", "Elements"]
{Data, Lines, Plaintext, String, Words}

>> Import["ExampleData/ExampleData.txt", "Lines"]
{Example File Format, Created by Angus, 0.629452
 0.586355, 0.711009 0.687453, 0.246540 0.433973, 0.926871
 0.887255, 0.825141 0.940900, 0.847035 0.127464, 0.054348
 0.296494, 0.838545 0.247025, 0.838697 0.436220, 0.309496 0.833591}

```

```

>> Import["ExampleData/colors.json"]
{colorsArray -> {{colorName -> black, rgbValue -> (0, 0,
0), hexValue -> "#000000"}, {colorName -> red, rgbValue -> (255, 0,
0), hexValue -> "#FF0000"}, {colorName -> green, rgbValue -> (0, 255,
0), hexValue -> "#00FF00"}, {colorName -> blue, rgbValue -> (0, 0,
255), hexValue -> "#0000FF"}, {colorName -> yellow, rgbValue -> (255, 255,
0), hexValue -> "#FFFF00"}, {colorName -> cyan, rgbValue -> (0, 255,
255), hexValue -> "#00FFFF"}, {colorName -> magenta, rgbValue -> (255, 0,
255), hexValue -> "#FF00FF"}, {colorName -> white, rgbValue -> (255,
255, 255), hexValue -> "#FFFFFF"}}}

```

\$ImportFormats

WMA link

```

$ImportFormats
    returns a list of file formats supported by Import.

```

```

>> $ImportFormats
{BMP, Base64, CSV, ExampleFormat1, ExampleFormat2, GIF, HTML, ICO, JPEG, JPEG2000, JSON, PBM, PCX, PGM}

```

ImportString

WMA link

```

ImportString["data", "format"]
    imports data in the specified format from a string.
ImportString["file", "elements"]
    imports the specified elements from a string.
ImportString["data"]
    attempts to determine the format of the string from its content.

```

```

>> str = "Hello!\n This is a testing text\n";
>> ImportString[str, "Elements"]
{Data, Lines, Plaintext, String, Words}
>> ImportString[str, "Lines"]
{Hello!, This is a testing text}

```

RegisterExport

```

RegisterExport["format", func]
    register func as the default function used when exporting from a file of type "format".

```

Simple text exporter

```

>> ExampleExporter1[filename_, data_, opts___] := Module[{strm =
OpenWrite[filename], char = data}, WriteString[strm, char]; Close[
strm]]

```

```

>> ImportExport`RegisterExport["ExampleFormat1", ExampleExporter1]
>> Export["sample.txt", "Encode this string!", "ExampleFormat1"];
>> FilePrint["sample.txt"]

```

Very basic encrypted text exporter

```

>> ExampleExporter2[filename_, data_, opts___] := Module[{strm =
  OpenWrite[filename], char}, (* TODO: Check data *)char =
  FromCharacterCode[Mod[ToCharacterCode[data] - 84, 26] + 97];
  WriteString[strm, char]; Close[strm]]
>> ImportExport`RegisterExport["ExampleFormat2", ExampleExporter2]
>> Export["sample.txt", "encodethisstring", "ExampleFormat2"];
>> FilePrint["sample.txt"]

```

RegisterImport

```

RegisterImport["format", defaultFunction]
  register defaultFunction as the default function used when importing from a file of type "format".
RegisterImport["format", {"elem1" :> conditionalFunction1, "elem2" :> conditionalFunction2, ..., defaultFunction}]
  registers multiple elements (elem1, ...) and their corresponding converter functions (conditionalFunction1, ...) in addition to the defaultFunction.
RegisterImport["format", {"conditionalFunctions", defaultFunction, "elem3" :> postFunction3, "elem4" :> postFunction4, ...}]
  also registers additional elements (elem3, ...) whose converters (postFunction3, ...) act on output from the low-level functions.

```

First, define the default function used to import the data.

```

>> ExampleFormat1Import[filename_String] := Module[{stream, head, data},
  stream = OpenRead[filename]; head = ReadList[stream, String, 2];
  data = Partition[ReadList[stream, Number], 2]; Close[stream]; {"Header" -> head, "Data" -> data}]

```

RegisterImport is then used to register the above function to a new data format.

```

>> ImportExport`RegisterImport["ExampleFormat1", ExampleFormat1Import]
>> FilePrint["ExampleData/ExampleData.txt"]
Example File Format
Created by Angus
0.629452 0.586355
0.711009 0.687453
0.246540 0.433973
0.926871 0.887255
0.825141 0.940900
0.847035 0.127464
0.054348 0.296494
0.838545 0.247025
0.838697 0.436220
0.309496 0.833591
>> Import["ExampleData/ExampleData.txt", {"ExampleFormat1", "Elements"}]
{Data,Header}

```

```

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat1", "Header"}]
{Example File Format, Created by Angus}

Conditional Importer:
>> ExampleFormat2DefaultImport[filename_String] := Module[{stream, head},
  stream = OpenRead[filename]; head = ReadList[stream, String, 2];
  Close[stream]; {"Header" -> head}]

>> ExampleFormat2DataImport[filename_String] := Module[{stream, data},
  stream = OpenRead[filename]; Skip[stream, String, 2]; data =
  Partition[ReadList[stream, Number], 2]; Close[stream]; {"Data" ->
  data}]

>> ImportExport`RegisterImport["ExampleFormat2", {"Data" :>
  ExampleFormat2DataImport, ExampleFormat2DefaultImport}]

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Elements"}]
{Data, Header}

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Header"}]
{Example File Format, Created by Angus}

>> Import["ExampleData/ExampleData.txt", {"ExampleFormat2", "Data"}] // Grid
0.629452 0.586355
0.711009 0.687453
0.24654 0.433973
0.926871 0.887255
0.825141 0.9409
0.847035 0.127464
0.054348 0.296494
0.838545 0.247025
0.838697 0.43622
0.309496 0.833591

```

URLFetch

WMA link

URLFetch[<i>URL</i>]	# =
Returns the content of <i>URL</i> as a string.	

...

29. Integer Functions

Integer Functions can work on integers of any size.

Contents

Combinatorial Functions	348	RussellRaoDissimilarity	350	Mod	353
Binomial	349	SokalSneathDissimilarity	351	ModularInverse	354
CatalanNumber	349	Subsets	352	PowerMod	354
DiceDissimilarity	349	YuleDissimilarity	352	Quotient	354
JaccardDissimilarity	349			QuotientRemainder	355
MatchingDissimilarity	349	Division-Related Functions	352	Recurrence and Sum Functions	355
Multinomial	350	CompositeQ	352	Fibonacci	355
RogersTanimotoDissimilarity	350	Divisible	352	HarmonicNumber	355
		GCD	353	StirlingS1	356
		LCM	353	StirlingS2	356

Combinatorial Functions

Combinatorial Functions

Combinatorics is an area of mathematics primarily concerned with counting, both as a means and an end in obtaining results, and certain properties of finite structures.

It is closely related to many other areas of Mathematics and has many applications ranging from logic to statistical physics, from evolutionary biology to computer science, etc.

Binomial

Binomial Coefficient (SymPy, WMA)

`Binomial[n, k]`

gives the binomial coefficient n choose k .

```
>> Binomial[5, 3]
10
```

Binomial supports inexact numbers:

```
>> Binomial[10.5, 3.2]
165.286
```

Some special cases:

```
>> Binomial[10, -2]
0
>> Binomial[-10.5, -3.5]
0.
```

CatalanNumber

Catalan Number (SymPy, WMA)

```
CatalanNumber[n]  
gives the  $n$ th Catalan number.
```

A list of the first five Catalan numbers:

```
>> Table[CatalanNumber[n], {n, 1, 5}]  
{1, 2, 5, 14, 42}
```

DiceDissimilarity

Sørensen–Dice coefficient (Sympy, DiceDissimilarity)

```
DiceDissimilarity[u, v]  
returns the Dice dissimilarity between the two boolean 1-D lists  $u$  and  $v$ . This is defined as  $(c_{tf} + c_{ft}) / (2 * c_{tt} + c_{ft} + c_{tf})$ .  $n$  is  $\text{len}(u)$  and  $c_{ij}$  is the number of occurrences of  $u[k]=i$  and  $v[k]=j$  for  $k < n$ .
```

```
>> DiceDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]  
1  
--  
2
```

JaccardDissimilarity

Jaccard index (SciPy, WMA)

```
JaccardDissimilarity[u, v]  
returns the Jaccard–Needham dissimilarity between the two boolean 1-D lists  $u$  and  $v$ , which is defined as  $(c_{tf} + c_{ft}) / (c_{tt} + c_{ft} + c_{tf})$ , where  $n$  is  $\text{len}(u)$  and  $c_{ij}$  is the number of occurrences of  $u[k]=i$  and  $v[k]=j$  for  $k < n$ .
```

```
>> JaccardDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]  
2  
--  
3
```

MatchingDissimilarity

WMA link

```
MatchingDissimilarity[u, v]  
returns the Matching dissimilarity between the two boolean 1-D lists  $u$  and  $v$ , which is defined as  $(c_{tf} + c_{ft}) / n$ , where  $n$  is  $\text{len}(u)$  and  $c_{ij}$  is the number of occurrences of  $u[k]=i$  and  $v[k]=j$  for  $k < n$ .
```

```
>> MatchingDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]  
4  
--  
7
```

Multinomial

Multinomial distribution (WMA)

```
Multinomial[n1, n2, ...]
gives the multinomial coefficient  $(n_1+n_2+\dots)!/(n_1!n_2!\dots)$ .
```

```
>> Multinomial[2, 3, 4, 5]
2522520
>> Multinomial[]
1
```

Multinomial is expressed in terms of Binomial:

```
>> Multinomial[a, b, c]
Binomial[a, a] Binomial[a + b, b] Binomial[a + b + c, c]
```

Multinomial[n-k, k] is equivalent to Binomial[n, k].

```
>> Multinomial[2, 3]
10
```

RogersTanimotoDissimilarity

WMA link

```
RogersTanimotoDissimilarity[u, v]
returns the Rogers-Tanimoto dissimilarity between the two boolean 1-D lists u and v, which
is defined as  $R / (c_{tt} + c_{ff} + R)$  where  $n$  is  $\text{len}(u)$ ,  $c_{ij}$  is the number of occurrences of  $u[k]=i$ 
and  $v[k]=j$  for  $k < n$ , and  $R = 2 * (c_{tf} + c_{ft})$ .
```

```
>> RogersTanimotoDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0,
1}]
8
11
```

RussellRaoDissimilarity

WMA link

```
RussellRaoDissimilarity[u, v]
returns the Russell-Rao dissimilarity between the two boolean 1-D lists u and v, which is
defined as  $(n - c_{tt}) / c_{tt}$  where  $n$  is  $\text{len}(u)$  and  $c_{ij}$  is the number of occurrences of  $u[k]=i$  and
 $v[k]=j$  for  $k < n$ .
```

```
>> RussellRaoDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]
5
7
```

SokalSneathDissimilarity

WMA link

SokalSneathDissimilarity[*u*, *v*]
 returns the Sokal-Sneath dissimilarity between the two boolean 1-D lists *u* and *v*, which is defined as $R / (c_{tt} + R)$ where n is $\text{len}(u)$, c_{ij} is the number of occurrences of $u[k]=i$ and $v[k]=j$ for $k < n$, and $R = 2 * (c_{tf} + c_{ft})$.

```
>> SokalSneathDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]
4
5
```

Subsets

WMA link

Subsets[*list*]
 finds a list of all possible subsets of *list*.
Subsets[*list*, *n*]
 finds a list of all possible subsets containing at most *n* elements.
Subsets[*list*, {*n*}]
 finds a list of all possible subsets containing exactly *n* elements.
Subsets[*list*, {*min*, *max*}]
 finds a list of all possible subsets containing between *min* and *max* elements.
Subsets[*list*, *spec*, *n*]
 finds a list of the first *n* possible subsets.
Subsets[*list*, *spec*, {*n*}]
 finds the *n*th possible subset.

All possible subsets (power set):

```
>> Subsets[{a, b, c}]
{{}, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}
```

All possible subsets containing up to 2 elements:

```
>> Subsets[{a, b, c, d}, 2]
{{}, {a}, {b}, {c}, {d}, {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}}
```

Subsets containing exactly 2 elements:

```
>> Subsets[{a, b, c, d}, {2}]
{{a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}}
```

The first 5 subsets containing 3 elements:

```
>> Subsets[{a, b, c, d, e}, {3}, 5]
{{a, b, c}, {a, b, d}, {a, b, e}, {a, c, d}, {a, c, e}}
```

All subsets with even length:

```
>> Subsets[{a, b, c, d, e}, {0, 5, 2}]
{{}, {a, b}, {a, c}, {a, d}, {a, e}, {b, c}, {b, d}, {b, e}, {c, d}, {c, e}, {d, e}, {a, b, c, d}, {a, b, c, e}, {a, b, d, e}, {a, c, d, e}}
```

The 25th subset:

```
>> Subsets[Range[5], All, {25}]
{{2, 4, 5}}
```

The odd-numbered subsets of {a,b,c,d} in reverse order:

```
>> Subsets[{a, b, c, d}, All, {15, 1, -2}]
{{b,c,d},{a,b,d},{c,d},{b,c},{a,c},{d},{b},{}}
```

YuleDissimilarity

WMA link

YuleDissimilarity[*u*, *v*]

returns the Yule dissimilarity between the two boolean 1-D lists *u* and *v*, which is defined as $R / (c_{tt} * c_{ff} + R / 2)$ where *n* is len(*u*), c_{ij} is the number of occurrences of $u[k]=i$ and $v[k]=j$ for $k < n$, and $R = 2 * c_{tf} * c_{ft}$.

```
>> YuleDissimilarity[{1, 0, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 0, 1}]
6
5
```

Division-Related Functions

Division-Related Function

CompositeQ

WMA link

CompositeQ[*n*]

returns True if *n* is a composite number

- A composite number is a positive number that is the product of two integers other than 1.
- For negative integer *n*, **CompositeQ**[*n*] is effectively equivalent to **CompositeQ**[*-n*].

```
>> Table[CompositeQ[n], {n, 0, 10}]
{False, False, False, False, True, False, True, False, True, True}
```

Divisible

WMA link

Divisible[*n*, *m*]

returns True if *n* is divisible by *m*, and False otherwise.

 n is divisible by *m* if *n* is the product of *m* by an integer. **Divisible**[*n*,*m*] is effectively equivalent to **Mod**[*n*,*m*]==0.

Test whether the number 10 is divisible by 2

```
>> Divisible[10, 2]
True
```

But the other way around is False: 2 is not divisible by 10:

```
>> Divisible[2, 10]
False
```

GCD

WMA link

```
GCD[n1, n2, ...]  
computes the greatest common divisor of the given integers.
```

```
>> GCD[20, 30]  
10  
>> GCD[10, y]  
GCD [10,y]  
  
GCD is Listable:  
>> GCD[4, {10, 11, 12, 13, 14}]  
{2,1,4,1,2}
```

GCD does not work for rational numbers and Gaussian integers yet.

LCM

WMA link

```
LCM[n1, n2, ...]  
computes the least common multiple of the given integers.
```

```
>> LCM[15, 20]  
60  
>> LCM[20, 30, 40, 50]  
600
```

Mod

WMA link

```
Mod[x, m]  
returns  $x$  modulo  $m$ .
```

```
>> Mod[14, 6]  
2  
>> Mod[-3, 4]  
1  
>> Mod[-3, -4]  
-3  
>> Mod[5, 0]  
The argument 0 should be nonzero.  
Mod[5,0]
```

ModularInverse

Modular multiplicative inverse (SymPy, WMA)

```
ModularInverse[k, n]
    returns the modular inverse  $k^{-1} \pmod{n}$ .
```

`ModularInverse[k, n]` gives the smallest positive integer r where the remainder of the division of $r \times k$ by n is equal to 1.

```
>> ModularInverse[2, 3]
2
```

The following is be True for all values n, k which have a modular inverse:

```
>> k = 2; n = 3; Mod[ModularInverse[k, n] * k, n] == 1
True
```

Some modular inverses just do not exists. For example when k is a multiple of n :

```
>> ModularInverse[k, k]
ModularInverse[2, 2]
```

PowerMod

Modular exponentiaion. See https://en.wikipedia.org/wiki/Modular_exponentiation.

```
PowerMod[x, y, m]
    computes  $x^y \pmod{m}$ .
```

```
>> PowerMod[2, 10000000, 3]
1
>> PowerMod[3, -2, 10]
9
>> PowerMod[0, -1, 2]
0 is not invertible modulo 2.
PowerMod[0, -1, 2]
>> PowerMod[5, 2, 0]
The argument 0 should be nonzero.
PowerMod[5, 2, 0]
```

`PowerMod` does not support rational coefficients (roots) yet.

Quotient

WMA link

```
Quotient[m, n]
    computes the integer quotient of  $m$  and  $n$ .
```

```
>> Quotient[23, 7]
3
```

QuotientRemainder

WMA link

```
QuotientRemainder[m, n]
computes a list of the quotient and remainder from division of m by n.
```

```
>> QuotientRemainder[23, 7]
{3,2}
```

Recurrence and Sum Functions

Recurrence and Sum Functions

A recurrence relation is an equation that recursively defines a sequence or multidimensional array of values, once one or more initial terms are given; each further term of the sequence or array is defined as a function of the preceding terms.

Fibonacci

WMA link

```
Fibonacci[n]
computes the nth Fibonacci number.
```

```
>> Fibonacci[0]
0
>> Fibonacci[1]
1
>> Fibonacci[10]
55
>> Fibonacci[200]
280571172992510140037611932413038677189525
```

HarmonicNumber

Harmonic Number

(WMA link)

```
HarmonicNumber[n]
returns the nth harmonic number.
```

```
>> Table[HarmonicNumber[n], {n, 8}]
{1, 3/2, 11/6, 25/12, 137/60, 49/20, 363/140, 761/280}
>> HarmonicNumber[3.8]
2.03806
```

StirlingS1

Stirling numbers of first kind (WMA link)

```
StirlingS1[n, m]  
gives the Stirling number of the first kind  $n^m$ .
```

Integer mathematical function, suitable for both symbolic and numerical manipulation. gives the number of permutations of n elements that contain exactly m cycles.

```
>> StirlingS1[50, 1]  
- 608 281 864 034 267 560 872 252 163 321 295 376 887 552 831 379 210 240 000 000 000
```

StirlingS2

Stirling numbers of second kind (WMA link)

```
StirlingS2[n, m]  
gives the Stirling number of the second kind  $n^m$ .
```

returns the number of ways of partitioning a set of n elements into m non empty subsets.

```
>> Table[StirlingS2[10, m], {m, 10}]  
{1, 511, 9330, 34105, 42525, 22827, 5880, 750, 45, 1}
```

30. Integer and Number-Theoretical Functions

Contents

Algebraic Transformations	358	Solve	381	Inverse	394
Apart	359	Differential Equations	381	LeastSquares	394
Cancel	359	C	381	LinearModelFit	395
Coefficient	360	DSolve	381	LinearSolve	395
CoefficientArrays	360	Exponential Functions	381	MatrixExp	396
CoefficientList	361	Exp	382	MatrixPower	396
Collect	361	Log	382	MatrixRank	396
Denominator	362	Log10	383	NullSpace	397
Expand	363	Log2	383	PseudoInverse	397
ExpandAll	364	LogisticSigmoid	383	QRDecomposition	397
ExpandDenominator	364	Hyperbolic Functions	384	RowReduce	398
Exponent	364	ArcCosh	384	SingularValueDecomposition	398
Factor	365	ArcCoth	384	Tr	398
FactorTermsList	366	ArcCsch	385	Mathematical Constants	398
FullSimplify	366	ArcSech	385	Catalan	398
MinimalPolynomial	366	ArcSinh	385	ComplexInfinity	399
Numerator	366	ArcTanh	385	Degree	399
PolynomialQ	367	Cosh	386	E	399
PowerExpand	367	Coth	386	EulerGamma	400
Simplify	368	Gudermannian	386	Glaisher	400
Together	369	InverseGudermannian	387	GoldenRatio	400
Variables	369	Sech	387	Indeterminate	400
Calculus	369	Sinh	387	Infinity	401
Complexes	369	Tanh	387	Khinchin	401
D	371	Integer Functions	388	\$MaxMachineNumber	401
Derivative (')	372	BitLength	388	\$MinMachineNumber	401
DiscreteLimit	372	Ceiling	388	Overflow	402
FindMaximum	373	DigitCount	389	Pi	402
FindMinimum	373	Floor	389	Undefined	403
FindRoot	374	FromDigits	390	Underflow	403
Integers	375	IntegerDigits	391	Number theoretic functions	403
Integrate	376	IntegerReverse	391	ContinuedFraction	403
Limit	376	IntegerString	392	Divisors	404
NIntegrate	377	Linear algebra	392	EulerPhi	404
O	377	DesignMatrix	392	FactorInteger	405
Reals	377	Det	392	FractionalPart	405
Root	377	Eigensystem	393	FromContinuedFraction	405
Series	378	Eigenvalues	393		
SeriesData	379	Eigenvectors	393		
		FittedModel	394		

MantissaExponent	406	RandomInteger	411
NextPrime	406	RandomReal	411
PartitionsP	406	RandomSample	412
Prime	407	\$RandomState	412
PrimePi	407	SeedRandom	413
PrimePowerQ	408	Trigonometric Functions 413	
RandomPrime	408	AnglePath	414
Random number generation	408	ArcCos	414
Random	408	ArcCot	415
RandomChoice	409	ArcCsc	415
RandomComplex	410	ArcSec	415
		ArcSin	416
		ArcTan	416
		Cos	416
		Cot	416
		Csc	417
		Haversine	417
		InverseHaversine	417
		Sec	418
		Sin	418
		Tan	418

Algebraic Transformations

Algebraic Transformations

There are a number of built-in functions that perform:

- Structural Operations on Polynomials
- Finding the Structure of a Polynomial
- Structural Operations on Rational Expressions
- Polynomials over Algebraic Number Fields
- Simplification with or without Assumptions

Apart

WMA link

Apart [*expr*]

writes *expr* as a sum of individual fractions.

Apart [*expr*, *var*]

treats *var* as the main variable.

```
>> Apart[1 / (x^2 + 5x + 6)]

$$\frac{1}{2+x} - \frac{1}{3+x}$$

```

When several variables are involved, the results can be different depending on the main variable:

```
>> Apart[1 / (x^2 - y^2), x]
```

$$-\frac{1}{2y(x+y)} + \frac{1}{2y(x-y)}$$

```
>> Apart[1 / (x^2 - y^2), y]
```

$$\frac{1}{2x(x+y)} + \frac{1}{2x(x-y)}$$

Apart is Listable:

```
>> Apart[{1 / (x^2 + 5x + 6)}]
```

$$\left\{ \frac{1}{2+x} - \frac{1}{3+x} \right\}$$

But it does not touch other expressions:

```
>> Sin[1 / (x ^ 2 - y ^ 2)] // Apart
Sin  $\left[ \frac{1}{x^2 - y^2} \right]$ 
```

Cancel

WMA link

Cancel [*expr*]
cancels out common factors in numerators and denominators.

```
>> Cancel[x / x ^ 2]
 $\frac{1}{x}$ 

Cancel threads over sums:
>> Cancel[x / x ^ 2 + y / y ^ 2]
 $\frac{1}{x} + \frac{1}{y}$ 

>> Cancel[f[x] / x + x * f[x] / x ^ 2]
 $\frac{2f[x]}{x}$ 
```

Coefficient

WMA link

Coefficient [*expr*, *form*]
returns the coefficient of *form* in the polynomial *expr*.
Coefficient [*expr*, *form*, *n*]
return the coefficient of *form*^{*n*} in *expr*.

```
>> Coefficient[(x + y)^4, (x^2)*(y^2)]
6

>> Coefficient[a x^2 + b y^3 + c x + d y + 5, x]
c

>> Coefficient[(x + 3 y)^5, x]
405y^4

>> Coefficient[(x + 3 y)^5, x * y^4]
405

>> Coefficient[(x + 2)/(y - 3) + (x + 3)/(y - 2), x]
 $\frac{1}{-3 + y} + \frac{1}{-2 + y}$ 

>> Coefficient[x*Cos[x + 3] + 6*y, x]
Cos[3 + x]
```

```

>> Coefficient[(x + 1)^3, x, 2]
3
>> Coefficient[a x^2 + b y^3 + c x + d y + 5, y, 3]
b

```

Find the free term in a polynomial:

```

>> Coefficient[(x + 2)^3 + (x + 3)^2, x, 0]
17
>> Coefficient[(x + 2)^3 + (x + 3)^2, y, 0]
(2 + x)^3 + (3 + x)^2
>> Coefficient[a x^2 + b y^3 + c x + d y + 5, x, 0]
5 + b y^3 + d y

```

CoefficientArrays

WMA link

```
CoefficientArrays[polys, vars]
```

returns a list of arrays of coefficients of the variables *vars* in the polynomial *poly*.

```

>> CoefficientArrays[1 + x^3, x]
{1, {0}, {{0}}, {{{1}}}}
>> CoefficientArrays[1 + x y+ x^3, {x, y}]
{1, {0, 0}, {{0, 1}, {0, 0}}, {{{1, 0}, {0, 0}}, {{0, 0}, {0, 0}}}}
>> CoefficientArrays[{1 + x^2, x y}, {x, y}]
{{1, 0}, {{0, 0}, {0, 0}}, {{{1, 0}, {0, 0}}, {{0, 1}, {0, 0}}}}
>> CoefficientArrays[(x+y+Sin[z])^3, {x,y}]
{Sin[z]^3, {3Sin[z]^2, 3Sin[z]^2}, {{3Sin[z], 6Sin[z]}, {0, 3Sin[z]}}, {{{1, 3}, {0, 3}}, {{0, 0}, {0, 1}}}}
>> CoefficientArrays[(x + y + Sin[z])^3, {x, z}]
(x + y + Sin[z]) ^ 3 is not a polynomial in {x, z}
CoefficientArrays[(x + y + Sin[z])^3, {x, z}]

```

CoefficientList

WMA link

```
CoefficientList[poly, var]
```

returns a list of coefficients of powers of *var* in *poly*, starting with power 0.

```
CoefficientList[poly, {var1, var2, ...}]
```

returns an array of coefficients of the *vari*.

```

>> CoefficientList[(x + 3)^5, x]
{243, 405, 270, 90, 15, 1}

>> CoefficientList[(x + y)^4, x]
{y^4, 4y^3, 6y^2, 4y, 1}

>> CoefficientList[a x^2 + b y^3 + c x + d y + 5, x]
{5 + by^3 + dy, c, a}

>> CoefficientList[(x + 2)/(y - 3) + x/(y - 2), x]
{2/(-3 + y), 1/(-3 + y) + 1/(-2 + y)}

>> CoefficientList[(x + y)^3, z]
{(x + y)^3}

>> CoefficientList[a x^2 + b y^3 + c x + d y + 5, {x, y}]
{{5, d, 0, b}, {c, 0, 0, 0}, {a, 0, 0, 0} }

>> CoefficientList[(x - 2 y + 3 z)^3, {x, y, z}]
{{0, 0, 0, 27}, {0, 0, -54, 0}, {0, 36, 0, 0}, {-8, 0, 0, 0}}, {{0, 0, 27, 0}, {0, -36, 0, 0}, {12, 0, 0, 0}, {0, 0, 0, 0}}, {{0,

```

Collect

WMA link

<pre>Collect[expr, x]</pre>	Expands <i>expr</i> and collect together terms having the same power of <i>x</i> .
<pre>Collect[expr, {x_1, x_2, ...}]</pre>	Expands <i>expr</i> and collect together terms having the same powers of <i>x</i> ₁ , <i>x</i> ₂ ,
<pre>Collect[expr, {x_1, x_2, ...}, filter]</pre>	After collect the terms, applies <i>filter</i> to each coefficient.

```

>> Collect[(x+y)^3, y]
x^3 + 3x^2y + 3xy^2 + y^3

>> Collect[2 Sin[x z] (x+2 y^2 + Sin[y] x), y]
2xSin[xz] + 2xSin[xz]Sin[y] + 4y^2Sin[xz]

>> Collect[3 x y+2 Sin[x z] (x+2 y^2 + x)+ (x+y)^3, y]
4xSin[xz] + x^3 + y(3x + 3x^2) + y^2(3x + 4Sin[xz]) + y^3

>> Collect[3 x y+2 Sin[x z] (x+2 y^2 + x)+ (x+y)^3, {x,y}]
4xSin[xz] + x^3 + 3xy + 3x^2y + 4y^2Sin[xz] + 3xy^2 + y^3

>> Collect[3 x y+2 Sin[x z] (x+2 y^2 + x)+ (x+y)^3, {x,y}, h]
xh[4Sin[xz]] + x^3h[1] + xyh[3] + x^2yh[3] + y^2h[4Sin[xz]] + xy^2h[3] + y^3h[1]

```

Denominator

WMA link

```
Denominator[expr]
gives the denominator in expr.
```

```
>> Denominator[a / b]
b
>> Denominator[2 / 3]
3
>> Denominator[a + b]
1
```

Expand

WMA link

```
Expand[expr]
expands out positive integer powers and products of sums in expr, as well as trigonometric identities.
Expand[expr, target]
just expands those parts involving target.
```

```
>> Expand[(x + y)^ 3]
x3 + 3x2y + 3xy2 + y3
>> Expand[(a + b)(a + c + d)]
a2 + ab + ac + ad + bc + bd
>> Expand[(a + b)(a + c + d)(e + f) + e a a]
2a2e + a2f + abe + abf + ace + acf + ade +adf + bce + bcf + bde + bdf
>> Expand[(a + b)^ 2 * (c + d)]
a2c + a2d + 2abc + 2abd + b2c + b2d
>> Expand[(x + y)^ 2 + x y]
x2 + 3xy + y2
>> Expand[((a + b)(c + d))^ 2 + b (1 + a)]
a2c2 + 2a2cd + a2d2 + b + ab + 2abc2 + 4abcd + 2abd2 + b2c2 + 2b2cd + b2d2
```

Expand expands items in lists and rules:

```
>> Expand[{4 (x + y), 2 (x + y) -> 4 (x + y)}]
{4x + 4y, 2x + 2y -> 4x + 4y}
```

Expand expands trigonometric identities

```
>> Expand[Sin[x + y], Trig -> True]
Cos[x] Sin[y] + Cos[y] Sin[x]
```

```
>> Expand[Tanh[x + y], Trig -> True]

$$\frac{\cosh[x]\sinh[y]}{\cosh[x]\cosh[y] + \sinh[x]\sinh[y]} + \frac{\cosh[y]\sinh[x]}{\cosh[x]\cosh[y] + \sinh[x]\sinh[y]}$$

```

Expand does not change any other expression.

```
>> Expand[Sin[x (1 + y)]]
```

$$\sin[x(1+y)]$$

Using the second argument, the expression only expands those subexpressions containing *pat*:

```
>> Expand[(x+a)^2+(y+a)^2+(x+y)(x+a), y]
```

$$a^2 + 2ay + x(a+x) + y(a+x) + y^2 + (a+x)^2$$

Expand also works in Galois fields

```
>> Expand[(1 + a)^12, Modulus -> 3]
```

$$1 + a^3 + a^9 + a^{12}$$

```
>> Expand[(1 + a)^12, Modulus -> 4]
```

$$1 + 2a^2 + 3a^4 + 3a^8 + 2a^{10} + a^{12}$$

ExpandAll

[WMA link](#)

<pre>ExpandAll[expr]</pre> <p>expands out negative integer powers and products of sums in <i>expr</i>.</p>
<pre>ExpandAll[expr, target]</pre> <p>just expands those parts involving <i>target</i>.</p>

```
>> ExpandAll[(a + b)^2 / (c + d)^2]

$$\frac{a^2}{c^2 + 2cd + d^2} + \frac{2ab}{c^2 + 2cd + d^2} + \frac{b^2}{c^2 + 2cd + d^2}$$

```

ExpandAll descends into sub expressions

```
>> ExpandAll[(a + Sin[x (1 + y)])^2]
```

$$2a\sin[x + xy] + a^2 + \sin[x + xy]^2$$

```
>> ExpandAll[Sin[(x+y)^2]]
```

$$\sin[x^2 + 2xy + y^2]$$

```
>> ExpandAll[Sin[(x+y)^2], Trig->True]
```

$$\cos[x^2]\cos[2xy]\sin[y^2] + \cos[x^2]\cos[y^2]\sin[2xy] + \cos[2xy]\cos[y^2]\sin[x^2] - \sin[x^2]\sin[2xy]\sin[y^2]$$

ExpandAll also expands heads

```
>> ExpandAll[((1 + x)(1 + y))[x]]
```

$$(1 + x + y + xy)[x]$$

ExpandAll can also work in finite fields

```
>> ExpandAll[(1 + a)^ 6 / (x + y)^3, Modulus -> 3]

$$\frac{1 + 2a^3 + a^6}{x^3 + y^3}$$

```

ExpandDenominator

WMA link

```
ExpandDenominator[expr]
expands out negative integer powers and products of sums in expr.
```

```
>> ExpandDenominator[(a + b)^ 2 / ((c + d)^2 (e + f))]

$$\frac{(a + b)^2}{c^2e + c^2f + 2cde + 2cdf + d^2e + d^2f}$$

```

Exponent

WMA link

```
Exponent[expr, form]
returns the maximum power with which form appears in the expanded form of expr.
Exponent[expr, form, h]
applies h to the set of exponents with which form appears in expr.
```

```
>> Exponent[5 x^2 - 3 x + 7, x]
2
>> Exponent[(x^3 + 1)^2 + 1, x]
6
>> Exponent[x^(n + 1) + Sqrt[x] + 1, x]
Max  $\left[\frac{1}{2}, 1 + n\right]$ 
>> Exponent[x / y, y]
-1
>> Exponent[(x^2 + 1)^3 - 1, x, Min]
2
>> Exponent[0, x]
-\infty
>> Exponent[1, x]
0
```

Factor

WMA link

```
Factor[expr]
factors the polynomial expression expr.
```

```

>> Factor[x ^ 2 + 2 x + 1]
(1 + x)^2

>> Factor[1 / (x^2+2x+1)+ 1 / (x^4+2x^2+1)]

$$\frac{2 + 2x + 3x^2 + x^4}{(1 + x)^2 (1 + x^2)^2}$$


```

Factor can also be used with equations:

```

>> Factor[x a == x b + x c]
ax==x(b+c)

```

And lists:

```

>> Factor[{x + x^2, 2 x + 2 y + 2}]
{x(1+x), 2(1+x+y)}

```

It also works with more complex expressions:

```

>> Factor[x ^ 3 + 3 x ^ 2 y + 3 x y ^ 2 + y ^ 3]
(x+y)^3

```

You can use Factor to find when a polynomial is zero:

```

>> x^2 - x == 0 // Factor
x(-1+x)==0

```

FactorTermsList

WMA link

FactorTermsList [poly]

returns a list of 2 elements. The first element is the numerical factor in *poly*. The second one is the remaining of the polynomial with numerical factor removed.

FactorTermsList [poly, {x₁, x₂, ...}]

returns a list of factors in *poly*. The first element is the numerical factor in *poly*. The next ones are factors that are independent of variables lists which are created by removing each variable *x_i* from right to left. The last one is the remaining of polynomial after dividing *poly* to all previous factors.

```

>> FactorTermsList[2 x^2 - 2]
{2, -1 + x^2}

>> FactorTermsList[x^2 - 2 x + 1]
{1, 1 - 2x + x^2}

>> f = 3 (-1 + 2 x)(-1 + y)(1 - a)
3(-1+2x)(-1+y)(1-a)

>> FactorTermsList[f]
{-3, -1 + a - 2ax - ay + 2x + y - 2xy + 2axy}

>> FactorTermsList[f, x]
{-3, 1 - a - y + ay, -1 + 2x}

```

FullSimplify

WMA link

```
FullSimplify[expr]
    simplifies expr using an extended set of simplification rules.
FullSimplify[expr, assump]
    simplifies expr assuming assump instead of Assumptions.
```

TODO:

implement the extension. By now, this does the same than Simplify...

```
>> FullSimplify[2*Sin[x]^2 + 2*Cos[x]^2]
2
```

MinimalPolynomial

WMA link

```
MinimalPolynomial[s, x]
gives the minimal polynomial in x for which the algebraic number s is a root.
```

```
>> MinimalPolynomial[7, x]
-7 + x

>> MinimalPolynomial[Sqrt[2] + Sqrt[3], x]
1 - 10x^2 + x^4

>> MinimalPolynomial[Sqrt[1 + Sqrt[3]], x]
-2 - 2x^2 + x^4

>> MinimalPolynomial[Sqrt[I + Sqrt[6]], x]
49 - 10x^4 + x^8
```

Numerator

WMA link

```
Numerator[expr]
gives the numerator in expr.
```

```
>> Numerator[a / b]
a

>> Numerator[2 / 3]
2

>> Numerator[a + b]
a + b
```

PolynomialQ

WMA link

```

PolynomialQ[expr, var]
    returns True if expr is a polynomial in var, and returns False otherwise.
PolynomialQ[expr, {var1, ...}]
    tests whether expr is a polynomial in the vari.

```

```

>> PolynomialQ[x^3 - 2 x/y + 3xz, x]
True
>> PolynomialQ[x^3 - 2 x/y + 3xz, y]
False
>> PolynomialQ[f[a] + f[a]^2, f[a]]
True
>> PolynomialQ[x^2 + axy^2 - bSin[c], {x, y}]
True
>> PolynomialQ[x^2 + axy^2 - bSin[c], {a, b, c}]
False

```

PowerExpand

WMA link

```

PowerExpand[expr]
    expands out powers of the form  $(x^a y)^b$  and  $(x y^a)^b$  in expr.

```

```

>> PowerExpand[(a ^ b)^ c]
abc
>> PowerExpand[(a * b)^ c]
ac bc

```

PowerExpand is not correct without certain assumptions:

```

>> PowerExpand[(x ^ 2)^(1/2)]
x

```

Simplify

WMA link

```

Simplify[expr]
    simplifies expr.
Simplify[expr, assump]
    simplifies expr assuming assump instead of Assumptions.

```

```

>> Simplify[2*Sin[x]^2 + 2*Cos[x]^2]
2
>> Simplify[x]
x

```

```

>> Simplify[f[x]]
f[x]

Simplify over conditional expressions uses $Assumptions, or assump to evaluate the condition:
>> $Assumptions={a <= 0};

>> Simplify[ConditionalExpression[1, a > 0]]
Undefined

```

The *assump* option override \$Assumption:

```

>> Simplify[ConditionalExpression[1, a > 0] ConditionalExpression[1, b > 0], {b > 0}]
ConditionalExpression[1, a > 0]

```

On the other hand, *Assumptions* option does not override \$Assumption, but add to them:

```

>> Simplify[ConditionalExpression[1, a > 0] ConditionalExpression[1, b > 0], Assumptions -> {b > 0}]
ConditionalExpression[1, a > 0]

```

Passing both options overwrites \$Assumptions with the union of *assump* the option

```

>> Simplify[ConditionalExpression[1, a > 0] ConditionalExpression[1, b > 0], {a>0}, Assumptions -> {b > 0}]
1
>> $Assumptions={};

```

The option *ComplexityFunction* allows to control the way in which the evaluator decides if one expression is simpler than another. For example, by default, Simplify tries to avoid expressions involving numbers with many digits:

```

>> Simplify[20 Log[2]]
20Log[2]

```

This behaviour can be modified by setting LeafCount as the ComplexityFunction:

```

>> Simplify[20 Log[2], ComplexityFunction->LeafCount]
Log[1048576]

```

Together

WMA link

Together[<i>expr</i>]
writes sums of fractions in <i>expr</i> together.

```

>> Together[a / c + b / c]

$$\frac{a+b}{c}$$


```

Together operates on lists:

```
>> Together[{x / (y+1) + x / (y+1)^2}]

$$\left\{ \frac{x(2+y)}{(1+y)^2} \right\}$$

```

But it does not touch other functions:

```
>> Together[f[a / c + b / c]]
f \left[ \frac{a}{c} + \frac{b}{c} \right]
```

Variables

WMA link

Variables [*expr*]
gives a list of the variables that appear in the polynomial *expr*.

```
>> Variables[a x^2 + b x + c]
{a, b, c, x}

>> Variables[{a + b x, c y^2 + x/2}]
{a, b, c, x, y}

>> Variables[x + Sin[y]]
{x, Sin[y]}
```

Calculus

Calculus

Originally called infinitesimal calculus or “the calculus of infinitesimals”, is the mathematical study of continuous change, in the same way that geometry is the study of shape and algebra is the study of generalizations of arithmetic operations.

Complexes

WMA link

Complexes
the domain of complex numbers, as in *x* in Complexes.

D

Derivative (WMA)

$D[f, x]$	gives the partial derivative of f with respect to x .
$D[f, x, y, \dots]$	differentiates successively with respect to x, y , etc.
$D[f, \{x, n\}]$	gives the multiple derivative of order n .
$D[f, \{\{x1, x2, \dots\}\}]$	gives the vector derivative of f with respect to $x1, x2$, etc.

First-order derivative of a polynomial:

```
>> D[x^3 + x^2, x]
2x + 3x^2
```

Second-order derivative:

```
>> D[x^3 + x^2, {x, 2}]
2 + 6x
```

Trigonometric derivatives:

```
>> D[Sin[Cos[x]], x]
-Cos[Cos[x]] Sin[x]

>> D[Sin[x], {x, 2}]
-Sin[x]

>> D[Cos[t], {t, 2}]
-Cos[t]
```

Unknown variables are treated as constant:

```
>> D[y, x]
0

>> D[x, x]
1

>> D[x + y, x]
1
```

Derivatives of unknown functions are represented using Derivative:

```
>> D[f[x], x]
f'[x]

>> D[f[x, x], x]
f^{(0,1)}[x, x] + f^{(1,0)}[x, x]

>> D[f[x, x], x] // InputForm
Derivative[0, 1][f][x, x] + Derivative[1, 0][f][x, x]
```

Chain rule:

```
>> D[f[2x+1, 2y, x+y], x]
2f^{(1,0,0)}[1 + 2x, 2y, x + y] + f^{(0,0,1)}[1 + 2x, 2y, x + y]

>> D[f[x^2, x, 2y], {x, 2}, y] // Expand
8x f^{(1,1,1)}[x^2, x, 2y] + 8x^2 f^{(2,0,1)}[x^2, x, 2y] + 2f^{(0,2,1)}[x^2, x, 2y] + 4f^{(1,0,1)}[x^2, x, 2y]
```

Compute the gradient vector of a function:

```
>> D[x ^ 3 * Cos[y], {{x, y}}]
```

$$\{3x^2 \text{Cos}[y], -x^3 \text{Sin}[y]\}$$

Hesse matrix:

```
>> D[Sin[x] * Cos[y], {{x,y}, 2}]
```

$$\{\{-\text{Cos}[y] \text{Sin}[x], -\text{Cos}[x] \text{Sin}[y]\}, \{-\text{Cos}[x] \text{Sin}[y], -\text{Cos}[y] \text{Sin}[x]\}\}$$

Derivative (')

WMA link

```
Derivative[n][f]
represents the nth derivative of the function f.
Derivative[n1, n2, ...][f]
represents a multivariate derivative.
```

```
>> Derivative[1][Sin]
Cos[#1]&

>> Derivative[3][Sin]
-Cos[#1]&

>> Derivative[2][# ^ 3&]
6#1&
```

Derivative can be entered using ':

```
>> Sin'[x]
Cos[x]

>> (# ^ 4)& ''
12#1^2&

>> f'[x] // InputForm
Derivative[1][f][x]

>> Derivative[1][#2 Sin[#1]+Cos[#2]&]
Cos[#1]#2&

>> Derivative[1,2][#2^3 Sin[#1]+Cos[#2]&]
6Cos[#1]#2&
```

Deriving with respect to an unknown parameter yields 0:

```
>> Derivative[1,2,1][#2^3 Sin[#1]+Cos[#2]&]
0&
```

The 0th derivative of any expression is the expression itself:

```
>> Derivative[0,0,0][a+b+c]
a + b + c
```

You can calculate the derivative of custom functions:

```
>> f[x_] := x ^ 2
```

```
>> f'[x]
2x
```

Unknown derivatives:

```
>> Derivative[2, 1][h]
h(2,1)

>> Derivative[2, 0, 1, 0][h[g]]
h [g](2,0,1,0)
```

DiscreteLimit

WMA link

```
DiscreteLimit[f, k->Infinity]
gives the limit of the sequence  $f$  as  $k$  tends to infinity.
```

```
>> DiscreteLimit[n/(n + 1), n -> Infinity]
1

>> DiscreteLimit[f[n], n -> Infinity]
f[∞]
```

FindMaximum

WMA link

```
FindMaximum[f, {x, x0}]
searches for a numerical maximum of  $f$ , starting from  $x=x0$ .
```

FindMaximum by default uses Newton's method, so the function of interest should have a first derivative.

```
>> FindMaximum[-(x-3)^2+2., {x, 1}]
Encountered a gradient that is effectively zero. The result returned
may not be a maximum; it may be a minimum or a saddle point.
{2., {x -> 3.} }

>> FindMaximum[-10*x^30*(x-3)^2+2., {x, 1}]
Encountered a gradient that is effectively zero. The result returned
may not be a maximum; it may be a minimum or a saddle point.
{2., {x -> 3.} }

>> FindMaximum[Sin[x], {x, 1}]
{1., {x -> 1.5708} }

>> phi[x_?NumberQ]:=NIntegrate[u, {u, 0., x}, Method->"Internal"];
>> Quiet[FindMaximum[-phi[x] + x, {x, 1.2}, Method->"Newton"]]
{0.5, {x -> 1.00001} }

>> Clear[phi];
```

For a not so well behaving function, the result can be less accurate:

```
>> FindMaximum[-Exp[-1/x^2]+1., {x,1.2}, MaxIterations->10]
The maximum number of iterations was exceeded. The result might be
inaccurate.
```

$$\text{FindMaximum} \left[-\text{Exp} \left[-\frac{1}{x^2} \right] + 1., \{x, 1.2\}, \text{MaxIterations} -> 10 \right]$$

FindMinimum

WMA link

```
FindMinimum[f, {x, x0}]
```

searches for a numerical minimum of f , starting from $x=x0$.

FindMinimum by default uses Newton's method, so the function of interest should have a first derivative.

```
>> FindMinimum[(x-3)^2+2., {x, 1}]
Encountered a gradient that is effectively zero. The result returned
may not be a minimum; it may be a maximum or a saddle point.

{2., {x -> 3.}}

>> FindMinimum[10*x^-30 *(x-3)^2+2., {x, 1}]
Encountered a gradient that is effectively zero. The result returned
may not be a minimum; it may be a maximum or a saddle point.

{2., {x -> 3.}}

>> FindMinimum[Sin[x], {x, 1}]
{-1., {x -> -1.5708}}

>> phi[x_?NumberQ]:=NIntegrate[u,{u,0,x}, Method->"Internal"];

>> Quiet[FindMinimum[phi[x]-x,{x, 1.2}, Method->"Newton"]]
{-0.5, {x -> 1.00001}]

>> Clear[phi];
```

For a not so well behaving function, the result can be less accurate:

```
>> FindMinimum[Exp[-1/x^2]+1., {x,1.2}, MaxIterations->10]
The maximum number of iterations was exceeded. The result might be
inaccurate.
```

$$\text{FindMinimum} \left[\text{Exp} \left[-\frac{1}{x^2} \right] + 1., \{x, 1.2\}, \text{MaxIterations} -> 10 \right]$$

FindRoot

WMA link

```
FindRoot[f, {x, x0}]
```

searches for a numerical root of f , starting from $x=x0$.

```
FindRoot[lhs == rhs, {x, x0}]
```

tries to solve the equation $lhs == rhs$.

FindRoot by default uses Newton's method, so the function of interest

should have a first derivative.

```
>> FindRoot[Cos[x], {x, 1}]
{x -> 1.5708}

>> FindRoot[Sin[x] + Exp[x], {x, 0}]
{x -> -0.588533}

>> FindRoot[Sin[x] + Exp[x] == Pi, {x, 0}]
{x -> 0.866815}
```

FindRoot has attribute HoldAll and effectively uses Block to localize x . However, in the result x will eventually still be replaced by its value.

```
>> x = "I am the result!";

>> FindRoot[Tan[x] + Sin[x] == Pi, {x, 1}]
{I am the result! -> 1.14911}

>> Clear[x]
```

FindRoot stops after 100 iterations:

```
>> FindRoot[x^2 + x + 1, {x, 1}]
The maximum number of iterations was exceeded. The result might be
inaccurate.
{x -> -1.}
```

Find complex roots:

```
>> FindRoot[x^2 + x + 1, {x, -I}]
{x -> -0.5 - 0.866025 I}
```

The function has to return numerical values:

```
>> FindRoot[f[x] == 0, {x, 0}]
The function value is not a number at x = 0..
FindRoot[f[x] - 0, {x, 0}]
```

The derivative must not be 0:

```
>> FindRoot[Sin[x] == x, {x, 0}]
Encountered a singular derivative at the point x = 0..
FindRoot[Sin[x] - x, {x, 0}]

>> FindRoot[x^2 - 2, {x, 1, 3}, Method -> "Secant"]
{x -> 1.41421}
```

Integers

WMA link

Integers
the domain of integer numbers, as in x in Integers.

Limit a solution to integer numbers:

```
>> Solve[-4 - 4 x + x^4 + x^5 == 0, x, Integers]
{{x -> -1}}
```

```
>> Solve[x^4 == 4, x, Integers]
{}
```

Integrate

WMA link

Integrate[f, x]
integrates f with respect to x . The result does not contain the additive integration constant.
Integrate[f, {x, a, b}]
computes the definite integral of f with respect to x from a to b .

Integrate a polynomial:

```
>> Integrate[6 x ^ 2 + 3 x ^ 2 - 4 x + 10, x]
x (10 - 2x + 3x^2)
```

Integrate trigonometric functions:

```
>> Integrate[Sin[x] ^ 5, x]
Cos[x] \left( -1 - \frac{\text{Cos}[x]^4}{5} + \frac{2\text{Cos}[x]^2}{3} \right)
```

Definite integrals:

```
>> Integrate[x ^ 2 + x, {x, 1, 3}]
38
—
3
>> Integrate[Sin[x], {x, 0, Pi/2}]
1
```

Some other integrals:

```
>> Integrate[1 / (1 - 4 x + x^2), x]
\sqrt{3} \left( \text{Log} \left[ -2 - \sqrt{3} + x \right] - \text{Log} \left[ -2 + \sqrt{3} + x \right] \right)
—————
6
>> Integrate[4 Sin[x] Cos[x], x]
2Sin[x]^2
```

> Integrate[-Infinity, {x, 0, Infinity}] = -Infinity

> Integrate[-Infinity, {x, Infinity, 0}] = Infinity

Integration in TeX:

```
>> Integrate[f[x], {x, a, b}] // TeXForm
\int_a^b f\left[x\right] dx
```

Sometimes there is a loss of precision during integration. You can check the precision of your result with the following sequence of commands.

```
>> Integrate[Abs[Sin[phi]], {phi, 0, 2Pi}] // N
4.
>> % // Precision
MachinePrecision
```

```

>> Integrate[ArcSin[x / 3], x]
x ArcSin  $\left[\frac{x}{3}\right] + \sqrt{9 - x^2}$ 

>> Integrate[f'[x], {x, a, b}]
f[b] - f[a]

and,
>> D[Integrate[f[u, x], {u, a[x], b[x]}], x]
 $\int_{a[x]}^{b[x]} f^{(0,1)}[u, x] du + f[b[x], x] b'[x] - f[a[x], x] a'[x]$ 

>> N[Integrate[Sin[Exp[-x^2 / 2]], {x, 1, 2}]]
0.330804

```

Limit

WMA link

```

Limit[expr, x->x0]
gives the limit of expr as x approaches x0.
Limit[expr, x->x0, Direction->1]
approaches x0 from smaller values.
Limit[expr, x->x0, Direction->-1]
approaches x0 from larger values.

```

```

>> Limit[x, x->2]
2

>> Limit[Sin[x] / x, x->0]
1

>> Limit[1/x, x->0, Direction->-1]
∞

>> Limit[1/x, x->0, Direction->1]
-∞

```

NIntegrate

WMA link

```

NIntegrate[expr, interval]
returns a numeric approximation to the definite integral of expr with limits interval and with a precision of prec digits.
NIntegrate[expr, interval1, interval2, ...]
returns a numeric approximation to the multiple integral of expr with limits interval1, interval2 and with a precision of prec digits.

```

```

>> NIntegrate[Exp[-x], {x, 0, Infinity}, Tolerance->1*^-6, Method-"Internal"]
1.

```

```

>> NIntegrate[Exp[x], {x, -Infinity, 0}, Tolerance -> 1*^-6, Method ->
  "Internal"]
1.

>> NIntegrate[Exp[-x^2/2.], {x, -Infinity, Infinity}, Tolerance -> 1*^-6,
  Method -> "Internal"]
2.50664

```

O

WMA link

$O[x]^n$

Represents a term of order x^n .

$O[x]^n$ is generated to represent omitted higher order terms in power series.

```

>> Series[1/(1-x), {x, 0, 2}]
1 + x + x^2 + O[x]^3

```

When called alone, a 'SeriesData' expression is built:

```

>> O[x] // FullForm
SeriesData[x, 0, {}, 1, 1, 1]

```

Reals

WMA link

Reals

is the domain real numbers, as in x in Reals.

Limit a solution to real numbers:

```

>> Solve[x^3 == 1, x, Reals]
{{x -> 1}}

```

Root

WMA link

$\text{Root}[f, i]$

represents the i -th complex root of the polynomial f .

```

>> Root[#1 ^ 2 - 1&, 1]
-1

>> Root[#1 ^ 2 - 1&, 2]
1

```

Roots that can't be represented by radicals:

```

>> Root[#1 ^ 5 + 2 #1 + 1&, 2]
Root[#1^5 + 2#1 + 1&, 2]

```

Series

WMA link

```
Series[f, {x, x0, n}]  
Represents the series expansion around  $x=x_0$  up to order  $n$ .
```

For elementary expressions, `Series` returns the explicit power series as a `SeriesData` expression:

```
>> Series[Exp[x], {x, 0, 2}]  
1 + x +  $\frac{1}{2}x^2 + O[x]^3$   
>> % // FullForm  
SeriesData[x, 0, {1, 1, Rational[1, 2]}, 0, 3, 1]
```

Replacing the variable by a value, the series will not be evaluated as an expression, but as a `SeriesData` object:

```
>> s = Series[Exp[x^2], {x, 0, 2}]  
1 +  $x^2 + O[x]^3$   
>> s /. x -> 4  
1 +  $4^2 + O[4]^3$ 
```

`Normal` transforms a `SeriesData` expression into a polynomial:

```
>> s // Normal  
1 +  $x^2$   
>> (s // Normal) /. x -> 4  
17  
>> Clear[s];
```

We can also expand over multiple variables

```
>> Series[Exp[x-y], {x, 0, 2}, {y, 0, 2}]  

$$\left(1 - y + \frac{1}{2}y^2 + O[y]^3\right) + \left(1 - y + \frac{1}{2}y^2 + O[y]^3\right)x + \left(\frac{1}{2} + \left(-\frac{1}{2}\right)y + \frac{1}{4}y^2 + O[y]^3\right)x^2 + O[x]^3$$

```

SeriesData

WMA link

```
SeriesData[...]  
Represents a series expansion.
```

Sum of two series:

```
>> Series[Cosh[x], {x, 0, 2}] + Series[Sinh[x], {x, 0, 3}]  
1 + x +  $\frac{1}{2}x^2 + O[x]^3$ 
```

```

>> Series[f[x],{x,0,2}] * g[w]

$$f[0]g[w] + g[w]f'[0]x + \frac{g[w]f''[0]}{2}x^2 + O[x]^3$$


```

The product of two series on the same neighbourhood of the same variable are multiplied

```
>> Series[Exp[-a x],{x,0,2}] * Series[Exp[-b x],{x,0,2}]
```

$$1 + (-a - b)x + \left(\frac{a^2}{2} + ab + \frac{b^2}{2}\right)x^2 + O[x]^3$$

```

>> D[Series[Exp[-a x],{x,0,2}],a]

$$-x + ax^2 + O[x]^3$$


```

Solve

Equation solving (SymPy, WMA)

```

Solve[equation, vars]
attempts to solve equation for the variables vars.
Solve[equation, vars, domain]
restricts variables to domain, which can be Complexes or Reals or Integers.

```

```

>> Solve[x ^ 2 - 3 x == 4, x]

$$\{x > -1\}, \{x > 4\}$$


>> Solve[4 y - 8 == 0, y]

$$\{y > 2\}$$


```

Apply the solution:

```

>> sol = Solve[2 x^2 - 10 x - 12 == 0, x]

$$\{x > -1\}, \{x > 6\}$$


>> x /. sol

$$\{-1, 6\}$$


```

Contradiction:

```

>> Solve[x + 1 == x, x]

$$\{\}$$


```

Tautology:

```

>> Solve[x ^ 2 == x ^ 2, x]

$$\{\{\}\}$$


```

Rational equations:

```

>> Solve[x / (x ^ 2 + 1) == 1, x]

$$\left\{\left\{x > \frac{1}{2} - \frac{i}{2}\sqrt{3}\right\}, \left\{x > \frac{1}{2} + \frac{i}{2}\sqrt{3}\right\}\right\}

>> Solve[(x^2 + 3 x + 2)/(4 x - 2) == 0, x]

$$\{x > -2\}, \{x > -1\}$$$$

```

Transcendental equations:

```
>> Solve[Cos[x] == 0, x]
{ {x -> π/2}, {x -> 3π/2} }
```

Solve can only solve equations with respect to symbols or functions:

```
>> Solve[f[x + y] == 3, f[x + y]]
{ {f[x + y] -> 3} }

>> Solve[a + b == 2, a + b]
a + b is not a valid variable.
Solve[a + b == 2, a + b]
```

This happens when solving with respect to an assigned symbol:

```
>> x = 3;

>> Solve[x == 2, x]
3 is not a valid variable.
Solve[False, 3]

>> Clear[x]

>> Solve[a < b, a]
a < b is not a well-formed equation.
Solve[a < b, a]
```

Solve a system of equations:

```
>> eqs = {3 x^2 - 3 y == 0, 3 y^2 - 3 x == 0};

>> sol = Solve[eqs, {x, y}] // Simplify
{ {x -> 0, y -> 0}, {x -> 1, y -> 1}, {x -> -1/2 + I/2 Sqrt[3], y
 -> -1/2 - I/2 Sqrt[3]}, {x -> -1/2 - I/2 Sqrt[3], y -> -1/2 + I/2 Sqrt[3]} }

>> eqs /. sol // Simplify
{ {True, True}, {True, True}, {True, True}, {True, True} }
```

Solve when given an underdetermined system:

```
>> Solve[x^2 == 1 && z^2 == -1, {x, y, z}]
Equations may not give solutions for all "solve" variables.
{ {x -> -1, z -> -I}, {x -> -1, z -> I}, {x -> 1, z -> -I}, {x -> 1, z -> I} }
```

Examples using specifying the Domain in solutions:

```
>> Solve[x^2 == -1, x, Reals]
{}

>> Solve[x^2 == 1, x, Reals]
{ {x -> -1}, {x -> 1} }

>> Solve[x^2 == -1, x, Complexes]
{ {x -> -I}, {x -> I} }
```

```
>> Solve[4 - 4 * x^2 - x^4 + x^6 == 0, x, Integers]
{{x > -1}, {x > 1}}
```

Differential Equations

Differential Equation

C

WMA link

$C[n]$

represents the n th constant in a solution to a differential equation.

DSolve

WMA link

$DSolve[eq, y[x], x]$

solves a differential equation for the function $y[x]$.

```
>> DSolve[y''[x] == 0, y[x], x]
{{y[x] -> xC[2] + C[1]}}

>> DSolve[y''[x] == y[x], y[x], x]
{{y[x] -> C[1]E^{-x} + C[2]E^x}}

>> DSolve[y''[x] == y[x], y, x]
{{y -> (Function[{x}, C[1]E^{-x} + C[2]E^x])}}
```

DSolve can also solve basic PDE

```
>> DSolve[D[f[x, y], x] / f[x, y] + 3 D[f[x, y], y] / f[x, y] == 2, f, {x, y}]
{{f -> (Function[{x, y}, E^{x/5 + 3y/5} C[1] [3x - y]])}

>> DSolve[D[f[x, y], x] x + D[f[x, y], y] y == 2, f[x, y], {x, y}]
{{f[x, y] -> 2Log[x] + C[1] [y/x]}

>> DSolve[D[y[x, t], t] + 2 D[y[x, t], x] == 0, y[x, t], {x, t}]
{{y[x, t] -> C[1] [x - 2t]}}
```

Exponential Functions

Exponential Functions

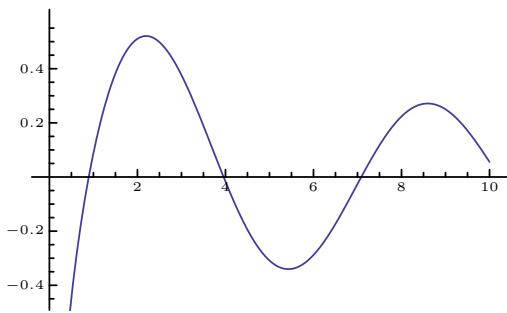
Numerical values and derivatives can be computed; however, most special exact values and simplification rules are not implemented yet.

Exp

WMA link

```
Exp[z]
    returns the exponential function of z.
```

```
>> Exp[1]
E
>> Exp[10.0]
22026.5
>> Exp[x] //FullForm
Power[E, x]
>> Plot[Exp[x], {x, 0, 3}]
```

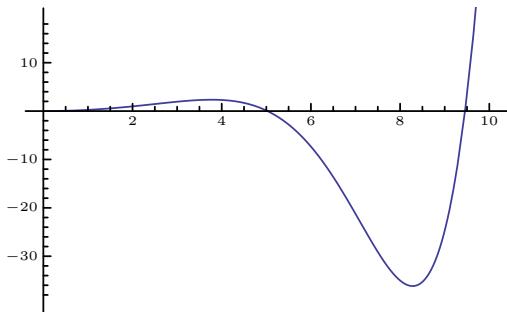


Log

WMA link

```
Log[z]
    returns the natural logarithm of z.
```

```
>> Log[{0, 1, E, E * E, E ^ 3, E ^ x}]
{-\infty, 0, 1, 2, 3, Log[E^x]}
>> Log[0.]
Indeterminate
>> Plot[Log[x], {x, 0, 5}]
```



Log10

WMA link

Log10[z]
returns the base-10 logarithm of z.

```
>> Log10[1000]
3
>> Log10[{2., 5.}]
{0.30103, 0.69897}
>> Log10[E ^ 3]

$$\frac{3}{\text{Log}[10]}$$

```

Log2

WMA link

Log2[z]
returns the base-2 logarithm of z.

```
>> Log2[4 ^ 8]
16
>> Log2[5.6]
2.48543
>> Log2[E ^ 2]

$$\frac{2}{\text{Log}[2]}$$

```

LogisticSigmoid

WMA link

LogisticSigmoid[z]
returns the logistic sigmoid of z.

```
>> LogisticSigmoid[0.5]
0.622459
>> LogisticSigmoid[0.5 + 2.3 I]
1.06475 + 0.808177I
>> LogisticSigmoid[{-0.2, 0.1, 0.3}]
{0.450166, 0.524979, 0.574443}
```

Hyperbolic Functions

Hyperbolic Functions

Hyperbolic functions are analogues of the ordinary trigonometric functions, but defined using the hyperbola rather than the circle.

Numerical values and derivatives can be computed; however, most special exact values and simplification rules are not implemented yet.

ArcCosh

Inverse hyperbolic cosine (SymPy, mpmath, WMA)

ArcCosh[z]

returns the inverse hyperbolic cosine of z.

```
>> ArcCosh[0]
```

$$\frac{I}{2}\pi$$

>> ArcCosh[0.]

$$0. + 1.5708I$$

1.57079632679489661923132169163975144211

ArcCoth

Inverse hyperbolic cotangent (SymPy, mpmath, WMA)

ArcCoth[z]

returns the inverse hyperbolic cotangent of z.

```
>> ArcCoth[0]
```

$$\frac{I}{2}\pi$$

```
>> ArcCoth[1]
```

∞

```
>> ArcCoth[0.0]
```

$$0. + 1.5708I$$

```
>> ArcCoth[0.5]
```

0.549306 –

ArcCsch

Inverse hyperbolic cosecant (SymPy, mpmath, WMA)

ArcCsch[z]

returns the inverse hyperbolic cosecant of z.

```
>> ArcCsch[0]
```

ComplexInfinity

```
>> ArcCsch[1.0]
0.881374
```

ArcSech

WMA link

ArcSech[z]
returns the inverse hyperbolic secant of z .

```
>> ArcSech[0]
∞
>> ArcSech[1]
0
>> ArcSech[0.5]
1.31696
```

ArcSinh

WMA link

ArcSinh[z]
returns the inverse hyperbolic sine of z .

```
>> ArcSinh[0]
0
>> ArcSinh[0.]
0.
>> ArcSinh[1.0]
0.881374
```

ArcTanh

WMA link

ArcTanh[z]
returns the inverse hyperbolic tangent of z .

```
>> ArcTanh[0]
>> ArcTanh[1]
∞
>> ArcTanh[0]
>> ArcTanh[.5 + 2 I]
0.0964156 + 1.12656I
>> ArcTanh[2 + I]
ArcTanh[2 + I]
```

Cosh

WMA link

Cosh[z]
returns the hyperbolic cosine of z .

```
>> Cosh[0]  
1
```

Coth

WMA link

Coth[z]
returns the hyperbolic cotangent of z .

```
>> Coth[0]  
ComplexInfinity
```

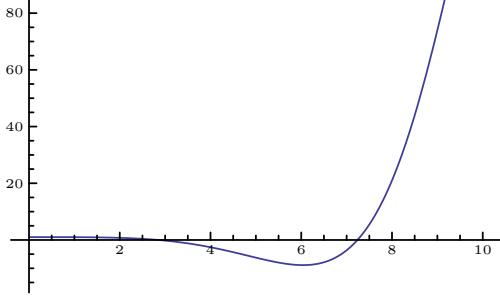
Gudermannian

Gudermannian function (WMA, MathWorld)

Gudermannian[z]
returns the Gudermannian function $gd(z)$.

```
>> Gudermannian[4.2]  
1.54081  
  
Gudermannian[-z] == -Gudermannian[z] :  
>> Gudermannian[-4.2] == -Gudermannian[4.2]  
True
```

```
>> Plot[Gudermannian[x], {x, -10, 10}]
```



InverseGudermannian

Inverse Gudermannian function (WMA, MathWorld)

InverseGudermannian[z]
returns the inverse Gudermannian function $gd^{-1}(z)$.

```

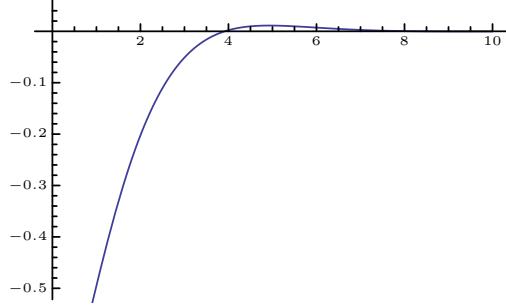
>> InverseGudermannian[.5]
0.522238

InverseGudermannian[-z] == -InverseGudermannian[z]:
>> InverseGudermannian[-.5] == -InverseGudermannian[.5]
True

InverseGudermannian is 0 at multiples of 8 Pi: = 0

```

```
>> Plot[InverseGudermannian[x], {x, -2 Pi, 2 Pi}]
```



Sech

WMA link

```
Sech[z]
returns the hyperbolic secant of z.
```

```

>> Sech[0]
1

```

Sinh

WMA link

```
Sinh[z]
returns the hyperbolic sine of z.
```

```

>> Sinh[0]
0

```

Tanh

WMA link

```
Tanh[z]
returns the hyperbolic tangent of z.
```

```

>> Tanh[0]
0

```

Integer Functions

Integer Function

BitLength

WMA link

`BitLength[x]`

gives the number of bits needed to represent the integer x . x 's sign is ignored.

```
>> BitLength[1023]
10
>> BitLength[100]
7
>> BitLength[-5]
3
>> BitLength[0]
0
```

Ceiling

WMA link

`Ceiling[x]`

gives the smallest integer greater than or equal to x .

```
>> Ceiling[1.2]
2
>> Ceiling[3/2]
2
```

For complex x , take the ceiling of real and imaginary parts.

```
>> Ceiling[1.3 + 0.7 I]
2 + I
```

DigitCount

WMA link

`DigitCount[n, b, d]`

returns the number of times digit d occurs in the base b representation of n .

`DigitCount[n, b]`

returns a list indicating the number of times each digit occurs in the base b representation of n .

`DigitCount[n, b]`

returns a list indicating the number of times each digit occurs in the decimal representation of n .

```

>> DigitCount[1022]
{1,2,0,0,0,0,0,0,0,1}

>> DigitCount[Floor[Pi * 10^100]]
{8,12,12,10,8,9,8,12,14,8}

>> DigitCount[1022, 2]
{9,1}

>> DigitCount[1022, 2, 1]
9

```

Floor

WMA link

Floor [<i>x</i>] <i>gives the greatest integer less than or equal to</i> <i>x</i> . Floor [<i>x</i> , <i>a</i>] <i>gives the greatest multiple of</i> <i>a</i> <i>less than or equal to</i> <i>x</i> .

```

>> Floor[10.4]
10

>> Floor[10/3]
3

>> Floor[10]
10

>> Floor[21, 2]
20

>> Floor[2.6, 0.5]
2.5

>> Floor[-10.4]
-11

```

For complex *x*, take the floor of real and imaginary parts.

```

>> Floor[1.5 + 2.7 I]
1 + 2I

```

For negative *a*, the smallest multiple of *a* greater than or equal to *x* is returned.

```

>> Floor[10.4, -1]
11

>> Floor[-10.4, -1]
-10

```

FromDigits

WMA link

FromDigits[*l*]
 returns the integer corresponding to the decimal representation given by *l*. *l* can be a list of digits or a string.

FromDigits[*l*, *b*]
 returns the integer corresponding to the base *b* representation given by *l*. *l* can be a list of digits or a string.

```
>> FromDigits["123"]
123

>> FromDigits[{1, 2, 3}]
123

>> FromDigits[{1, 0, 1}, 1000]
1000001
```

FromDigits can handle symbolic input:

```
>> FromDigits[{a, b, c}, 5]
c + 5 (5a + b)
```

Note that FromDigits does not automatically detect if you are providing a non-decimal representation:

```
>> FromDigits["a0"]
100

>> FromDigits["a0", 16]
160
```

FromDigits on empty lists or strings returns 0:

```
>> FromDigits[{}]
0

>> FromDigits[""]
0
```

IntegerDigits

WMA link

IntegerDigits[*n*]
 returns the decimal representation of integer *x* as list of digits. *x*'s sign is ignored.

IntegerDigits[*n*, *b*]
 returns the base *b* representation of integer *x* as list of digits. *x*'s sign is ignored.

IntegerDigits[*n*, *b*, *length*]
 returns a list of length *length*. If the number is too short, the list gets padded with 0 on the left.
 If the number is too long, the *length* least significant digits are returned.

```
>> IntegerDigits[76543]
{7,6,5,4,3}
```

The same thing specifying base 10 explicitly:

```
>> IntegerDigits[76543, 10]
{7,6,5,4,3}
```

The sign is discarded:

```
>> IntegerDigits[-76543]
{7,6,5,4,3}
```

Just the last 3 digits:

```
>> IntegerDigits[76543, 10, 3]
{5,4,3}
```

A geeky way to relate Christmas with Halloween is to note that Dec(imal) 25 is Oct(al) 31

```
>> IntegerDigits[25, 8]
{3,1}
```

IntegerReverse

WMA link

IntegerReverse[n]

returns the integer that has the reverse decimal representation of x without sign.

IntegerReverse[n, b]

returns the integer that has the reverse base b representation of x without sign.

```
>> IntegerReverse[1234]
4321

>> IntegerReverse[1022, 2]
511

>> IntegerReverse[-123]
321
```

IntegerString

WMA link

IntegerString[n]

returns the decimal representation of integer x as string. x 's sign is ignored.

IntegerString[n, b]

returns the base b representation of integer x as string. x 's sign is ignored.

IntegerString[n, b, length]

returns a string of length $length$. If the number is too short, the string gets padded with 0 on the left. If the number is too long, the $length$ least significant digits are returned.

For bases > 10, alphabetic characters a, b, ... are used to represent digits 11, 12, Note that base must be an integer in the range from 2 to 36.

```
>> IntegerString[12345]
12345

>> IntegerString[-500]
500

>> IntegerString[12345, 10, 8]
00012345
```

```

>> IntegerString[12345, 10, 3]
345
>> IntegerString[11, 2]
1011
>> IntegerString[123, 8]
173
>> IntegerString[32767, 16]
7fff
>> IntegerString[98765, 20]
c6i5

```

Linear algebra

Linear algebra

DesignMatrix

WMA link

```
DesignMatrix[m, f, x]
```

returns the design matrix for a linear model f in the variables x .

```

>> DesignMatrix[{{2, 1}, {3, 4}, {5, 3}, {7, 6}}, x, x]
{{1,2},{1,3},{1,5},{1,7}}
>> DesignMatrix[{{2, 1}, {3, 4}, {5, 3}, {7, 6}}, f[x], x]
{{1,f[2]},{1,f[3]},{1,f[5]},{1,f[7]}}

```

Det

Matrix Determinant (WMA link)

```
Det[m]
```

computes the determinant of the matrix m .

```

>> Det[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}]
-2

```

Symbolic determinant:

```

>> Det[{{a, b, c}, {d, e, f}, {g, h, i}}]
aei - afh - bdi + bfg + cdh - ceg

```

Eigensystem

Matrix Eigenvalues (WMA)

```
Eigensystem[m]
    returns the list {Eigenvalues[m], Eigenvectors[m]}.
```

```
>> Eigensystem[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}]
{{2, -1, 1}, {{1, 1, 1}, {1, -2, 1}, {-1, 0, 1}}}
```

Eigenvalues

Matrix Eigenvalues (WMA link)

```
Eigenvalues[m]
    computes the eigenvalues of the matrix m. By default Sympy's routine is used. Sometimes
    this is slow and less good than the corresponding mpmath routine. Use option Method-
    ->"mpmath" if you want to use mpmath's routine instead.
```

Numeric eigenvalues are sorted in order of decreasing absolute value:

```
>> Eigenvalues[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}]
{2, -1, 1}
```

Symbolic eigenvalues:

```
>> Eigenvalues[{{Cos[theta], Sin[theta], 0}, {-Sin[theta], Cos[theta],
    0}, {0, 0, 1}}] // Sort
{{1, Cos[theta] + Sqrt[(-1 + Cos[theta])(1 + Cos[theta])], Cos[
    theta] - Sqrt[(-1 + Cos[theta])(1 + Cos[theta])]}}

>> Eigenvalues[{{7, 1}, {-4, 3}}]
>> Eigenvalues[{{7, 1}, {-4, 3}}]
```

Eigenvectors

Matrix Eigenvalues (WMA link)

```
Eigenvectors[m]
    computes the eigenvectors of the matrix m.
```

```
>> Eigenvectors[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}]
{{1, 1, 1}, {1, -2, 1}, {-1, 0, 1}}

>> Eigenvectors[{{1, 0, 0}, {0, 1, 0}, {0, 0, 0}}]
{{0, 1, 0}, {1, 0, 0}, {0, 0, 1}}

>> Eigenvectors[{{2, 0, 0}, {0, -1, 0}, {0, 0, 0}}]
{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}

>> Eigenvectors[{{0.1, 0.2}, {0.8, 0.5}}]
{{-0.355518, -1.15048}, {-0.62896, 0.777438}}
```

FittedModel

WMA link

```
FittedModel[...]  
Result of a linear fit
```

Inverse

WMA link

```
Inverse[m]  
computes the inverse of the matrix m.
```

```
>> Inverse[{{1, 2, 0}, {2, 3, 0}, {3, 4, 1}}]  
{{-3, 2, 0}, {2, -1, 0}, {1, -2, 1}}  
  
>> Inverse[{{1, 0}, {0, 0}}]  
The matrix {{1, 0}, {0, 0}} is singular.  
Inverse[{{1, 0}, {0, 0}}]
```

LeastSquares

WMA link

```
LeastSquares[m, b]  
computes the least squares solution to  $m x = b$ , finding an  $x$  that solves for  $b$  optimally.
```

```
>> LeastSquares[{{1, 2}, {2, 3}, {5, 6}}, {1, 5, 3}]  
{-\frac{28}{13}, \frac{31}{13}}  
  
>> Simplify[LeastSquares[{{1, 2}, {2, 3}, {5, 6}}, {1, x, 3}]]  
\frac{12}{13} - \frac{8x}{13}, -\frac{4}{13} + \frac{7x}{13}  
  
>> LeastSquares[{{1, 1, 1}, {1, 1, 2}}, {1, 3}]  
Solving for underdetermined system not implemented.  
LeastSquares[{{1, 1, 1}, {1, 1, 2}}, {1, 3}]
```

LinearModelFit

WMA link

```
LinearModelFit[m, f, x]  
fits a linear model  $f$  in the variables  $x$  to the dataset  $m$ .
```

```
>> m = LinearModelFit[{{2, 1}, {3, 4}, {5, 3}, {7, 6}}, x, x];  
>> m["BasisFunctions"]
```

```

>> m["BestFit"]
0.186441 + 0.779661x

>> m["BestFitParameters"]
{0.186441, 0.779661}

>> m["DesignMatrix"]
{{1,2},{1,3},{1,5},{1,7} }

>> m["Function"]

>> m["Response"]
{1,4,3,6}

>> m["FitResiduals"]

>> m = LinearModelFit[{{2, 2, 1}, {3, 2, 4}, {5, 6, 3}, {7, 9, 6}}, {Sin[x], Cos[y]}, {x, y}];

>> m["BasisFunctions"]

>> m["Function"]

>> m = LinearModelFit[{{{1, 4}, {1, 5}, {1, 7}}, {1, 2, 3}}];

>> m["BasisFunctions"]

>> m["FitResiduals"]

```

LinearSolve

WMA link

```
LinearSolve[matrix, right]
  solves the linear equation system matrix . x = right and returns one corresponding solution x.
```

```
>> LinearSolve[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}, {1, 2, 3}]
{0,1,2}
```

Test the solution:

```
>> {{1, 1, 0}, {1, 0, 1}, {0, 1, 1}} . {0, 1, 2}
{1,2,3}
```

If there are several solutions, one arbitrary solution is returned:

```
>> LinearSolve[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, {1, 1, 1}]
{-1,1,0}
```

Infeasible systems are reported:

```
>> LinearSolve[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, {1, -2, 3}]
Linear equation encountered that has no solution.
LinearSolve[{{1,2,3},{4,5,6},{7,8,9}},{1,-2,3}]
```

MatrixExp

WMA link

```
MatrixExp[m]
computes the exponential of the matrix m.
```

```
>> MatrixExp[{{0, 2}, {0, 1}}]
{{1, -2 + 2E}, {0, E}}
>> MatrixExp[{{1.5, 0.5}, {0.5, 2.0}}]
{5.16266, 3.02952}, {3.02952, 8.19218}
```

MatrixPower

WMA link

```
MatrixPower[m, n]
computes the nth power of a matrix m.
```

```
>> MatrixPower[{{1, 2}, {1, 1}}, 10]
{{3363, 4756}, {2378, 3363}}
>> MatrixPower[{{1, 2}, {2, 5}}, -3]
{{169, -70}, {-70, 29}}
```

MatrixRank

WMA link

```
MatrixRank[matrix]
returns the rank of matrix.
```

```
>> MatrixRank[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
2
>> MatrixRank[{{1, 1, 0}, {1, 0, 1}, {0, 1, 1}}]
3
>> MatrixRank[{{a, b}, {3 a, 3 b}}]
1
```

NullSpace

Kernel (null space) (WMA link)

```
NullSpace[matrix]
returns a list of vectors that span the nullspace of matrix.
```

```
>> NullSpace[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
{{1, -2, 1}}
```

```

>> A = {{1, 1, 0}, {1, 0, 1}, {0, 1, 1}};
>> NullSpace[A]
{ }
>> MatrixRank[A]
3

```

Pseudoinverse

WMA link

Pseudoinverse[*m*]

computes the Moore-Penrose pseudoinverse of the matrix *m*. If *m* is invertible, the pseudoinverse equals the inverse.

```

>> Pseudoinverse[{{1, 2}, {2, 3}, {3, 4}}]
{{{-11/6, -1/3, 7/6}, {4/3, 1/3, -2/3}}}
>> Pseudoinverse[{{1, 2, 0}, {2, 3, 0}, {3, 4, 1}}]
{{{-3, 2, 0}, {2, -1, 0}, {1, -2, 1}}}
>> Pseudoinverse[{{1.0, 2.5}, {2.5, 1.0}}]
{{-0.190476, 0.47619}, {0.47619, -0.190476}}

```

QRDecomposition

QR Decomposition (WMA link)

QRDecomposition[*m*]

computes the QR decomposition of the matrix *m*.

```

>> QRDecomposition[{{1, 2}, {3, 4}, {5, 6}}]
{{{{{\sqrt{35}/35, 3\sqrt{35}/35, \sqrt{35}/7}}, {{13\sqrt{210}/210, 2\sqrt{210}/105, -\sqrt{210}/42}}}, {{{{\sqrt{35}, 44\sqrt{35}/35}}, {{0, 2\sqrt{210}/35}}}}}

```

RowReduce

WMA link

RowReduce[*matrix*]

returns the reduced row-echelon form of *matrix*.

```

>> RowReduce[{{1, 0, a}, {1, 1, b}}]
{{1, 0, a}, {0, 1, -a + b}}
>> RowReduce[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}] // MatrixForm
\left( \begin{array}{ccc} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{array} \right)

```

SingularValueDecomposition

Singular Value Decomposition (WMA link)

```
SingularValueDecomposition[m]
calculates the singular value decomposition for the matrix m.
```

SingularValueDecomposition returns u, s, w such that $m = u s v$, $uu=1$, $vv=1$, and s is diagonal.

```
>> SingularValueDecomposition[{{1.5, 2.0}, {2.5, 3.0}}]
{{{{0.538954, 0.842335}, {0.842335, -0.538954}}, {{4.63555, 0.}, {0., 0.107862}}}, {{{0.628678, 0.777666}, {-0.777666, 0.628678}}}}
```

Tr

Matrix trace (WMA link)

```
Tr[m]
computes the trace of the matrix m.
```

```
>> Tr[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
15
```

Symbolic trace:

```
>> Tr[{{a, b, c}, {d, e, f}, {g, h, i}}]
a + e + i
```

Mathematical Constants

Mathematical Constants

Numeric, Arithmetic, or Symbolic constants like Pi, E, or Infinity.

Catalan

Catalan's constant (SymPy, WMA)

```
Catalan
is Catalan's constant with numerical value  $\simeq 0.915966$ .
```

```
>> Catalan // N
0.915966
>> N[Catalan, 20]
0.91596559417721901505
```

ComplexInfinity

Complex Infinity is an infinite number in the complex plane whose complex argument is unknown or undefined. (SymPy, MathWorld, WMA)

ComplexInfinity
represents an infinite complex quantity of undetermined direction.

```
>> 1 / ComplexInfinity
0
>> ComplexInfinity * Infinity
ComplexInfinity
>> FullForm[ComplexInfinity]
DirectedInfinity[]
```

Degree

Degree (angle) (WMA)

Degree
is the number of radians in one degree. It has a numerical value of $\pi / 180$.

```
>> Cos[60 Degree]
1
2
Degree has the value of Pi / 180
>> Degree == Pi / 180
True
>> N[\[Degree]] == N[Degree]
True
```

E

Euler's number (SymPy, WMA)

E
is the constant e with numerical value $\simeq 2.71828$.

```
>> N[E]
2.71828
>> N[E, 50]
2.7182818284590452353602874713526624977572470937000
```

EulerGamma

Euler's constant (SymPy, WMA)

EulerGamma
is Euler's constant γ with numerical value $\simeq 0.577216$.

```
>> EulerGamma // N
0.577216
```

```
>> N[EulerGamma, 40]
0.5772156649015328606065120900824024310422
```

Glaisher

Glaisher–Kinkelin constant (mpmath, WMA)

Glaisher
is Glaisher's constant, with numerical value $\simeq 1.28243$.

```
>> N[Glaisher]
1.28243

>> N[Glaisher, 50]
1.2824271291006226368753425688697917277676889273250

# 1.2824271291006219541941391071304678916931152343750
```

GoldenRatio

Golden ratio (mpmath, WMA)

GoldenRatio
is the golden ratio, $\phi = (1+\text{Sqrt}[5])/2$.

```
>> GoldenRatio // N
1.61803

>> N[GoldenRatio, 40]
1.618033988749894848204586834365638117720
```

Indeterminate

Indeterminate form (SymPy, WMA)

Indeterminate
represents an indeterminate result.

```
>> 0^0
Indeterminate expression 0 ^ 0 encountered.

Indeterminate

>> Tan[Indeterminate]
Indeterminate
```

Infinity

Infinity (SymPy, WMA)

Infinity
a symbol that represents an infinite real quantity.

```

>> 1 / Infinity
0
>> Infinity + 100
∞

```

Use `Infinity` in sum and limit calculations:

```

>> Sum[1/x^2, {x, 1, Infinity}]

$$\frac{\pi^2}{6}$$


```

Khinchin

Khinchin's constant (mpmath, WMA)

`Khinchin`
is Khinchin's constant, with numerical value $\simeq 2.68545$.

```

>> N[Khinchin]
2.68545
>> N[Khinchin, 50]
2.6854520010653064453097148354817956938203822939945
# = 2.6854520010653075701156922150403261184692382812500

```

\$MaxMachineNumber

Largest normalizable machine number (WMA)

`$MaxMachineNumber`
Represents the largest positive number that can be represented as a normalized machine number in the system.

The product of `$MaxMachineNumber` and `$MinMachineNumber` is a constant:

```

>> $MaxMachineNumber * $MinMachineNumber
4.

```

\$MinMachineNumber

Smallest normalizable machine number (WMA)

`$MinMachineNumber`
Represents the smallest positive number that can be represented as a normalized machine number in the system.

`MachinePrecision` minus the `Log` base 10 of this number is the `Accuracy` of 0':

```

>> MachinePrecision -Log[10., $MinMachineNumber]==Accuracy[0']
True

```

Overflow

Numeric Overflow (WMA)

See also Integer Overflow.

Overflow[]
represents a number too large to be represented by Mathics.

```
>> Exp[10.*^20]
Overflow occurred in computation.
Overflow[]

>> Table[Exp[10.^k],{k, 3}]
Overflow occurred in computation.
{22026.5, 2.68812^43, Overflow[]}

>> 1 / Underflow[]
Overflow[]
```

Pi

Pi, π (SymPy, WMA)

Pi
is the constant π .

```
>> Pi
π

>> N[Pi]
3.14159
```

Pi to a numeric precision of 20 digits:

```
>> N[Pi, 20]
3.1415926535897932385
```

Note that the above is not the same thing as the number of digits *after* the decimal point. This may differ from similar concepts from other mathematical libraries, including those which Mathics uses!

Use numpy to compute Pi to 20 digits:

```
>> N[Pi, 20, Method -> "numpy"]
3.1415926535897930000
```

“sympy” is the default method.

```
>> Attributes[Pi]
{Constant, Protected, ReadProtected}
```

Undefined

Undefined symbol/value (WMA)

Undefined
a symbol that represents a quantity with no defined value.

```

>> ConditionalExpression[a, False]
Undefined
>> Attributes[Undefined]
{Protected}

```

Underflow

Arithmetic underflow (WMA)

Overflow[]
represents a number too small to be represented by Mathics.

```

>> 1 / Overflow[]
Underflow []
>> 5 * Underflow []
5Underflow []
>> % // N

```

Underflow[] is kept symbolic in operations against integer numbers, but taken as 0. in numeric evaluations:

```

>> 1 - Underflow []
1 - Underflow []
>> % // N

```

Number theoretic functions

Number theoretic function

ContinuedFraction

Continued fraction (SymPy, WMA)

ContinuedFraction[x, n]
generate the first n terms in the continued fraction representation of x .
ContinuedFraction[x]
the complete continued fraction representation for a rational or quadratic irrational number.

```

>> ContinuedFraction[Pi, 10]
{3, 7, 15, 1, 292, 1, 1, 1, 2, 1}
>> ContinuedFraction[(1 + 2 Sqrt[3])/5]
{0, 1, {8, 3, 34, 3}}
>> ContinuedFraction[Sqrt[70]]
{8, {2, 1, 2, 1, 2, 16}}

```

Divisors

WMA link

```
Divisors[n]
    returns a list of the integers that divide n.
```

```
>> Divisors[96]
{1,2,3,4,6,8,12,16,24,32,48,96}

>> Divisors[704]
{1,2,4,8,11,16,22,32,44,64,88,176,352,704}

>> Divisors[{87, 106, 202, 305}]
{{1,3,29,87},{1,2,53,106},{1,2,101,202},{1,5,61,305}}
```

EulerPhi

Euler's totient function (SymPy, WMA) This function counts positive integers up to *n* that are relatively prime to *n*. It is typically used in cryptography and in many applications in elementary number theory.

```
EulerPhi[n]
    returns the Euler totient function .
```

Compute the Euler totient function:

```
>> EulerPhi[9]
6
```

EulerPhi of a negative integer is same as its positive counterpart:

```
>> EulerPhi[-11] == EulerPhi[11]
True
```

Large arguments are computed quickly:

```
>> EulerPhi[40!]
1213437467632817072749054151808044236800000000000
```

EulerPhi threads over lists:

```
>> EulerPhi[Range[1, 17, 2]]
{1,2,4,6,10,12,8,16}
```

Above, we get consecutive even numbers when the input is prime.

Compare the results above with:

```
>> EulerPhi[Range[1, 17]]
{1,1,2,2,4,2,6,4,6,4,10,4,12,6,8,8,16}
```

FactorInteger

WMA link

```
FactorInteger[n]
    returns the factorization of n as a list of factors and exponents.
```

```
>> factors = FactorInteger[2010]
{{2,1},{3,1},{5,1},{67,1}}
```

To get back the original number:

```
>> Times @@ Power @@@ factors
2010
```

FactorInteger factors rationals using negative exponents:

```
>> FactorInteger[2010 / 2011]
{{2,1},{3,1},{5,1},{67,1},{2011,-1}}
```

FractionalPart

WMA link

```
FractionalPart[n]
finds the fractional part of n.
```

```
>> FractionalPart[4.1]
0.1
>> FractionalPart[-5.25]
-0.25
```

FromContinuedFraction

WMA link

```
FromContinuedFraction[list]
reconstructs a number from the list of its continued fraction terms.
```

```
>> FromContinuedFraction[{3, 7, 15, 1, 292, 1, 1, 1, 2, 1}]
1146408
—————
364913
>> FromContinuedFraction[Range[5]]
225
———
157
```

MantissaExponent

WMA link

```
MantissaExponent[n]
finds a list containing the mantissa and exponent of a given number n.
MantissaExponent[n, b]
finds the base b mantissa and exponent of n.
```

```
>> MantissaExponent[2.5*10^20]
{0.25,21}
```

```

>> MantissaExponent[125.24]
{0.12524,3}

>> MantissaExponent[125., 2]
{0.976563,7}

>> MantissaExponent[10, b]
MantissaExponent[10, b]

```

NextPrime

WMA link

```

NextPrime[n]
gives the next prime after n.

NextPrime[n,k]
gives the kth prime after n.

```

```

>> NextPrime[10000]
10007

>> NextPrime[100, -5]
73

>> NextPrime[10, -5]
-2

>> NextPrime[100, 5]
113

>> NextPrime[5.5, 100]
563

>> NextPrime[5, 10.5]
NextPrime[5,10.5]

```

PartitionsP

WMA link

```

PartitionsP[n]
return the number  $p(n)$  of unrestricted partitions of the integer  $n$ .

```

```

>> Table[PartitionsP[k], {k, -2, 12}]
{0,0,1,1,2,3,5,7,11,15,22,30,42,56,77}

```

Prime

WMA link

```
Prime[n]
Prime[{n0, n1, ...}]
```

returns the n th prime number where n is a positive Integer. If given a list of integers, the return value is a list with Prime applied to each.

Note that the first prime is 2, not 1:

```
>> Prime[1]
2
>> Prime[167]
991
```

When given a list of integers, a list is returned:

```
>> Prime[{5, 10, 15}]
{11, 29, 47}
```

1.2 isn't an integer

```
>> Prime[1.2]
Prime[1.2]
```

Since 0 is less than 1, like 1.2 it is invalid.

```
>> Prime[{0, 1, 1.2, 3}]
{Prime[0], 2, Prime[1.2], 5}
```

PrimePi

Prime numbers

```
PrimePi[x]
```

gives the number of primes less than or equal to x .

PrimePi is the inverse of Prime:

```
>> PrimePi[2]
1
>> PrimePi[100]
25
>> PrimePi[-1]
0
>> PrimePi[3.5]
2
>> PrimePi[E]
1
```

PrimePowerQ

Prime numbers

```
PrimePowerQ[n]
```

returns True if n is a power of a prime number.

```

>> PrimePowerQ[9]
True

>> PrimePowerQ[52142]
False

>> PrimePowerQ[-8]
True

>> PrimePowerQ[371293]
True

```

RandomPrime

Prime numbers

```

RandomPrime[{imin, $imax}]
  gives a random prime between imin and imax.
RandomPrime[imax]
  gives a random prime between 2 and imax.
RandomPrime[range, n]
  gives a list of n random primes in range.

```

```

>> RandomPrime[{14, 17}]
17

>> RandomPrime[{14, 16}, 1]
There are no primes in the specified interval.
RandomPrime[{14, 16}, 1]

>> RandomPrime[{8, 12}, 3]
{11, 11, 11}

>> RandomPrime[{10, 30}, {2, 5}]
{{29, 29, 29, 29}, {29, 29, 29, 29}}

```

Random number generation

Random number generation

Random numbers are generated using the Mersenne Twister.

Random

WMA

```

Random[]
  gives a uniformly distributed pseudorandom Real number in the range 0 to 1.
Random[type, range]
  gives a uniformly distributed pseudorandom number of the type type, in the specified interval
  range. Possible types are Integer, Real or Complex.

```

Legacy

function. Superseded by RandomReal, RandomInteger and RandomComplex.

RandomChoice

WMA

```
RandomChoice[items]
  randomly picks one item from items.
RandomChoice[items, n]
  randomly picks n items from items. Each pick in the n picks happens from the given set of items, so each item can be picked any number of times.
RandomChoice[items, {n1, n2, ...}]
  randomly picks items from items and arranges the picked items in the nested list structure described by {n1, n2, ...}.
RandomChoice[weights -> items, n]
  randomly picks n items from items and uses the corresponding numeric values in weights to determine how probable it is for each item in items to get picked (in the long run, items with higher weights will get picked more often than ones with lower weight).
RandomChoice[weights -> items]
  randomly picks one items from items using weights weights.
RandomChoice[weights -> items, {n1, n2, ...}]
  randomly picks a structured list of items from items using weights weights.
```

Note: SeedRandom is used below so we get repeatable “random” numbers that we can test.

```
>> SeedRandom[42]

>> RandomChoice[{a, b, c}]
{c}

>> SeedRandom[42] (* Set for repeatable randomness *)

>> RandomChoice[{a, b, c}, 20]
{c, a, c, c, a, a, c, b, c, c, c, a, c, b, a, b, b, b, b}

>> SeedRandom[42]

>> RandomChoice[{"a", {1, 2}, x, {}}, 10]
{x, {}, a, x, x, {}, a, a, x, {1, 2} }

>> SeedRandom[42]

>> RandomChoice[{a, b, c}, {5, 2}]
{{c, a}, {c, c}, {a, a}, {c, b}, {c, c} }

>> SeedRandom[42]

>> RandomChoice[{1, 100, 5} -> {a, b, c}, 20]
{b, b, b, b, b, b, b, b, b, c, b, b, b, b, b, b, b, b}
```

RandomComplex

WMA)

```

RandomComplex[{z_min, z_max}]
    yields a pseudorandom complex number in the rectangle with complex corners z_min and z_max.
RandomComplex[z_max]
    yields a pseudorandom complex number in the rectangle with corners at the origin and at z_max.
RandomComplex[]
    yields a pseudorandom complex number with real and imaginary parts from 0 to 1.
RandomComplex[range, n]
    gives a list of n pseudorandom complex numbers.
RandomComplex[range, {n1, n2, ...}]
    gives a nested list of pseudorandom complex numbers.

```

```

>> RandomComplex[]
0.734271 + 0.391106I

>> RandomComplex[{1+I, 5+5I}]
1.33561 + 4.18874I

>> RandomComplex[1+I, 5]
{0.519712 + 0.663766I, 0.53529 + 0.903976I, 0.824985
 + 0.659807I, 0.566475 + 0.175172I, 0.749211 + 0.748907I}

>> RandomComplex[{1+I, 2+2I}, {2, 2}]
{{1.65438 + 1.66021I, 1.68615 + 1.3922I}, {1.94362 + 1.7918I, 1.20313 + 1.14307I}}

```

RandomInteger

WMA)

```

RandomInteger[{min, max}]
    yields a pseudorandom integer in the range from min to max inclusive.
RandomInteger[max]
    yields a pseudorandom integer in the range from 0 to max inclusive.
RandomInteger[]
    gives 0 or 1.
RandomInteger[range, n]
    gives a list of n pseudorandom integers.
RandomInteger[range, {n1, n2, ...}]
    gives a nested list of pseudorandom integers.

```

```

>> RandomInteger[{1, 5}]
2

>> RandomInteger[100, {2, 3}] // TableForm
15 30 69
7 41 72

```

Calling **RandomInteger** changes \$RandomState:

```

>> previousState = $RandomState;

>> RandomInteger[]
0

```

```
>> $RandomState != previousState
True
```

RandomReal

WMA)

```
RandomReal [{min, max}]
yields a pseudorandom real number in the range from min to max.
RandomReal [max]
yields a pseudorandom real number in the range from 0 to max.
RandomReal []
yields a pseudorandom real number in the range from 0 to 1.
RandomReal [range, n]
gives a list of n pseudorandom real numbers.
RandomReal [range, {n1, n2, ...}]
gives an n1 x n2 array of pseudorandom real numbers.
```

```
>> RandomReal []
0.579255
>> RandomReal[{1, 5}]
4.14801
```

RandomSample

WMA

```
RandomSample [items]
randomly picks one item from items.
RandomSample [items, n]
randomly picks n items from items. Each pick in the n picks happens after the previous items picked have been removed from items, so each item can be picked at most once.
RandomSample [items, {n1, n2, ...}]
randomly picks items from items and arranges the picked items in the nested list structure described by {n1, n2, ...}. Each item gets picked at most once.
RandomSample [weights -> items, n]
randomly picks n items from items and uses the corresponding numeric values in weights to determine how probable it is for each item in items to get picked (in the long run, items with higher weights will get picked more often than ones with lower weight). Each item gets picked at most once.
RandomSample [weights -> items]
randomly picks one items from items using weights weights. Each item gets picked at most once.
RandomSample [weights -> items, {n1, n2, ...}]
randomly picks a structured list of items from items using weights weights. Each item gets picked at most once.
```

```
>> SeedRandom[42]
>> RandomSample[{a, b, c, d}]
{b, d, a, c}
```

```

>> SeedRandom[42]

>> RandomSample[{a, b, c, d, e, f, g, h}, 7]
{b,f,a,h,c,e,d}

>> SeedRandom[42]

>> RandomSample[{"a", {1, 2}, x, {}}, 3]
{{1,2},{},{},a}

>> SeedRandom[42]

>> RandomSample[Range[10]]
{9,2,6,1,8,3,10,5,4,7}

>> SeedRandom[42]

>> RandomSample[Range[100], {2, 3}]
{{84,54,71},{46,45,40} }

>> SeedRandom[42]

>> RandomSample[Range[100] -> Range[100], 5]
{62,98,86,78,40}

```

\$RandomState

WMA

`$RandomState`

is a long number representing the internal state of the pseudo-random number generator.

```

>> Mod[$RandomState, 10^100]
7151709537032956679736318598511784252645280944999536205629334077952165935975232623744815787081503790

>> IntegerLength[$RandomState]
6440

```

So far, it is not possible to assign values to `$RandomState`.

```

>> $RandomState = 42
It is not possible to change the random state.

42

```

Not even to its own value:

```

>> $RandomState = $RandomState;
It is not possible to change the random state.

```

SeedRandom

WMA)

```
SeedRandom[n]
    resets the pseudorandom generator with seed n.
SeedRandom[]
    uses the current date and time as the seed.
```

SeedRandom can be used to get reproducible random numbers:

```
>> SeedRandom[42]

>> RandomInteger[100]

>> RandomInteger[100]

>> SeedRandom[42]

>> RandomInteger[100]

>> RandomInteger[100]
```

String seeds are supported as well:

```
>> SeedRandom["Mathics"]

>> RandomInteger[100]
```

Calling **SeedRandom** without arguments will seed the random number generator to a random state:

```
>> SeedRandom []

>> RandomInteger[100]
```

Trigonometric Functions

Trigonometric Functions

Numerical values and derivatives can be computed; however, most special exact values and simplification rules are not implemented yet.

AnglePath

WMA link

```
AnglePath[phi1, phi2, ...]
    returns the points formed by a turtle starting at {0, 0} and angled at 0 degrees going through
    the turns given by angles phi1, phi2, ... and using distance 1 for each step.
AnglePath[{{r1, phi1}, {r2, phi2}, ...}]
    instead of using 1 as distance, use r1, r2, ... as distances for the respective steps.
AnglePath[phi0, {phi1, phi2, ...}]
    starts with direction phi0 instead of 0.
AnglePath[{x, y}, {phi1, phi2, ...}]
    starts at {$x, $y} instead of {0, 0}.
AnglePath[{{x, y}, phi0}, {phi1, phi2, ...}]
    specifies initial position {x, y} and initial direction phi0.
AnglePath[{{x, y}, {dx, dy}}, {phi1, phi2, ...}]
    specifies initial position {x, y} and a slope {dx, dy} that is understood to be the initial direction
    of the turtle.
```

```

>> AnglePath[{90 Degree, 90 Degree, 90 Degree, 90 Degree}]
{{0,0},{0,1},{-1,1},{-1,0},{0,0}]

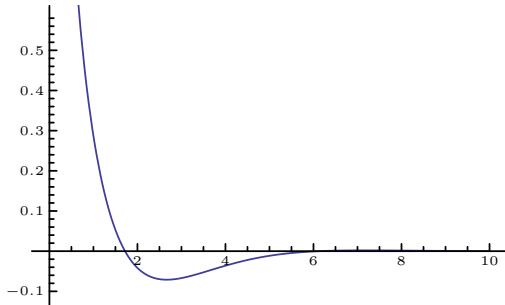
>> AnglePath[{{1, 1}, 90 Degree}, {{1, 90 Degree}, {2, 90 Degree}, {1,
90 Degree}, {2, 90 Degree}}]
{{1,1},{0,1},{0,-1},{1,-1},{1,1}]

>> AnglePath[{a, b}]
{{0,0},{Cos[a],Sin[a]},{Cos[a]+Cos[a+b],Sin[a]+Sin[a+b]}}

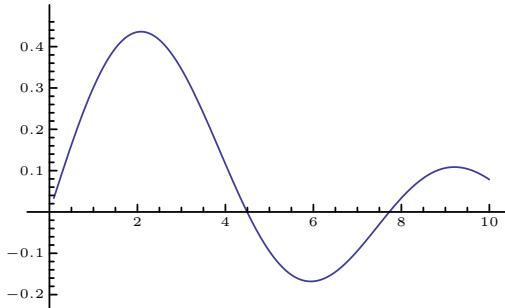
>> Precision[Part[AnglePath[{N[1/3, 100], N[2/3, 100]}], 2, 1]]
100.

>> Graphics[Line[AnglePath[Table[1.7, {50}]]]]

```



```
>> Graphics[Line[AnglePath[RandomReal[{-1, 1}, {100}]]]]
```



ArcCos

Inverse cosine, arccosine (SymPy, mpmath, WMA)

```
ArcCos[z]
returns the inverse cosine of z.
```

```

>> ArcCos[1]
0

>> ArcCos[0]
π
—
2

>> Integrate[ArcCos[x], {x, -1, 1}]
π

```

ArcCot

Inverse cotangent, arccotangent (SymPy, mpmath, WMA)

```
ArcCot[z]  
    returns the inverse cotangent of z.
```

```
>> ArcCot[0]  
    π  
    --  
    2
```

```
>> ArcCot[1]  
    π  
    --  
    4
```

ArcCsc

Inverse cosecant, arccosecant (SymPy, mpmath, WMA)

```
ArcCsc[z]  
    returns the inverse cosecant of z.
```

```
>> ArcCsc[1]  
    π  
    --  
    2
```

```
>> ArcCsc[-1]  
    -π  
    --  
    2
```

ArcSec

Inverse secant, arcsecant (SymPy, mpmath, WMA)

```
ArcSec[z]  
    returns the inverse secant of z.
```

```
>> ArcSec[1]  
    0
```

```
>> ArcSec[-1]  
    π
```

ArcSin

Inverse sine, arcsine (SymPy, mpmath, WMA)

```
ArcSin[z]  
    returns the inverse sine of z.
```

```
>> ArcSin[0]  
    0
```

```
>> ArcSin[1]
 $\frac{\pi}{2}$ 
```

ArcTan

Inverse tangent, arctangent (SymPy, mpmath, WMA)

```
ArcTan[z]
returns the inverse tangent of z.
```

```
>> ArcTan[1]
 $\frac{\pi}{4}$ 
>> ArcTan[1.0]
0.785398
>> ArcTan[-1.0]
-0.785398
>> ArcTan[1, 1]
 $\frac{\pi}{4}$ 
```

Cos

Cosine (SymPy, mpmath, WMA)

```
Cos[z]
returns the cosine of z.
```

```
>> Cos[3 Pi]
-1
```

Cot

Cotangent (SymPy, mpmath, WMA)

```
Cot[z]
returns the cotangent of z.
```

```
>> Cot[0]
ComplexInfinity
>> Cot[1.]
0.642093
```

Csc

Cosecant (SymPy, mpmath, WMA)

Csc[z]
returns the cosecant of z .

```
>> Csc[0]
ComplexInfinity

>> Csc[1] (* Csc[1] in Mathematica *)
      1
      Sin[1]

>> Csc[1.]
1.1884
```

Haversine

WMA link

Haversine[z]
returns the haversine function of z .

```
>> Haversine[1.5]
0.464631

>> Haversine[0.5 + 2I]
-1.15082 + 0.869405I
```

InverseHaversine

WMA link

InverseHaversine[z]
returns the inverse haversine function of z .

```
>> InverseHaversine[0.5]
1.5708

>> InverseHaversine[1 + 2.5 I]
1.76459 + 2.33097I
```

Sec

Secant (SymPy, mpmath, WMA)

Sec[z]
returns the secant of z .

```
>> Sec[0]
1

>> Sec[1] (* Sec[1] in Mathematica *)
      1
      Cos[1]
```

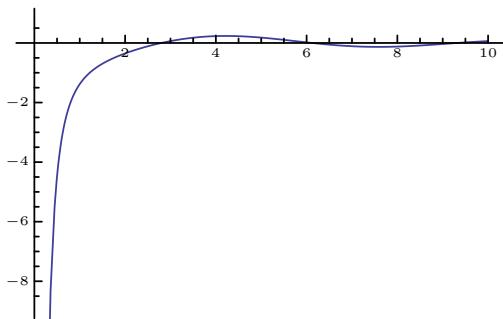
```
>> Sec[1.]  
1.85082
```

Sin

Sine (SymPy, mpmath, WMA)

```
Sin[z]  
returns the sine of z.
```

```
>> Sin[0]  
0  
>> Sin[0.5]  
0.479426  
>> Sin[3 Pi]  
0  
>> Sin[1.0 + I]  
1.29846 + 0.634964I  
>> Plot[Sin[x], {x, -Pi, Pi}]
```



Tan

Tangent (SymPy, mpmath, WMA)

```
Tan[z]  
returns the tangent of z.
```

```
>> Tan[0]  
0  
>> Tan[Pi / 2]  
ComplexInfinity
```

31. Interactive Manipulation

32. Layout

This module contains symbols used to define the high level layout for expression formatting. For instance, to represent a set of consecutive expressions in a row, we can use `Row`.

Contents

<code>Center</code>	420	<code>MakeBoxes</code>	422	<code>Row</code>	424
<code>Format</code>	421	<code>NonAssociative</code>	423	<code>Style</code>	425
<code>Grid</code>	421	<code>Postfix (//)</code>	423	<code>Subscript</code>	425
<code>Infix</code>	422	<code>Precedence</code>	423	<code>Subsuperscript</code>	425
<code>Left</code>	422	<code>Prefix (@)</code>	424	<code>Superscript</code>	426
		<code>Right</code>	424		

Center

WMA link

Center

is used with the `ColumnAlignments` option to `Grid` or `TableForm` to specify a centered column.

Format

WMA link

Format [*expr*]

holds values specifying how *expr* should be printed.

Assign values to `Format` to control how particular expressions should be formatted when printed to the user.

```
>> Format[f[x___]] := Infix[{x}, "~"]

>> f[1, 2, 3]
1 ~ 2 ~ 3

>> f[1]
1
```

Raw objects cannot be formatted:

```
>> Format[3] = "three";
      Cannot assign to raw object 3.
```

Format types must be symbols:

```
>> Format[r, a + b] = "r";
      Format type a + b is not a symbol.
```

Formats must be attached to the head of an expression:

```
>> f /: Format[g[f]] = "my f";
   Tag f not found or too deep for an assigned rule.
```

Grid

WMA link

```
Grid[{{a1, a2, ...}, {b1, b2, ...}, ...}]
formats several expressions inside a GridBox.
```

```
>> Grid[{{a, b}, {c, d}}]
a  b
c  d
```

For shallow lists, elements are shown as a column:

```
>> Grid[{a, b, c}]
a
b
c
```

If the sublists have different sizes, the grid has the number of columns of the largest one. Incomplete rows are completed with empty strings:

```
>> Grid[{{"first", "second", "third"}, {a}, {1, 2, 3}}]
first  second  third
      a
      1      2      3
```

If the list is a mixture of lists and other expressions, the non-list expressions are shown as rows:

```
>> Grid[{"This is a long title", {"first", "second", "third"}, {a}, {1, 2, 3}]}
This is a long title
first  second  third
      a
      1      2      3
```

Infix

WMA link

```
Infix[expr, oper, prec, assoc]
displays expr with the infix operator oper, with precedence prec and associativity assoc.
```

Infix can be used with Format to display certain forms with user-defined infix notation:

```
>> Format[g[x_, y_]] := Infix[{x, y}, "#", 350, Left]
>> g[a, g[b, c]]
a#(b#c)
>> g[g[a, b], c]
a#b#c
```

```

>> g[a + b, c]
(a + b) #c

>> g[a * b, c]
ab#c

>> g[a, b] + c
c + a#b

>> g[a, b] * c
c (a#b)

>> Infix[{a, b, c}, {"+", "-"}]
a + b - c

```

Left

WMA link

Left

is used with operator formatting constructs to specify a left-associative operator.

MakeBoxes

WMA link

MakeBoxes [expr]

is a low-level formatting primitive that converts *expr* to box form, without evaluating it.
 $\backslash(x \dots \backslash)$
directly inputs box objects.

String representation of boxes

```

>> \(\(x \^ 2\)
SuperscriptBox[x, 2]

>> \(\(x \_ 2\)
SubscriptBox[x, 2]

>> \(\(a \+ b \% c\)
UnderoverscriptBox[a, b, c]

>> \(\(a \& b \% c\)
UnderoverscriptBox[a, c, b]

>> \(\(x \& y \)
OverscriptBox[x, y]

>> \(\(x \+ y \)
UnderscriptBox[x, y]

```

NonAssociative

on, logic, comparison, datentime, attributes and binary)

NonAssociative

is used with operator formatting constructs to specify a non-associative operator.

Postfix (//)

WMA link

$x // f$

is equivalent to $f[x]$.

```
>> b // a  
a[b]
```

```
>> c // b // a  
a[b[c]]
```

The postfix operator // is parsed to an expression before evaluation:

```
>> Hold[x // a // b // c // d // e // f]  
Hold[f[e[d[c[b[a[x]]]]]]]
```

Precedence

on, logic, comparison, datentime, attributes and binary)

Precedence [*op*]

returns the precedence of the built-in operator *op*.

```
>> Precedence[Plus]  
310.  
  
>> Precedence[Plus] < Precedence[Times]  
True
```

Unknown symbols have precedence 670:

```
>> Precedence[f]  
670.
```

Other expressions have precedence 1000:

```
>> Precedence[a + b]  
1000.
```

Prefix (@)

WMA link

$f @ x$

is equivalent to $f[x]$.

```

>> a @ b
a[b]

>> a @ b @ c
a[b[c]]

>> Format[p[x_]] := Prefix[{x}, "*"]

>> p[3]
*3

>> Format[q[x_]] := Prefix[{x}, "~", 350]

>> q[a+b]
~ (a + b)

>> q[a*b]
~ ab

>> q[a]+b
b+ ~ a

```

The prefix operator `@` is parsed to an expression before evaluation:

```

>> Hold[a @ b @ c @ d @ e @ f @ x]
Hold[a[b[c[d[e[f[x]]]]]]]

```

Right

WMA link

Right

is used with operator formatting constructs to specify a right-associative operator.

Row

WMA link

Row[*{expr, ...}*]

formats several expressions inside a RowBox.

Style

WMA link

Style[*expr*, *options*]
 displays *expr* formatted using the specified option settings.

Style[*expr*, ‘‘style’’]
 uses the option settings for the specified style in the current notebook.

Style[*expr*, *color*]
 displays using the specified color.

Style[*expr*, *Bold*]
 displays with fonts made bold.

Style[*expr*, *Italic*]
 displays with fonts made italic.

Style[*expr*, *Underlined*]
 displays with fonts underlined.

'Style[*expr*, *Larger*]
 displays with fonts made larger.

Style[*expr*, *Smaller*]
 displays with fonts made smaller.

Style[*expr*, *n*]
 displays with font size *n*.

Style[*expr*, *Tiny*]
Style[*expr*, *Small*], etc.
 display with fonts that are tiny, small, etc.

Subscript

WMA link

Subscript[*a*, *i*]
 displays as *a*_*i*.

```
>> Subscript[x,1,2,3] // TeXForm
x_{1,2,3}
```

Subsuperscript

WMA link

Subsuperscript[*a*, *b*, *c*]
 displays as \$*a*_*b*^{*c*}\$.

```
>> Subsuperscript[a, b, c] // TeXForm
a_b^c
```

Superscript

WMA link

Superscript[*x*, *y*]
 displays as *x*^{*y*}.

```
>> Superscript[x,3] // TeXForm  
x^3
```

33. List Functions

Generalized Lists make up a core part of Mathics. In fact, to first approximation Evaluation works on a special kind of List called an M-Expression.

As a result, there about a hundred list functions.

Contents

Associations	427	Count	436	TakeSmallestBy . . .	448
Association	428	Delete	437	Predicates on Lists . . .	448
AssociationQ	428	DeleteCases	438	ContainsOnly	448
Key	428	Drop	438	Rearranging and Restructuring Lists 449	
Keys	429	Extract	438	Catenate	449
Lookup	429	First	439	Complement	449
Missing	429	FirstCase	439	DeleteDuplicates . . .	450
RowBox	429	FirstPosition	440	Flatten	450
Values	430	Insert	440	Gather	450
Constructing Lists . . .	430	Last	440	GatherBy	451
Array	430	Length	441	Intersection	451
ConstantArray . . .	431	Most	441	Join	452
List	431	Part	443	PadLeft	452
Normal	431	Pick	443	PadRight	453
Permutations	432	Position	444	Partition	453
Range	432	Prepend	444	Reverse	454
Reap	433	PrependTo	445	Riffle	454
RowBox	433	ReplacePart	445	RotateLeft	455
Sow	433	Rest	446	RotateRight	455
Table	434	RowBox	446	Split	456
Tuples	434	Select	446	SplitBy	456
Elements of Lists . . .	434	Span (;;)	447	Tally	456
Append	435	Take	447	Union	457
AppendTo	435	UpTo	447		
Cases	436	Math & Counting Operations on Lists 447			
		TakeLargestBy . . .	448		

Associations

Associations

An Association maps keys to values and is similar to a dictionary in Python; it is often sparse in that their key space is much larger than the number of actual keys found in the collection.

Association

WMA link

```
Association[key1 -> val1, key2 -> val2, ...]
<|key1 -> val1, key2 -> val2, ...|>
represents an association between keys and values.
```

Association is the head of associations:

```
>> Head[<|a -> x, b -> y, c -> z|>]
```

Association

```
>> <|a -> x, b -> y|>
```

```
<|a -> x, b -> y|>
```

```
>> Association[{a -> x, b -> y}]
```

```
<|a -> x, b -> y|>
```

Associations can be nested:

```
>> <|a -> x, b -> y, <|a -> z, d -> t|>|>
```

```
<|a -> z, b -> y, d -> t|>
```

AssociationQ

WMA link

```
AssociationQ[expr]
```

return True if *expr* is a valid Association object, and False otherwise.

```
>> AssociationQ[<|a -> 1, b :> 2|>]
```

True

```
>> AssociationQ[<|a, b|>]
```

False

Key

WMA link

```
Key[key]
```

represents a key used to access a value in an association.

```
Key[key][assoc]
```

Keys

WMA link

```
Keys[<|key1 -> val1, key2 -> val2, ...|>]
```

return a list of the keys *key*_{*i*} in an association.

```
Keys[{key1 -> val1, key2 -> val2, ...}]
```

return a list of the *key*_{*i*} in a list of rules.

```
>> Keys[<|a -> x, b -> y|>]
```

```
{a, b}
```

```
>> Keys[{a -> x, b -> y}]  
{a, b}
```

Keys automatically threads over lists:

```
>> Keys[{{<|a -> x, b -> y|>, {w -> z, {}}}}]  
{ {a, b}, {w, {}} }
```

Keys are listed in the order of their appearance:

```
>> Keys[{c -> z, b -> y, a -> x}]  
{c, b, a}
```

Lookup

WMA link

Lookup[*assoc, key*]

looks up the value associated with *key* in the association *assoc*, or **Missing**[*KeyAbsent*].

Missing

WMA link

Missing[]

represents a data that is missing.

```
>> ElementData["Meitnerium","MeltingPoint"]  
Missing[NotAvailable]
```

RowBox

WMA link

RowBox[{...}]

is a box construct that represents a sequence of boxes arranged in a horizontal row.

Values

WMA link

Values[<|*key1* -> *val1*, *key2* -> *val2*, ...|>]

return a list of the values *vali* in an association.

Values[{*key1* -> *val1*, *key2* -> *val2*, ...}]

return a list of the *vali* in a list of rules.

```
>> Values[<|a -> x, b -> y|>]  
{x, y}  
  
>> Values[{a -> x, b -> y}]  
{x, y}
```

Values automatically threads over lists:

```
>> Values[{\<|a -> x, b -> y|>, {c -> z, {}}}]  
{{x,y},{z,{}}}
```

Values are listed in the order of their appearance:

```
>> Values[{c -> z, b -> y, a -> x}]  
{z,y,x}
```

Constructing Lists

Constructing Lists

Functions for constructing lists of various sizes and structure.

See also Constructing Vectors.

Array

WMA link

```
Array[f, n]  
    returns the  $n$ -element list  $\{f[1], \dots, f[n]\}$ .  
Array[f, n, a]  
    returns the  $n$ -element list  $\{f[a], \dots, f[a + n]\}$ .  
Array[f, {n, m}, {a, b}]  
    returns an  $n$ -by- $m$  matrix created by applying  $f$  to indices ranging from  $(a, b)$  to  $(a + n, b + m)$ .  
Array[f, dims, origins, h]  
    returns an expression with the specified dimensions and index origins, with head  $h$  (instead of List).
```

```
>> Array[f, 4]  
{f[1],f[2],f[3],f[4]}  
>> Array[f, {2, 3}]  
{\{f[1,1],f[1,2],f[1,3]\},\{f[2,1],f[2,2],f[2,3]\}}  
>> Array[f, {2, 3}, 3]  
{\{f[3,3],f[3,4],f[3,5]\},\{f[4,3],f[4,4],f[4,5]\}}  
>> Array[f, {2, 3}, {4, 6}]  
{\{f[4,6],f[4,7],f[4,8]\},\{f[5,6],f[5,7],f[5,8]\}}  
>> Array[f, {2, 3}, 1, Plus]  
f[1,1]+f[1,2]+f[1,3]+f[2,1]+f[2,2]+f[2,3]
```

ConstantArray

WMA link

```
ConstantArray[expr, n]  
    returns a list of  $n$  copies of  $expr$ .
```

```

>> ConstantArray[a, 3]
{a, a, a}

>> ConstantArray[a, {2, 3}]
{{a, a, a}, {a, a, a}}

```

List

WMA link

`List[e1, e2, ..., ei]`
`{e1, e2, ..., ei}`
represents a list containing the elements $e_1 \dots e_i$.

`List` is the head of lists:

```

>> Head[{1, 2, 3}]
List

```

Lists can be nested:

```

>> {{a, b, {c, d}}}
{{a, b, {c, d}}}

```

Normal

WMA link

`Normal[expr_]`
Brings especial expressions to a normal expression from different especial forms.

Permutations

WMA link

`Permutations[list]`
gives all possible orderings of the items in `list`.
`Permutations[list, n]`
gives permutations up to length n .
`Permutations[list, {n}]`
gives permutations of length n .

```

>> Permutations[{y, 1, x}]
{{y, 1, x}, {y, x, 1}, {1, y, x}, {1, x, y}, {x, y, 1}, {x, 1, y}}

```

Elements are differentiated by their position in `list`, not their value.

```

>> Permutations[{a, b, b}]
{{a, b, b}, {a, b, b}, {b, a, b}, {b, b, a}, {b, a, b}, {b, b, a}}

>> Permutations[{1, 2, 3}, 2]
{{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 1}, {2, 3}, {3, 1}, {3, 2}}

```

```
>> Permutations[{1, 2, 3}, {2}]
{{1,2},{1,3},{2,1},{2,3},{3,1},{3,2}}
```

Range

WMA link

```
Range[n]
    returns a list of integers from 1 to n.
Range[a, b]
    returns a list of integers from a to b.
```

```
>> Range[5]
{1,2,3,4,5}

>> Range[-3, 2]
{-3,-2,-1,0,1,2}

>> Range[0, 2, 1/3]
{0, 1/3, 2/3, 1, 4/3, 5/3, 2}
```

Reap

WMA link

```
Reap[expr]
    gives the result of evaluating expr, together with all values sown during this evaluation. Values sown with different tags are given in different lists.
Reap[expr, pattern]
    only yields values sown with a tag matching pattern. Reap[expr] is equivalent to Reap[expr, _].
Reap[expr, {pattern1, pattern2, ...}]
    uses multiple patterns.
Reap[expr, pattern, f]
    applies f on each tag and the corresponding values sown in the form f[tag, {e1, e2, ...}].
```

```
>> Reap[Sow[3]; Sow[1]]
{1,{3,1}{{1,3,1}}}

>> Reap[Sow[2, {x, x, x}]; Sow[3, x]; Sow[4, y]; Sow[4, 1], {_Symbol,
_Integer, x}, f]
{4,{f[x,{2,2,2,3}],f[y,{4}],{f[1,{4}]},{f[x,{2,2,2,3}]}}}
```

Find the unique elements of a list, keeping their order:

```
>> Reap[Sow[Null, {a, a, b, d, c, a}], _, # &][[2]]
{a,b,d,c}
```

Sown values are reaped by the innermost matching Reap:

```

>> Reap[Reap[Sow[a, x]; Sow[b, 1], _Symbol, Print["Inner: ", #1]&], _, ,
f]
Inner: x
{Null, {f [1, {b}]}}

```

When no value is sown, an empty list is returned:

```

>> Reap[x]
{x, {}}

```

RowBox

WMA link

`RowBox[{...}]`

is a box construct that represents a sequence of boxes arranged in a horizontal row.

Sow

WMA link

`Sow[e]`

sends the value e to the innermost `Reap`.

`Sow[e, tag]`

sows e using tag . `Sow[e]` is equivalent to `Sow[e, Null]`.

`Sow[e, {tag1, tag2, ...}]`

uses multiple tags.

Table

WMA link

`Table[expr, n]`

generates a list of n copies of $expr$.

`Table[expr, {i, n}]`

generates a list of the values of $expr$ when i runs from 1 to n .

`Table[expr, {i, start, stop, step}]`

evaluates $expr$ with i ranging from $start$ to $stop$, incrementing by $step$.

`Table[expr, {i, {e1, e2, ..., ei}}]`

evaluates $expr$ with i taking on the values $e1, e2, \dots, ei$.

```

>> Table[x, 3]
{x,x,x}

>> n = 0; Table[n = n + 1, {5}]
{1,2,3,4,5}

>> Table[i, {i, 4}]
{1,2,3,4}

>> Table[i, {i, 2, 5}]
{2,3,4,5}

```

```

>> Table[i, {i, 2, 6, 2}]
{2,4,6}

>> Table[i, {i, Pi, 2 Pi, Pi / 2}]
 $\left\{\pi, \frac{3\pi}{2}, 2\pi\right\}$ 

>> Table[x^2, {x, {a, b, c}}]
 $\left\{a^2, b^2, c^2\right\}$ 

Table supports multi-dimensional tables:
>> Table[{i, j}, {i, {a, b}}, {j, 1, 2}]
{{{{a, 1}, {a, 2}}, {{b, 1}, {b, 2}}}}

```

Tuples

WMA link

```

Tuples[list, n]
    returns a list of all  $n$ -tuples of elements in list.
Tuples[{list1, list2, ...}]
    returns a list of tuples with elements from the given lists.

```

```

>> Tuples[{a, b, c}, 2]
{{{a, a}, {a, b}, {a, c}, {b, a}, {b, b}, {b, c}, {c, a}, {c, b}, {c, c}}}

>> Tuples[{}, 2]
{{}}
```

```

>> Tuples[{a, b, c}, 0]
{{}}
```

The head of *list* need not be *List*:

```

>> Tuples[f[a, b, c], 2]
{f[a, a], f[a, b], f[a, c], f[b, a], f[b, b], f[b, c], f[c, a], f[c, b], f[c, c]}
```

However, when specifying multiple expressions, *List* is always used:

```

>> Tuples[{f[a, b], g[c, d]}]
{{a, c}, {a, d}, {b, c}, {b, d}}}
```

Elements of Lists

Elements of Lists

Functions for accessing elements of lists using either indices, positions, or patterns of criteria.

Append

WMA link

```
Append[expr, elem]
      returns expr with elem appended.
```

```
>> Append[{1, 2, 3}, 4]
{1, 2, 3, 4}
```

Append works on expressions with heads other than List:

```
>> Append[f[a, b], c]
f[a, b, c]
```

Unlike Join, Append does not flatten lists in item:

```
>> Append[{a, b}, {c, d}]
{a, b, {c, d}}
```

AppendTo

WMA link

```
AppendTo[s, elem]
append elem to value of s and sets s to the result.
```

```
>> s = {};
>> AppendTo[s, 1]
{1}
>> s
{1}
```

Append works on expressions with heads other than List:

```
>> y = f[];
>> AppendTo[y, x]
f[x]
>> y
f[x]
```

Cases

WMA link

```
Cases[list, pattern]
      returns the elements of list that match pattern.
Cases[list, pattern, ls]
      returns the elements matching at levelspec ls.
Cases[list, pattern, Heads->bool]
      Match including the head of the expression in the search.
```

```

>> Cases[{a, 1, 2.5, "string"}, _Integer|_Real]
{1,2.5}

>> Cases[_Complex] [{1, 2I, 3, 4-I, 5}]
{2I,4 - I}

```

Find symbols among the elements of an expression:

```

>> Cases[{b, 6, \[Pi]}, _Symbol]
{b,\[Pi]}

```

Also include the head of the expression in the previous search:

```

>> Cases[{b, 6, \[Pi]}, _Symbol, Heads -> True]
{List,b,\[Pi]}

```

Count

WMA link

```

Count [list, pattern]
    returns the number of times pattern appears in list.
Count [list, pattern, ls]
    counts the elements matching at levelspec ls.

```

```

>> Count[{3, 7, 10, 7, 5, 3, 7, 10}, 3]
2

>> Count[{{a, a}, {a, a, a}, a, {2}}]
5

```

Delete

WMA link

```

Delete[expr, i]
    deletes the element at position i in expr. The position is counted from the end if i is negative.
Delete[expr, {m, n, ...}]
    deletes the element at position {m, n, ...}.
Delete[expr, {{m1, n1, ...}, {m2, n2, ...}, ...}]
    deletes the elements at several positions.

```

Delete the element at position 3:

```

>> Delete[{a, b, c, d}, 3]
{a,b,d}

```

Delete at position 2 from the end:

```

>> Delete[{a, b, c, d}, -2]
{a,b,d}

```

Delete at positions 1 and 3:

```

>> Delete[{a, b, c, d}, {{1}, {3}}]
{b,d}

```

Delete in a 2D array:

```
>> Delete[{{a, b}, {c, d}}, {2, 1}]  
{{a, b}, {d}}
```

Deleting the head of a whole expression gives a Sequence object:

```
>> Delete[{a, b, c}, 0]
```

Delete in an expression with any head:

```
>> Delete[f[a, b, c, d], 3]  
f[a, b, d]
```

Delete a head to splice in its arguments:

```
>> Delete[f[a, b, u + v, c], {3, 0}]  
f[a, b, u, v, c]  
  
>> Delete[{a, b, c}, 0]
```

Delete without the position:

```
>> Delete[{a, b, c, d}]  
Delete called with 1 argument; 2 arguments are expected.  
Delete[{a, b, c, d}]
```

Delete with many arguments:

```
>> Delete[{a, b, c, d}, 1, 2]  
Delete called with 3 arguments; 2 arguments are expected.  
Delete[{a, b, c, d}, 1, 2]
```

Delete the element out of range:

```
>> Delete[{a, b, c, d}, 5]  
Part {5} of {a, b, c, d} does not exist.  
Delete[{a, b, c, d}, 5]
```

Delete the position not integer:

```
>> Delete[{a, b, c, d}, {1, n}]  
Position specification n in {a, b, c, d} is not a machine-sized  
integer or a list of machine-sized integers.  
Delete[{a, b, c, d}, {1, n}]
```

DeleteCases

WMA link

```
DeleteCases[list, pattern]  
    returns the elements of list that do not match pattern.  
DeleteCases[list, pattern, levelspec]  
    removes all parts of $list on levels specified by levelspec that match pattern (not fully imple-  
    mented).  
DeleteCases[list, pattern, levelspec, n]  
    removes the first n parts of list that match pattern.
```

```
>> DeleteCases[{a, 1, 2.5, "string"}, _Integer | _Real]  
{a, string}
```

```
>> DeleteCases[{a, b, 1, c, 2, 3}, _Symbol]
{1,2,3}
```

Drop

WMA link

```
Drop[expr, n]
    returns expr with the first n elements removed.
```

```
>> Drop[{a, b, c, d}, 3]
{d}
>> Drop[{a, b, c, d}, -2]
{a, b}
>> Drop[{a, b, c, d, e}, {2, -2}]
{a, e}
```

Drop a submatrix:

```
>> A = Table[i*10 + j, {i, 4}, {j, 4}]
{{11, 12, 13, 14}, {21, 22, 23, 24}, {31, 32, 33, 34}, {41, 42, 43, 44}}
>> Drop[A, {2, 3}, {2, 3}]
{{11, 14}, {41, 44}}
```

Extract

WMA link

```
Extract[expr, list]
    extracts parts of expr specified by list.
Extract[expr, {list1, list2, ...}]
    extracts a list of parts.
```

Extract[expr, i, j, ...] is equivalent to *Part[expr, {i, j, ...}]*.

```
>> Extract[a + b + c, {2}]
b
>> Extract[{{a, b}, {c, d}}, {{1}, {2, 2}}]
{{a, b}, d}
```

First

WMA link

```
First[expr]
    returns the first element in expr.
```

First[expr] is equivalent to *expr[[1]]*.

```

>> First[{a, b, c}]
a

>> First[a + b + c]
a

>> First[x]
Nonatomic expression expected.

First[x]

>> First[{}]
{} has zero length and no first element.

First[{}]

```

FirstCase

WMA link

FirstCase[{e1, e2, ...}, pattern] gives the first <i>ei</i> to match <i>pattern</i> , or \$Missing["NotFound"]\$ if none matching pattern is found.
FirstCase[{e1,e2, ...}, pattern -> rhs] gives the value of <i>rhs</i> corresponding to the first <i>ei</i> to match pattern.
FirstCase[expr, pattern, default] gives <i>default</i> if no element matching <i>pattern</i> is found.
FirstCase[expr, pattern, default, levelspec] finds only objects that appear on levels specified by <i>levelspec</i> .
FirstCase[pattern] represents an operator form of FirstCase that can be applied to an expression.

FirstPosition

WMA link

FirstPosition[expr, pattern] gives the position of the first element in <i>expr</i> that matches <i>pattern</i> , or Missing["NotFound"] if no such element is found.
FirstPosition[expr, pattern, default] gives <i>default</i> if no element matching <i>pattern</i> is found.
FirstPosition[expr, pattern, default, levelspec] finds only objects that appear on levels specified by <i>levelspec</i> .

```

>> FirstPosition[{a, b, a, a, b, c, b}, b]
{2}

>> FirstPosition[{{a, a, b}, {b, a, a}, {a, b, a}}, b]
{1,3}

>> FirstPosition[{x, y, z}, b]
Missing[NotFound]

```

Find the first position at which x^2 to appears:

```
>> FirstPosition[{1 + x^2, 5, x^4, a + (1 + x^2)^2}, x^2]
{1,2}
```

Insert

WMA link

Insert[*list*, *elem*, *n*]

inserts *elem* at position *n* in *list*. When *n* is negative, the position is counted from the end.

```
>> Insert[{a,b,c,d,e}, x, 3]
{a,b,x,c,d,e}
>> Insert[{a,b,c,d,e}, x, -2]
{a,b,c,d,x,e}
```

Last

WMA link

Last[*expr*]

returns the last element in *expr*.

Last[*expr*] is equivalent to *expr*[[-1]].

```
>> Last[{a, b, c}]
c
>> Last[x]
Nonatomic expression expected.
Last[x]
>> Last[{}]
{} has zero length and no last element.
Last[{}]
```

Length

WMA link

Length[*expr*]

returns the number of elements in *expr*.

Length of a list:

```
>> Length[{1, 2, 3}]
3
```

Length operates on the **FullForm** of expressions:

```
>> Length[Exp[x]]
2
>> FullForm[Exp[x]]
Power[E, x]
```

The length of atoms is 0:

```
>> Length[a]  
0
```

Note that rational and complex numbers are atoms, although their `FullForm` might suggest the opposite:

```
>> Length[1/3]  
0  
>> FullForm[1/3]  
Rational[1,3]
```

Most

WMA link

`Most[expr]`

returns *expr* with the last element removed.

`Most[expr]` is equivalent to `expr[[;;-2]]`.

```
>> Most[{a, b, c}]  
{a, b}  
>> Most[a + b + c]  
a + b  
>> Most[x]  
Nonatomic expression expected.  
Most[x]
```

Part

WMA link

`Part[expr, i]`

returns part *i* of *expr*.

Extract an element from a list:

```
>> A = {a, b, c, d};  
  
>> A[[3]]  
c
```

Negative indices count from the end:

```
>> {a, b, c}[[ -2]]  
b
```

`Part` can be applied on any expression, not necessarily lists.

```
>> (a + b + c)[[2]]  
b
```

`expr[[0]]` gives the head of *expr*:

```
>> (a + b + c)[[0]]  
Plus
```

Parts of nested lists:

```
>> M = {{a, b}, {c, d}};  
  
>> M[[1, 2]]  
b
```

You can use Span to specify a range of parts:

```
>> {1, 2, 3, 4}[[2;;4]]  
{2,3,4}  
  
>> {1, 2, 3, 4}[[2;;-1]]  
{2,3,4}
```

A list of parts extracts elements at certain indices:

```
>> {a, b, c, d}[[{1, 3, 3}]]  
{a,c,c}
```

Get a certain column of a matrix:

```
>> B = {{a, b, c}, {d, e, f}, {g, h, i}};  
  
>> B[;, 2]  
{b,e,h}
```

Extract a submatrix of 1st and 3rd row and the two last columns:

```
>> B = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  
  
>> B[{{1, 3}, -2;;-1}]  
{ {2,3}, {8,9}}
```

The 3d column of a matrix:

```
>> {{a, b, c}, {d, e, f}, {g, h, i}}[[All, 3]]  
{c,f,i}
```

Further examples:

```
>> (a+b+c+d)[[-1;;-2]]  
0  
  
>> x[[2]]  
Part specification is longer than depth of object.  
x[[2]]
```

Assignments to parts are possible:

```
>> B[;, 2] = {10, 11, 12}  
{10,11,12}  
  
>> B  
  
>> B[;, 3] = 13  
13  
  
>> B  
  
>> B[[1;;-2]] = t;  
  
>> B
```

```

>> F = Table[i*j*k, {i, 1, 3}, {j, 1, 3}, {k, 1, 3}];
>> F[[;; All, 2 ;; 3, 2]] = t;
>> F
>> F[[;; All, 1 ;; 2, 3 ;; 3]] = k;
>> F

```

Of course, part specifications have precedence over most arithmetic operations:

```

>> A[[1]] + B[[2]] + C[[3]] // Hold // FullForm
Hold[Plus[Part[A, 1], Part[B, 2], Part[C, 3]]]

```

Pick

WMA link

```

Pick[list, sel]
  returns those items in list that are True in sel.
Pick[list, sel, patt]
  returns those items in list that match patt in sel.

```

```

>> Pick[{a, b, c}, {False, True, False}]
{b}
>> Pick[f[g[1, 2], h[3, 4]], {{True, False}, {False, True}}]
f[g[1], h[4]]
>> Pick[{a, b, c, d, e}, {1, 2, 3.5, 4, 5.5}, _Integer]
{a, b, d}

```

Position

WMA link

```

Position[expr, patt]
  returns the list of positions for which expr matches patt.
Position[expr, patt, ls]
  returns the positions on levels specified by levelspec ls.

```

```

>> Position[{1, 2, 2, 1, 2, 3, 2}, 2]
{{2}, {3}, {5}, {7}}

```

Find positions upto 3 levels deep:

```

>> Position[{1 + Sin[x], x, (Tan[x] - y)^2}, x, 3]
{{1, 2, 1}, {2}}

```

Find all powers of x:

```

>> Position[{1 + x^2, x y ^ 2, 4 y, x ^ z}, x^_]
{{1, 2}, {4}}

```

Use Position as an operator:

```
>> Position[_Integer][{1.5, 2, 2.5}]  
{{2}}
```

Prepend

WMA link

```
Prepend[expr, item]  
    returns expr with item prepended to its elements.  
Prepend[expr]  
    Prepend[elem] [expr] is equivalent to Prepend[expr, elem].
```

Prepend is similar to Append, but adds *item* to the beginning of *expr*:

```
>> Prepend[{2, 3, 4}, 1]  
{1,2,3,4}
```

Prepend works on expressions with heads other than List:

```
>> Prepend[f[b, c], a]  
f[a,b,c]
```

Unlike Join, Prepend does not flatten lists in *item*:

```
>> Prepend[{c, d}, {a, b}]  
{\{a,b\},c,d}
```

PrependTo

WMA link

```
PrependTo[s, item]  
    prepends item to value of s and sets s to the result.
```

Assign s to a list

```
>> s = {1, 2, 4, 9}  
{1,2,4,9}
```

Add a new value at the beginning of the list:

```
>> PrependTo[s, 0]  
{0,1,2,4,9}
```

The value assigned to s has changed:

```
>> s  
{0,1,2,4,9}
```

PrependTo works with a head other than List:

```
>> y = f[a, b, c];  
>> PrependTo[y, x]  
f[x,a,b,c]
```

```
>> y  
f[x,a,b,c]
```

ReplacePart

WMA link

```
ReplacePart[expr, i -> new]  
  replaces part i in expr with new.  
ReplacePart[expr, {{i, j} -> e1, {k, l} -> e2}]  
  replaces parts i and j with e1, and parts k and l with e2.
```

```
>> ReplacePart[{a, b, c}, 1 -> t]  
{t,b,c}  
  
>> ReplacePart[{{a, b}, {c, d}}, {2, 1} -> t]  
{ {a,b},{t,d}}  
  
>> ReplacePart[{{a, b}, {c, d}}, {{2, 1} -> t, {1, 1} -> t}]  
{ {t,b},{t,d}}  
  
>> ReplacePart[{a, b, c}, {{1}, {2}} -> t]  
{t,t,c}
```

Delayed rules are evaluated once for each replacement:

```
>> n = 1;  
  
>> ReplacePart[{a, b, c, d}, {{1}, {3}} :> n++]  
{1,b,2,d}
```

Non-existing parts are simply ignored:

```
>> ReplacePart[{a, b, c}, 4 -> t]  
{a,b,c}
```

You can replace heads by replacing part 0:

```
>> ReplacePart[{a, b, c}, 0 -> Times]  
abc
```

(This is equivalent to `Apply`.)

Negative part numbers count from the end:

```
>> ReplacePart[{a, b, c}, -1 -> t]  
{a,b,t}
```

Rest

WMA link

```
Rest[expr]  
  returns expr with the first element removed.
```

`Rest[expr]` is equivalent to `expr[[2;;]]`.

```

>> Rest[{a, b, c}]
{b,c}

>> Rest[a + b + c]
b+c

>> Rest[x]
Nonatomic expression expected.
Rest[x]

>> Rest[{}]
Cannot take Rest of expression {} with length zero.
Rest[{}]

```

RowBox

WMA link

`RowBox[{...}]`
is a box construct that represents a sequence of boxes arranged in a horizontal row.

Select

WMA link

`Select[{e1, e2, ...}, f]`
returns a list of the elements e_i for which $f[e_i]$ returns True.

Find numbers greater than zero:

```

>> Select[{-3, 0, 1, 3, a}, #>0&]
{1,3}

```

Select works on an expression with any head:

```

>> Select[f[a, 2, 3], NumberQ]
f[2,3]

>> Select[a, True]
Nonatomic expression expected.
Select[a,True]

```

Span (; ;)

WMA link

`Span`
is the head of span ranges like `1 ; ; 3`.

```

>> ; ; // FullForm
Span[1,All]

```

```

>> 1;;4;;2 // FullForm
Span[1,4,2]

>> 2;;-2 // FullForm
Span[2, -2]

>> ;;3 // FullForm
Span[1,3]

```

Take

WMA link

Take[*expr*, *n*]
 returns *expr* with all but the first *n* elements removed.

```

>> Take[{a, b, c, d}, 3]
{a,b,c}

>> Take[{a, b, c, d}, -2]
{c,d}

>> Take[{a, b, c, d, e}, {2, -2}]
{b,c,d}

```

Take a submatrix:

```

>> A = {{a, b, c}, {d, e, f}};

>> Take[A, 2, 2]
{{a,b},{d,e}}

```

Take a single column:

```

>> Take[A, All, {2}]
{{b},{e}}

```

UpTo

WMA link

UpTo[*n*]
 is a symbolic specification that represents up to *n* objects or positions. If *n* objects or positions are available, all are used. If fewer are available, only those available are used.

Math & Counting Operations on Lists

Math & Counting Operations on List

TakeLargestBy

WMA link

TakeLargestBy[*list*, *f*, *n*]

returns the a sorted list of the *n* largest items in *list* using *f* to retrieve the items' keys to compare them.

For details on how to use the ExcludedForms option, see TakeLargest[].

```
>> TakeLargestBy[{{1, -1}, {10, 100}, {23, 7, 8}, {5, 1}}, Total, 2]
{{10, 100}, {23, 7, 8}}
>> TakeLargestBy[{"abc", "ab", "x"}, StringLength, 1]
{abc}
```

TakeSmallestBy

WMA link

TakeSmallestBy[*list*, *f*, *n*]

returns the a sorted list of the *n* smallest items in *list* using *f* to retrieve the items' keys to compare them.

For details on how to use the ExcludedForms option, see TakeLargest[].

```
>> TakeSmallestBy[{{1, -1}, {10, 100}, {23, 7, 8}, {5, 1}}, Total, 2]
{{1, -1}, {5, 1}}
>> TakeSmallestBy[{"abc", "ab", "x"}, StringLength, 1]
{x}
```

Predicates on Lists

Predicates on List

ContainsOnly

WMA link

ContainsOnly[*list1*, *list2*]

yields True if *list1* contains only elements that appear in *list2*.

```
>> ContainsOnly[{b, a, a}, {a, b, c}]
True
```

The first list contains elements not present in the second list:

```
>> ContainsOnly[{b, a, d}, {a, b, c}]
False
>> ContainsOnly[{}, {a, b, c}]
True
```

Use Equal as the comparison function to have numerical tolerance:

```
>> ContainsOnly[{a, 1.0}, {1, a, b}, {SameTest -> Equal}]
True
```

Rearranging and Restructuring Lists

Rearranging and Restructuring Lists

These functions reorder and rearrange lists.

Catenate

WMA link

```
Catenate[{l1, l2, ...}]  
concatenates the lists l1, l2, ...
```

```
>> Catenate[{{1, 2, 3}, {4, 5}}]  
{1,2,3,4,5}
```

Complement

WMA link

```
Complement[all, e1, e2, ...]  
returns an expression containing the elements in the set all that are not in any of e1, e2, etc.  
Complement[all, e1, e2, ..., SameTest->test]  
applies test to the elements in all and each of the ei to determine equality.
```

The sets *all*, *e1*, etc can have any head, which must all match.

The returned expression has the same head as the input expressions. The expression will be sorted and each element will only occur once.

```
>> Complement[{a, b, c}, {a, c}]  
{b}  
  
>> Complement[{a, b, c}, {a, c}, {b}]  
{}  
  
>> Complement[f[z, y, x, w], f[x], f[x, z]]  
f[w, y]  
  
>> Complement[{c, b, a}]  
{a, b, c}
```

DeleteDuplicates

WMA link

```
DeleteDuplicates[list]  
deletes duplicates from list.  
DeleteDuplicates[list, test]  
deletes elements from list based on whether the function test yields True on pairs of elements.  
DeleteDuplicates does not change the order of the remaining elements.
```

```
>> DeleteDuplicates[{1, 7, 8, 4, 3, 4, 1, 9, 9, 2, 1}]  
{1, 7, 8, 4, 3, 9, 2}
```

```
>> DeleteDuplicates[{3,2,1,2,3,4}, Less]
{3,2,1}
```

Flatten

WMA link

```
Flatten[expr]
  flattens out nested lists in expr.
Flatten[expr, n]
  stops flattening at level n.
Flatten[expr, n, h]
  flattens expressions with head h instead of List.
```

```
>> Flatten[{{a, b}, {c, {d}, e}, {f, {g, h}}}]
{a,b,c,d,e,f,g,h}

>> Flatten[{{a, b}, {c, {e}, e}, {f, {g, h}}}, 1]
{a,b,c,{e},e,f,{g,h}}

>> Flatten[f[a, f[b, f[c, d]], e], Infinity, f]
f[a,b,c,d,e]

>> Flatten[{{a, b}, {c, d}}, {{2}, {1}}]
{{a,c},{b,d}}

>> Flatten[{{a, b}, {c, d}}, {{1, 2}}]
{a,b,c,d}
```

Flatten also works in irregularly shaped arrays

```
>> Flatten[{{1, 2, 3}, {4}, {6, 7}, {8, 9, 10}}, {{2}, {1}}]
{{1,4,6,8},{2,7,9},{3,10}}
```

Gather

WMA link

```
Gather[list, test]
  gathers elements of list into sub lists of items that are the same according to test.
Gather[list]
  gathers elements of list into sub lists of items that are the same.
```

The order of the items inside the sub lists is the same as in the original list.

```
>> Gather[{1, 7, 3, 7, 2, 3, 9}]
{{1},{7,7},{3,3},{2},{9}}

>> Gather[{1/3, 2/6, 1/9}]
{{1/3,1/3},{1/9}}
```

GatherBy

WMA link

```
GatherBy[list, f]
```

gathers elements of *list* into sub lists of items whose image under *f* identical.

```
GatherBy[list, {f, g, ...}]
```

gathers elements of *list* into sub lists of items whose image under *f* identical. Then, gathers these sub lists again into sub sub lists, that are identical under *g*.

```
>> GatherBy[{{1, 3}, {2, 2}, {1, 1}}, Total]
{{{1,3},{2,2}},{1,1}}
```



```
>> GatherBy[{"xy", "abc", "ab"}, StringLength]
{{xy,ab},{abc}}
```



```
>> GatherBy[{{2, 0}, {1, 5}, {1, 0}}, Last]
{{2,0},{1,0}},{1,5}}
```



```
>> GatherBy[{{1, 2}, {2, 1}, {3, 5}, {5, 1}, {2, 2, 2}}, {Total, Length}]
{{{1,2},{2,1}}},{{{3,5}}},{{{5,1}}},{{{2,2,2}}}}
```

Intersection

WMA link

```
Intersection[a, b, ...]
```

gives the intersection of the sets. The resulting list will be sorted and each element will only occur once.

```
>> Intersection[{1000, 100, 10, 1}, {1, 5, 10, 15}]
{1,10}
```



```
>> Intersection[{{a, b}, {x, y}}, {{x, x}, {x, y}, {x, z}}]
{{x,y}}
```



```
>> Intersection[{c, b, a}]
{a,b,c}
```



```
>> Intersection[{1, 2, 3}, {2, 3, 4}, SameTest->Less]
{3}
```

Join

WMA link

```
Join[l1, l2]
```

concatenates the lists *l1* and *l2*.

Join concatenates lists:

```

>> Join[{a, b}, {c, d, e}]
{a, b, c, d, e}

>> Join[{{a, b}, {c, d}}, {{1, 2}, {3, 4}}]
{{a, b}, {c, d}, {1, 2}, {3, 4}}

```

The concatenated expressions may have any head:

```

>> Join[a + b, c + d, e + f]
a + b + c + d + e + f

```

However, it must be the same for all expressions:

```

>> Join[a + b, c * d]
Heads Plus and Times are expected to be the same.

Join[a + b, cd]

```

PadLeft

WMA link

PadLeft[list, n] pads <i>list</i> to length <i>n</i> by adding 0 on the left. PadLeft[list, n, x] pads <i>list</i> to length <i>n</i> by adding <i>x</i> on the left. PadLeft[list, {n1, \$n2, ...}, x] pads <i>list</i> to lengths <i>n1, n2</i> at levels 1, 2, ... respectively by adding <i>x</i> on the left. PadLeft[list, n, x, m] pads <i>list</i> to length <i>n</i> by adding <i>x</i> on the left and adding a margin of <i>m</i> on the right. PadLeft[list, n, x, {m1, m2, ...}] pads <i>list</i> to length <i>n</i> by adding <i>x</i> on the left and adding margins of <i>m1, m2, ...</i> on levels 1, 2, ... on the right. PadLeft[list] turns the ragged list <i>list</i> into a regular list by adding 0 on the left.

```

>> PadLeft[{1, 2, 3}, 5]
{0, 0, 1, 2, 3}

>> PadLeft[x[a, b, c], 5]
x[0, 0, a, b, c]

>> PadLeft[{1, 2, 3}, 2]
{2, 3}

>> PadLeft[{{}, {1, 2}, {1, 2, 3}}]
{{0, 0, 0}, {0, 1, 2}, {1, 2, 3} }

>> PadLeft[{1, 2, 3}, 10, {a, b, c}, 2]
{b, c, a, b, c, 1, 2, 3, a, b}

>> PadLeft[{{1, 2, 3}}, {5, 2}, x, 1]
{{x, x}, {x, x}, {x, x}, {3, x}, {x, x}}

```

PadRight

WMA link

```
PadRight[list, n]
  pads list to length n by adding 0 on the right.
PadRight[list, n, x]
  pads list to length n by adding x on the right.
PadRight[list, {n1, $n2, ...}, x]
  pads list to lengths n1, n2 at levels 1, 2, ... respectively by adding x on the right.
PadRight[list, n, x, m]
  pads list to length n by adding x on the left and adding a margin of m on the left.
PadRight[list, n, x, {m1, m2, ...}]
  pads list to length n by adding x on the right and adding margins of m1, m2, ... on levels 1, 2,
  ... on the left.
PadRight[list]
  turns the ragged list list into a regular list by adding 0 on the right.
```

```
>> PadRight[{1, 2, 3}, 5]
{1,2,3,0,0}

>> PadRight[x[a, b, c], 5]
x[a,b,c,0,0]

>> PadRight[{1, 2, 3}, 2]
{1,2}

>> PadRight[{{}, {1, 2}, {1, 2, 3}}]
{{0,0,0},{1,2,0},{1,2,3}>

>> PadRight[{1, 2, 3}, 10, {a, b, c}, 2]
{b,c,1,2,3,a,b,c,a,b}

>> PadRight[{{1, 2, 3}}, {5, 2}, x, 1]
{{x,x},{x,1},{x,x},{x,x},{x,x}}
```

Partition

WMA link

```
Partition[list, n]
  partitions list into sublists of length n.
Partition[list, n, d]
  partitions list into sublists of length n which overlap d indices.
```

```
>> Partition[{a, b, c, d, e, f}, 2]
{{a,b},{c,d},{e,f}>

>> Partition[{a, b, c, d, e, f}, 3, 1]
{{a,b,c},{b,c,d},{c,d,e},{d,e,f}}
```

Reverse

WMA link

```
Reverse[expr]
  reverses the order of expr's items (on the top level)
Reverse[expr, n]
  reverses the order of items in expr on level n
Reverse[expr, {n1, n2, ...}]
  reverses the order of items in expr on levels n1, n2, ...
```

```
>> Reverse[{1, 2, 3}]
{3,2,1}

>> Reverse[x[a, b, c]]
x[c,b,a]

>> Reverse[{{1, 2}, {3, 4}}, 1]
{{3,4},{1,2}]

>> Reverse[{{1, 2}, {3, 4}}, 2]
{{2,1},{4,3}]

>> Reverse[{{1, 2}, {3, 4}}, {1, 2}]
{{4,3},{2,1}}
```

Riffle

WMA link

```
Riffle[list, x]
  inserts a copy of x between each element of list.
Riffle[{a1, a2, ...}, {b1, b2, ...}]
  interelements the elements of both lists, returning {a1, b1, a2, b2, ...}.
```

```
>> Riffle[{a, b, c}, x]
{a,x,b,x,c}

>> Riffle[{a, b, c}, {x, y, z}]
{a,x,b,y,c,z}

>> Riffle[{a, b, c, d, e, f}, {x, y, z}]
{a,x,b,y,c,z,d,x,e,y,f}
```

RotateLeft

WMA link

RotateLeft [*expr*]
 rotates the items of *expr* by one item to the left.
RotateLeft [*expr*, *n*]
 rotates the items of *expr* by *n* items to the left.
RotateLeft [*expr*, {*n*₁, *n*₂, ...}]
 rotates the items of *expr* by *n*₁ items to the left at the first level, by *n*₂ items to the left at the second level, and so on.

```
>> RotateLeft[{1, 2, 3}]
{2,3,1}

>> RotateLeft[Range[10], 3]
{4,5,6,7,8,9,10,1,2,3}

>> RotateLeft[x[a, b, c], 2]
x[c,a,b]

>> RotateLeft[{{a, b, c}, {d, e, f}, {g, h, i}}, {1, 2}]
{{f,d,e},{i,g,h},{c,a,b}}
```

RotateRight

WMA link

RotateRight [*expr*]
 rotates the items of *expr* by one item to the right.
RotateRight [*expr*, *n*]
 rotates the items of *expr* by *n* items to the right.
RotateRight [*expr*, {*n*₁, *n*₂, ...}]
 rotates the items of *expr* by *n*₁ items to the right at the first level, by *n*₂ items to the right at the second level, and so on.

```
>> RotateRight[{1, 2, 3}]
{3,1,2}

>> RotateRight[Range[10], 3]
{8,9,10,1,2,3,4,5,6,7}

>> RotateRight[x[a, b, c], 2]
x[b,c,a]

>> RotateRight[{{a, b, c}, {d, e, f}, {g, h, i}}, {1, 2}]
{{h,i,g},{b,c,a},{e,f,d}}
```

Split

WMA link

Split [*list*]
 splits *list* into collections of consecutive identical elements.
Split [*list*, *test*]
 splits *list* based on whether the function *test* yields True on consecutive elements.

```

>> Split[{x, x, x, y, x, y, y, z}]
{{x, x, x}, {y}, {x}, {y, y}, {z} }

Split into increasing or decreasing runs of elements
>> Split[{1, 5, 6, 3, 6, 1, 6, 3, 4, 5, 4}, Less]
{{1, 5, 6}, {3, 6}, {1, 6}, {3, 4, 5}, {4} }

>> Split[{1, 5, 6, 3, 6, 1, 6, 3, 4, 5, 4}, Greater]
{{1}, {5}, {6, 3}, {6, 1}, {6, 3}, {4}, {5, 4} }

Split based on first element
>> Split[{x -> a, x -> y, 2 -> a, z -> c, z -> a}, First[#1] === First
[#2] &]
{{x -> a, x -> y}, {2 -> a}, {z -> c, z -> a}}

```

SplitBy

WMA link

```

SplitBy[list, f]
splits list into collections of consecutive elements that give the same result when f is applied.

```

```

>> SplitBy[Range[1, 3, 1/3], Round]
{{1, 4/3}, {5/3, 2, 7/3}, {8/3, 3}}
>> SplitBy[{1, 2, 1, 1.2}, {Round, Identity}]
{{{1}}, {{2}}, {{1}, {1.2}}}

```

Tally

WMA link

```

Tally[list]
counts and returns the number of occurrences of objects and returns the result as a list of pairs
{object, count}.
Tally[list, test]
counts the number of occurrences of objects and uses $test to determine if two objects should
be counted in the same bin.

```

```

>> Tally[{a, b, c, b, a}]
{{a, 2}, {b, 2}, {c, 1}}

```

Tally always returns items in the order as they first appear in *list*:

```

>> Tally[{b, b, a, a, a, d, d, d, d, c}]
{{b, 2}, {a, 3}, {d, 4}, {c, 1}}

```

Union

WMA link

Union[a, b, ...]

gives the union of the given set or sets. The resulting list will be sorted and each element will only occur once.

```
>> Union[{5, 1, 3, 7, 1, 8, 3}]
{1,3,5,7,8}

>> Union[{a, b, c}, {c, d, e}]
{a,b,c,d,e}

>> Union[{c, b, a}]
{a,b,c}

>> Union[{{a, 1}, {b, 2}}, {{c, 1}, {d, 3}}, SameTest->(SameQ[Last[#1],
Last[#2]]&)]
{{b,2},{c,1},{d,3}]

>> Union[{1, 2, 3}, {2, 3, 4}, SameTest->Less]
{1,2,2,3,4}
```

34. Low level Format definitions

Contents

\$BoxForms	458	MakeBoxes	458	ToBoxes	459
----------------------	-----	---------------------	-----	-------------------	-----

\$BoxForms

WMA link

\$BoxForms is the list of box formats.

```
>> $BoxForms
{StandardForm, TraditionalForm}
```

MakeBoxes

WMA link

MakeBoxes [*expr*]
is a low-level formatting primitive that converts *expr* to box form, without evaluating it.
\(... \)
directly inputs box objects.

String representation of boxes

```
>> \(\(x \^ 2\)\)
SuperscriptBox[x, 2]

>> \(\(x \_ 2\)\)
SubscriptBox[x, 2]

>> \(\( a \+ b \% c\)\)
UnderoverscriptBox[a, b, c]

>> \(\( a \& b \% c\)\)
UnderoverscriptBox[a, c, b]

>> \(\(x \& y \)\)
OverscriptBox[x, y]

>> \(\(x \+ y \)\)
UnderscriptBox[x, y]
```

ToBoxes

WMA link

ToBoxes [*expr*]
evaluates *expr* and converts the result to box form.

Unlike **MakeBoxes**, **ToBoxes** evaluates its argument:

```
>> ToBoxes[a + a]
RowBox[{2, ,a}]

>> ToBoxes[a + b]
RowBox[{a, +, b}]

>> ToBoxes[a ^ b] // FullForm
SuperscriptBox["a", "b"]
```

35. Mathematical Functions

Basic arithmetic functions, including complex number arithmetic.

Contents

\$Assumptions	460	Conjugate	463	Rational	467
Abs	461	DirectedInfinity	464	Re	467
Arg	461	I	464	Real	468
Assuming	462	Im	465	RealNumberQ	468
Boole	462	Integer	465	Sign	468
Complex	462	Piecewise	466	Sum	470
ConditionalExpression	463	Product	467		

\$Assumptions

WMA link

\$Assumptions

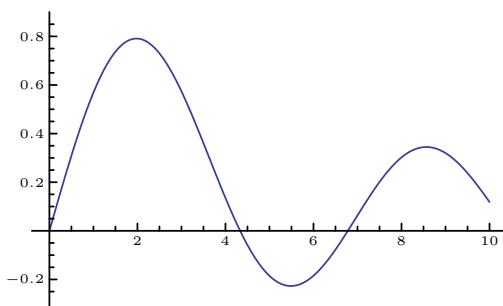
is the default setting for the Assumptions option used in such functions as Simplify, Refine, and Integrate.

Abs

Absolute value (SymPy, WMA)

Abs [*x*]
returns the absolute value of *x*.

```
>> Abs [-3]
3
>> Plot [Abs [x], {x, -4, 4}]
```



Abs returns the magnitude of complex numbers:

```
>> Abs[3 + I]
```

$$\sqrt{10}$$

```
>> Abs[3.0 + I]
```

$$3.16228$$

All of the below evaluate to Infinity:

```
>> Abs[Infinity] == Abs[I Infinity] == Abs[ComplexInfinity]
```

True

Arg

Argument (complex analysis) (WMA link)

Arg[*z, method_option*]

returns the argument of a complex value *z*.

- $\text{Arg}[z]$ is left unevaluated if *z* is not a numeric quantity.
- $\text{Arg}[z]$ gives the phase angle of *z* in radians.
- The result from $\text{Arg}[z]$ is always between $-\pi$ and π .
- $\text{Arg}[z]$ has a branch cut discontinuity in the complex *z* plane running from $-\infty$ to 0.
- $\text{Arg}[0]$ is 0.

```
>> Arg[-3]
```

$$\pi$$

Same as above using sympy's method:

```
>> Arg[-3, Method->"sympy"]
```

$$\pi$$

```
>> Arg[1-I]
```

$$-\frac{\pi}{4}$$

Arg evaluate the direction of DirectedInfinity quantities by the Arg of their arguments:

```
>> Arg[DirectedInfinity[1+I]]
```

$$\frac{\pi}{4}$$

```
>> Arg[DirectedInfinity[]]
```

$$1$$

Arg for 0 is assumed to be 0:

```
>> Arg[0]
```

$$0$$

Assuming

WMA link

Assuming[*cond, expr*]

Evaluates *expr* assuming the conditions *cond*.

```

>> $Assumptions = { x > 0 }
{x > 0}

>> Assuming[y>0, ConditionalExpression[y x^2, y>0]//Simplify]
x2y

>> Assuming[Not[y>0], ConditionalExpression[y x^2, y>0]//Simplify]
Undefined

>> ConditionalExpression[y x ^ 2, y > 0]//Simplify
ConditionalExpression[x2y, y > 0]

```

Boole

WMA link

Boole[expr]
returns 1 if expr is True and 0 if expr is False.

```

>> Boole[2 == 2]
1

>> Boole[7 < 5]
0

>> Boole[a == 7]
Boole[a==7]

```

Complex

WMA link

Complex
is the head of complex numbers.
Complex[a, b]
constructs the complex number $a + I b$.

```

>> Head[2 + 3*I]
Complex

>> Complex[1, 2/3]
1 +  $\frac{2I}{3}$ 

>> Abs[Complex[3, 4]]
5

```

ConditionalExpression

WMA link

ConditionalExpression[*expr*, *cond*]
 returns *expr* if *cond* evaluates to *True*, *Undefined* if *cond* evaluates to *False*.

```
>> ConditionalExpression[x^2, True]
x2
>> ConditionalExpression[x^2, False]
Undefined
>> f = ConditionalExpression[x^2, x>0]
ConditionalExpression [x2, x > 0]
>> f /. x -> 2
4
>> f /. x -> -2
Undefined
```

ConditionalExpression uses assumptions to evaluate the condition:

```
>> $Assumptions = x > 0;

>> ConditionalExpression[x ^ 2, x>0]//Simplify
x2
>> $Assumptions = True;

# » ConditionalExpression[ConditionalExpression[s,x>a], x<b] # = ConditionalExpression[s, And[x>a, x<b]]
```

Conjugate

Complex Conjugate (WMA)

Conjugate[*z*]
 returns the complex conjugate of the complex number *z*.

```
>> Conjugate[3 + 4 I]
3 - 4I
>> Conjugate[3]
3
>> Conjugate[a + b * I]
Conjugate[a] - IConjugate[b]
>> Conjugate[{{1, 2 + I 4, a + I b}, {I}}]
{{1, 2 - 4I, Conjugate[a] - IConjugate[b]}, {-I}}
>> Conjugate[1.5 + 2.5 I]
1.5 - 2.5I
```

DirectedInfinity

WMA link

DirectedInfinity[z]
represents an infinite multiple of the complex number z .
DirectedInfinity[]
is the same as **ComplexInfinity**.

```
>> DirectedInfinity[1]
∞

>> DirectedInfinity[]
ComplexInfinity

>> DirectedInfinity[1 + I]
 $\left(\frac{1}{2} + \frac{I}{2}\right)\sqrt{2}\infty$ 

>> 1 / DirectedInfinity[1 + I]
0

>> DirectedInfinity[1] + DirectedInfinity[-1]
Indeterminate expression -Infinity + Infinity encountered.

>> DirectedInfinity[0]
Indeterminate expression 0 Infinity encountered.

>> DirectedInfinity[0]
Indeterminate
```

I

Imaginary unit (WMA)

I
represents the imaginary number $\text{Sqrt}[-1]$.

```
>> I^2
-1

>> (3+I)*(3-I)
10
```

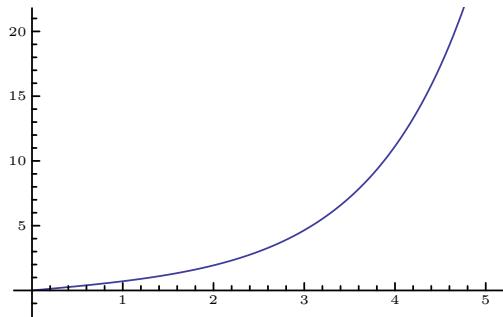
Im

WMA link

Im[z]
returns the imaginary component of the complex number z .

```
>> Im[3+4I]
4
```

```
>> Plot[{Sin[a], Im[E^(I a)]}, {a, 0, 2 Pi}]
```



Integer

WMA link

Integer

is the head of integers.

```
>> Head[5]
Integer
```

Piecewise

WMA link

Piecewise[{{expr1, cond1}, ...}]

represents a piecewise function.

Piecewise[{{expr1, cond1}, ...}, expr]

represents a piecewise function with default expr.

Heaviside function

```
>> Piecewise[{{0, x <= 0}}, 1]
Piecewise[{{0, x<=0}}, 1]

>> Integrate[Piecewise[{{1, x <= 0}, {-1, x > 0}}], x]
Piecewise[{{x, x<=0}}, -x]

>> Integrate[Piecewise[{{1, x <= 0}, {-1, x > 0}}], {x, -1, 2}]
-1
```

Piecewise defaults to 0 if no other case is matching.

```
>> Piecewise[{{1, False}}]
0
```

```

>> Plot[Piecewise[{{Log[x]}, x > 0}, {x*-0.5, x < 0}}], {x, -1, 1}]

```

```

>> Piecewise[{{0 ^ 0, False}}, -1]
-1

```

Product

WMA link

<pre>Product[expr, {i, imin, imax}] evaluates the discrete product of <i>expr</i> with <i>i</i> ranging from <i>imin</i> to <i>imax</i>.</pre> <pre>Product[expr, {i, imax}] same as Product[expr, {i, 1, imax}].</pre> <pre>Product[expr, {i, imin, imax, di}] <i>i</i> ranges from <i>imin</i> to <i>imax</i> in steps of <i>di</i>.</pre> <pre>Product[expr, {i, imin, imax}, {j, jmin, jmax}, ...] evaluates <i>expr</i> as a multiple product, with {<i>i</i>, ...}, {<i>j</i>, ...}, ... being in outermost-to-innermost order.</pre>

```

>> Product[k, {k, 1, 10}]
3628800

>> 10!
3628800

>> Product[x^k, {k, 2, 20, 2}]
x^10

>> Product[2 ^ i, {i, 1, n}]
2^(n+1)/2

>> Product[f[i], {i, 1, 7}]
f[1] f[2] f[3] f[4] f[5] f[6] f[7]

```

Symbolic products involving the factorial are evaluated:

```

>> Product[k, {k, 3, n}]
n!
--2

```

Evaluate the *n*th primorial:

```

>> primorial[0] = 1;

>> primorial[n_Integer] := Product[Prime[k], {k, 1, n}];

```

```
>> primorial[12]
7420738134810
```

Rational

WMA link

Rational
is the head of rational numbers.
Rational[*a*, *b*]
constructs the rational number *a* / *b*.

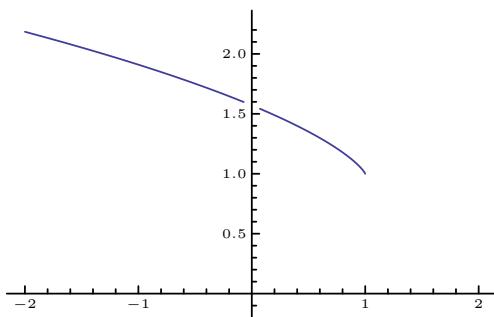
```
>> Head[1/2]
Rational
>> Rational[1, 2]
1
2
```

Re

WMA link

Re[*z*]
returns the real component of the complex number *z*.

```
>> Re[3+4I]
3
>> Plot[{Cos[a], Re[E^(I a)]}, {a, 0, 2 Pi}]
```



Real

WMA link

Real
is the head of real (inexact) numbers.

```
>> x = 3. ^ -20;
```

```

>> InputForm[x]
2.8679719907924413^ - 10
>> Head[x]
Real

```

RealNumberQ

RealNumberQ [*expr*]
 returns True if *expr* is an explicit number with no imaginary component.

```

>> RealNumberQ[10]
True
>> RealNumberQ[4.0]
True
>> RealNumberQ[1+I]
False
>> RealNumberQ[0 * I]
True
>> RealNumberQ[0.0 * I]
True

```

Sign

WMA link

Sign [*x*]
 return -1, 0, or 1 depending on whether *x* is negative, zero, or positive.

```

>> Sign[19]
1
>> Sign[-6]
-1
>> Sign[0]
0
>> Sign[{-5, -10, 15, 20, 0}]
{-1, -1, 1, 1, 0}
>> Sign[3 - 4*I]
 $\frac{3}{5} - \frac{4I}{5}$ 

```

Sum

WMA link

```
Sum[expr, {i, imin, imax}]
    evaluates the discrete sum of expr with i ranging from imin to imax.
Sum[expr, {i, imax}]
    same as Sum[expr, {i, 1, imax}].
Sum[expr, {i, imin, imax, di}]
    i ranges from imin to imax in steps of di.
Sum[expr, {i, imin, imax}, {j, jmin, jmax}, ...]
    evaluates expr as a multiple sum, with {i, ...}, {j, ...}, ... being in outermost-to-innermost order.
```

A sum that Gauss in elementary school was asked to do to kill time:

```
>> Sum[k, {k, 1, 10}]
      55
```

The symbolic form he used:

```
>> Sum[k, {k, 1, n}]
      
$$\frac{n(1+n)}{2}$$

```

A Geometric series with a finite limit:

```
>> Sum[1 / 2 ^ i, {i, 1, k}]
      
$$\frac{1}{1 - 2^{-k}}$$

```

A Geometric series using Infinity:

```
>> Sum[1 / 2 ^ i, {i, 1, Infinity}]
      1
```

Leibniz formula used in computing Pi:

```
>> Sum[1 / ((-1)^k (2k + 1)), {k, 0, Infinity}]
      
$$\frac{\pi}{4}$$

```

A table of double sums to compute squares:

```
>> Table[Sum[i * j, {i, 0, n}, {j, 0, n}], {n, 0, 4}]
      {0, 1, 9, 36, 100}
```

Computing Harmonic using a sum

```
>> Sum[1 / k ^ 2, {k, 1, n}]
      HarmonicNumber[n, 2]
```

Other symbolic sums:

```
>> Sum[k, {k, n, 2 n}]
      
$$\frac{3n(1+n)}{2}$$

```

A sum with Complex-number iteration values

```
>> Sum[k, {k, I, I + 1}]
      1 + 2I

>> Sum[k, {k, Range[5]}]
      15
```

```
>> Sum[f[i], {i, 1, 7}]
f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7]
Verify algebraic identities:
>> Sum[x^2, {x, 1, y}] - y * (y + 1)*(2 * y + 1)/ 6
0
```

36. Mathematical Optimization

Mathematical optimization is the selection of a best element, with regard to some criterion, from some set of available alternatives.

Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries.

We intend to provide local and global optimization techniques, both numeric and symbolic.

Contents

Maximize	471
Minimize	471

Maximize

WMA link

```
Maximize[f, x]
compute the maximum off respect x that change between a and b.
```

```
>> Maximize[-2 x^2 - 3 x + 5, x]
{ { 49/8, {x -> -3/4} } }

#>> Maximize[1 - (x y - 3)^2, {x, y}] = {{1, {x -> 3, y -> 1}}}
#>> Maximize[{x - 2 y, x^2 + y^2 <= 1}, {x, y}] = {{Sqrt[5], {x -> Sqrt[5] / 5, y -> -2 Sqrt[5] / 5}}}
```

Minimize

WMA link

```
Minimize[f, x]
compute the minimum of f respect x that change between a and b.
```

```
>> Minimize[2 x^2 - 3 x + 5, x]
{ { 31/8, {x -> 3/4} } }

#>> Minimize[(x y - 3)^2 + 1, {x, y}] = {{1, {x -> 3, y -> 1}}}
#>> Minimize[{x - 2 y, x^2 + y^2 <= 1}, {x, y}] = {{-Sqrt[5], {x -> -Sqrt[5] / 5, y -> 2 Sqrt[5] / 5}}}
```

37. Matrices and Linear Algebra

Construction and manipulation of Matrices.

Contents

Constructing Matrices	472	DiamondMatrix	473	Parts of Matrices	473
BoxMatrix	472	DiskMatrix	473	Diagonal	474
DiagonalMatrix	472	IdentityMatrix	473		

Constructing Matrices

Constructing Matrices

Methods for constructing Matrices.

BoxMatrix

WMA link

`BoxMatrix[$s]`

Gives a box shaped kernel of size $2s + 1$.

`>> BoxMatrix[3]`

$\{ \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\} \}$

DiagonalMatrix

WMA link

`DiagonalMatrix[list]`

gives a matrix with the values in *list* on its diagonal and zeroes elsewhere.

`>> DiagonalMatrix[{1, 2, 3}]`

$\{ \{1, 0, 0\}, \{0, 2, 0\}, \{0, 0, 3\} \}$

`>> MatrixForm[%]`

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

DiamondMatrix

WMA link

DiamondMatrix[\$s]

Gives a diamond shaped kernel of size $2s + 1$.

>> **DiamondMatrix[3]**

$\{\{0, 0, 0, 1, 0, 0, 0\}, \{0, 0, 1, 1, 1, 0, 0\}, \{0, 1, 1, 1, 1, 1, 0\}, \{1, 1, 1, 1, 1, 1, 1\}, \{0, 1, 1, 1, 1, 1, 0\}, \{0, 0, 1, 1, 1, 0, 0\}, \{0, 0, 0, 1, 1, 0, 0\}\}$

DiskMatrix

WMA link

DiskMatrix[\$s]

Gives a disk shaped kernel of size $2s + 1$.

>> **DiskMatrix[3]**

$\{\{0, 0, 1, 1, 1, 0, 0\}, \{0, 1, 1, 1, 1, 1, 0\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1\}, \{0, 1, 1, 1, 1, 1, 0\}, \{0, 0, 1, 1, 1, 0, 0\}\}$

IdentityMatrix

WMA link

IdentityMatrix[n]

gives the identity matrix with n rows and columns.

>> **IdentityMatrix[3]**

$\{\{1, 0, 0\}, \{0, 1, 0\}, \{0, 0, 1\}\}$

Parts of Matrices

Parts of Matrices

Methods for manipulating Matrices.

Diagonal

WMA link

Diagonal[m]

gives a list with the values in the diagonal of the matrix m .

Diagonal[m, k]

gives a list with the values in the k diagonal of the matrix m .

>> **Diagonal[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]**

$\{1, 5, 9\}$

Get the superdiagonal:

>> **Diagonal[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, 1]**

$\{2, 6\}$

Get the subdiagonal:

>> **Diagonal[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, -1]**

$\{4, 8\}$

Get the diagonal of a nonsquare matrix:

```
>> Diagonal[{{1, 2, 3}, {4, 5, 6}}]  
{1,5}
```

38. Message-related functions.

Contents

\$Aborted	475	General	476	On	477
\$Failed	475	Message	476	Quiet	478
Check	476	MessageName (:) . .	477	Syntax	478
Failure	476	Off	477		

\$Aborted

WMA link

\$Aborted

is returned by a calculation that has been aborted.

\$Failed

WMA link

\$Failed

is returned by some functions in the event of an error.

Check

WMA link

Check [*expr*, *failexpr*]

evaluates *expr*, and returns the result, unless messages were generated, in which case it evaluates and *failexpr* will be returned.

Check [*expr*, *failexpr*, {*s1*::*t1*, *s2*::*t2*, ...}]

checks only for the specified messages.

Return err when a message is generated:

```
>> Check[1/0, err]
    Infinite expression 1 / 0 encountered.
err
```

Check only for specific messages:

```
>> Check[Sin[0^0], err, Sin::argx]
    Indeterminate expression 0 ^ 0 encountered.
Indeterminate
```

```
>> Check[1/0, err, Power::infy]
Infinite expression 1 / 0 encountered.
err
```

Failure

WMA link

Failure[*tag, assoc*]

represents a failure of a type indicated by *tag*, with details given by the association *assoc*.

General

WMA link

General

is a symbol to which all general-purpose messages are assigned.

```
>> General::argr
'1' called with 1 argument; '2' arguments are expected.

>> Message[Rule::argr, Rule, 2]
Rule called with 1 argument; 2 arguments are expected.
```

Message

WMA link

Message[*symbol::msg, expr1, expr2, ...*]

displays the specified message, replacing placeholders in the message text with the corresponding expressions.

```
>> a::b = "Hello world!"
Hello world!

>> Message[a::b]
Hello world!

>> a::c := "Hello '1', Mr 00'2'!"
Hello '1', Mr 00'2'!

>> Message[a::c, "you", 3 + 4]
Hello you, Mr 007!
```

MessageName (: :)

WMA link

MessageName[*symbol, tag*]

symbol::tag

identifies a message.

`MessageName` is the head of message IDs of the form `symbol::tag`.

```
>> FullForm[a::b]  
MessageName[a, "b"]
```

The second parameter tag is interpreted as a string.

```
>> FullForm[a::"b"]  
MessageName[a, "b"]
```

Off

WMA link

```
Off[symbol::tag]  
turns a message off so it is no longer printed.
```

```
>> Off[Power::infy]  
  
>> 1 / 0  
ComplexInfinity  
  
>> Off[Power::indet, Syntax::com]  
  
>> {0 ^ 0,}  
{Indeterminate, Null}
```

On

WMA link

```
On[symbol::tag]  
turns a message on for printing.
```

```
>> Off[Power::infy]  
  
>> 1 / 0  
ComplexInfinity  
  
>> On[Power::infy]  
  
>> 1 / 0  
Infinite expression 1 / 0 encountered.  
ComplexInfinity
```

Quiet

WMA link

```

Quiet[expr, {s1::t1, ...}]
  evaluates expr, without messages {s1::t1, ...} being displayed.
Quiet[expr, All]
  evaluates expr, without any messages being displayed.
Quiet[expr, None]
  evaluates expr, without all messages being displayed.
Quiet[expr, off, on]
  evaluates expr, with messages off being suppressed, but messages on being displayed.

```

Evaluate without generating messages:

```

>> Quiet[1/0]
ComplexInfinity

```

Same as above:

```

>> Quiet[1/0, All]
ComplexInfinity

>> a::b = "Hello";
>> Quiet[x+x, {a::b}]
2x

>> Quiet[Message[a::b]; x+x, {a::b}]
2x

>> Message[a::b]; y=Quiet[Message[a::b]; x+x, {a::b}]; Message[a::b]; y
Hello
Hello
2x

>> Quiet[x + x, {a::b}, {a::b}]
In Quiet[x + x, {a::b}, {a::b}] the message name(s) {a::b} appear in
both the list of messages to switch off and the list of messages to
switch on.
Quiet [x + x, {a::b}, {a::b}]

```

Syntax

WMA link

```

Syntax
  is a symbol to which all syntax messages are assigned.

```

```

>> 1 +
>> Sin[1]
>> ^ 2
>> 1.5``

```

39. Numerical Functions

Support for approximate real numbers and exact real numbers represented in algebraic or symbolic form.

Contents

Chop	479	N	481	Round	483
		Rationalize	482		

Chop

WMA link

Chop [*expr*]
replaces floating point numbers close to 0 by 0.
Chop [*expr*, *delta*]
uses a tolerance of *delta*. The default tolerance is 10^{-10} .

```
>> Chop[10.0 ^ -16]
0

>> Chop[10.0 ^ -9]
1.*^ - 9

>> Chop[10 ^ -11 I]
I
100000000000

>> Chop[0. + 10 ^ -11 I]
0
```

N

WMA link

N [*expr*, *prec*]
evaluates *expr* numerically with a precision of *prec* digits.

```
>> N[Pi, 50]
3.1415926535897932384626433832795028841971693993751

>> N[1/7]
0.142857

>> N[1/7, 5]
0.14286
```

You can manually assign numerical values to symbols. When you do not specify a precision, `MachinePrecision` is taken.

```
>> N[a] = 10.9
10.9
>> a
a
```

`N` automatically threads over expressions, except when a symbol has attributes `NHoldAll`, `NHoldFirst`, or `NHoldRest`.

```
>> N[a + b]
10.9 + b
>> N[a, 20]
a
>> N[a, 20] = 11;
>> N[a + b, 20]
11.0000000000000000000000 + b
>> N[f[a, b]]
f[10.9, b]
>> SetAttributes[f, NHoldAll]
>> N[f[a, b]]
f[a, b]
```

The precision can be a pattern:

```
>> N[c, p_?(#>10&)] := p
>> N[c, 3]
c
>> N[c, 11]
11.000000000
```

You can also use `UpSet` or `TagSet` to specify values for `N`:

```
>> N[d] ^= 5;
```

However, the value will not be stored in `UpValues`, but in `NValues` (as for `Set`):

```
>> UpValues[d]
{}
>> NValues[d]
{HoldPattern[N[d, MachinePrecision]]:>5}
>> e /: N[e] = 6;
>> N[e]
6.
```

Values for `N[expr]` must be associated with the head of `expr`:

```
>> f /: N[e[f]] = 7;
Tag f not found or too deep for an assigned rule.
```

You can use Condition:

```

>> N[g[x_, y_], p_] := x + y * Pi /; x + y > 3
>> SetAttributes[g, NHoldRest]
>> N[g[1, 1]]
g[1., 1]
>> N[g[2, 2]] // InputForm
8.283185307179586

```

The precision of the result is no higher than the precision of the input

```
>> N[Exp[0.1], 100]
1.10517

>> % // Precision
MachinePrecision

>> N[Exp[1/10], 100]
1.105170918075647624811707826490246668224547194737518718792863289440967966747654302989143318970748654

>> % // Precision
100.

>> N[Exp[1.0^20], 100]
2.7182818284590452354

>> % // Precision
20.
```

N can also accept an option "Method". This establishes what is the preferred underlying method to compute numerical values:

```
>> N[F[Pi], 30, Method->"numpy"]
F[3.1415926535897930000000000000000]

>> N[F[Pi], 30, Method->"sympy"]
F[3.14159265358979323846264338328]
```

Rationalize

WMA link

Rationalize[x]
converts a real number x to a nearby rational number with small denominator.
Rationalize[x , dx]
finds the rational number lies within dx of x .

```
>> Rationalize[2.2]
```

11
5

For negative x , $\text{-Rationalize}[-x] == \text{Rationalize}[x]$ which gives symmetric results:

```
>> Rationalize[-11.5, 1]
-11
```

Not all numbers can be well approximated.

```
>> Rationalize[N[Pi]]  
3.14159
```

Find the exact rational representation of N[Pi]

```
>> Rationalize[N[Pi], 0]  
245850922  
78256779
```

Round

WMA link

```
Round[expr]  
rounds expr to the nearest integer.  
Round[expr, k]  
rounds expr to the closest multiple of k.
```

```
>> Round[10.6]  
11  
>> Round[0.06, 0.1]  
0.1  
>> Round[0.04, 0.1]  
0.
```

Constants can be rounded too

```
>> Round[Pi, .5]  
3.  
>> Round[Pi^2]  
10
```

Round to exact value

```
>> Round[2.6, 1/3]  
8  
3  
>> Round[10, Pi]  
3π
```

Round complex numbers

```
>> Round[6/(2 + 3 I)]  
1 - I  
>> Round[1 + 2 I, 2 I]  
2I
```

Round Negative numbers too

```
>> Round[-1.4]  
-1
```

Expressions other than numbers remain unevaluated:

```
>> Round[x]
Round[x]

>> Round[1.5, k]
Round[1.5, k]
```

40. Operations on Vectors

In mathematics and physics, a vector is a term that refers colloquially to some quantities that cannot be expressed by a single number. It is also a row or column of a matrix.

In computer science, it is an array data structure consisting of collection of elements identified by at least one array index or key.

In *Mathics3* vectors as are Lists. One never needs to distinguish between row and column vectors. As with other objects vectors can mix number and symbolic elements.

Vectors can be long, dense, or sparse.

Here are the grouping we have for Vector Operations:

Contents

Constructing Vectors	484	Curl	486	Projection	488
AngleVector	484	Norm	486	UnitVector	488
Mathematical Operations	485	Vector Space Operations	487	VectorAngle	489
Cross	485	KroneckerProduct	487		
		Normalize	487		

Constructing Vectors

Constructing Vectors

Functions for constructing lists of various sizes and structure.

See also Constructing Lists.

AngleVector

WMA link

```
AngleVector[phi]
    returns the point at angle phi on the unit circle.
AngleVector[{r, phi}]
    returns the point at angle phi on a circle of radius r.
AngleVector[{x, y}, phi]
    returns the point at angle phi on a circle of radius 1 centered at {x, y}.
AngleVector[{x, y}, {r, phi}]
    returns point at angle phi on a circle of radius r centered at {x, y}.
```

```
>> AngleVector[90 Degree]
{0,1}

>> AngleVector[{1, 10}, a]
{1 + Cos[a], 10 + Sin[a]}
```

Mathematical Operations

Mathematical Operation

Cross

Cross product (SymPy, WMA)

```
Cross[a, b]
```

computes the vector cross product of a and b .

Three-dimensional cross product:

```
>> Cross[{x1, y1, z1}, {x2, y2, z2}]  
{y1z2 - y2z1, -x1z2 + x2z1, x1y2 - x2y1}
```

Cross is antisymmetric, so:

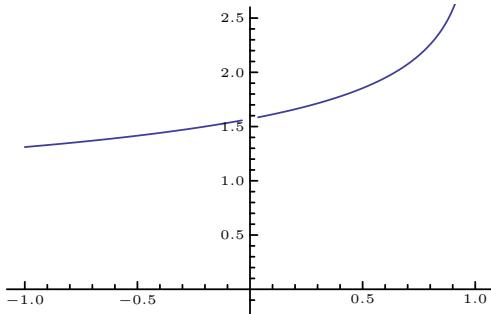
```
>> Cross[{x, y}]  
{-y, x}
```

Graph two-Dimensional cross product:

```
>> v1 = {1, Sqrt[3]}; v2 = Cross[v1]  
{-Sqrt[3], 1}
```

Visualize this:

```
>> Graphics[{Arrow[{{0, 0}, v1}], Red, Arrow[{{0, 0}, v2}]}, Axes -> True]
```



```
>> Cross[{1, 2}, {3, 4, 5}]
```

The arguments are expected to be vectors of equal length, and the number of arguments is expected to be 1 less than their length.

```
Cross[{1,2}, {3,4,5}]
```

Curl

Curl (SymPy, WMA)

```
Curl[{f1, f2}, {x1, x2}]
```

returns the curl $df2/dx1 - df1/dx2$

```
Curl[{f1, f2, $f3} {x1, x2, $x3}]
```

returns the curl $(df3/dx2 - df2/dx3, dx3/df3 - df3/dx1, df2/df1 - df1/dx2)$

Two-dimensional *Curl*:

```
>> Curl[{y, -x}, {x, y}]
-2
>> v[x_, y_] := {Cos[x] Sin[y], Cos[y] Sin[x]}
>> Curl[v[x, y], {x, y}]
0
```

Three-dimensional *Curl*:

```
>> Curl[{y, -x, 2 z}, {x, y, z}]
{0, 0, -2}
```

Norm

Matrix norms induced by vector p-norms (SymPy, WMA)

```
Norm[m, p]
computes the p-norm of matrix m.
Norm[m]
computes the 2-norm of matrix m.
```

The *Norm* of a vector is its Euclidean distance:

```
>> Norm[{x, y, z}]

$$\sqrt{\text{Abs}[x]^2 + \text{Abs}[y]^2 + \text{Abs}[z]^2}$$

```

By default, 2-norm is used for vectors, but you can be explicit:

```
>> Norm[{3, 4}, 2]
5
```

The 1-norm is the sum of the values:

```
>> Norm[{10, 100, 200}, 1]
310
>> Norm[{x, y, z}, Infinity]
Max [{Abs[x], Abs[y], Abs[z]}]
>> Norm[{-100, 2, 3, 4}, Infinity]
100
```

For complex numbers, *Norm*[z] is *Abs*[z]:

```
>> Norm[1 + I]

$$\sqrt{2}$$

```

So the norm is always real, even when the input is complex.

Norm[m, "Frobenius"] gives the Frobenius norm of m:

```
>> Norm[Array[Subscript[a, ##] &, {2, 2}], "Frobenius"]

$$\sqrt{\text{Abs}[a_{1,1}]^2 + \text{Abs}[a_{1,2}]^2 + \text{Abs}[a_{2,1}]^2 + \text{Abs}[a_{2,2}]^2}$$

```

Vector Space Operations

Vector Space Operation

KroneckerProduct

Kronecker product (SymPy, WMA)

```
KroneckerProduct[m1, m2, ...]  
    returns the Kronecker product of the arrays mi
```

Show symbolically how the Kronecker product works on two two-dimensional arrays:

```
>> a = {{a11, a12}, {a21, a22}}; b = {{b11, b12}, {b21, b22}};
```

```
>> KroneckerProduct[a, b]
```

```
{a11b11, a11b12, a12b11, a12b12}, {a11b21, a11b22, a12b21, a12b22}, {a21b11, a21b12, a22b11, a22b12}, {a21b21, a21b22, a22b21, a22b22}
```

Now do the same with discrete values:

```
>> a = {{0, 1}, {-1, 0}}; b = {{1, 2}, {3, 4}};
```

```
>> KroneckerProduct[a, b] // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \\ -1 & -2 & 0 & 0 \\ -3 & -4 & 0 & 0 \end{pmatrix}$$

Normalize

WMA link

```
Normalize[v]  
    calculates the normalized vector v.  
Normalize[z]  
    calculates the normalized complex number z.
```

```
>> Normalize[{1, 1, 1, 1}]
```

$$\left\{ \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right\}$$

```
>> Normalize[1 + I]
```

$$\left(\frac{1}{2} + \frac{I}{2} \right) \sqrt{2}$$

Projection

WMA link

```
Projection[u, v]  
    gives the projection of the vector u onto v
```

For vectors u and v , the projection is taken to be $(v \cdot u / v \cdot v)v$

For complex vectors u and v , the projection is taken to be $(v^* \cdot u / v^* \cdot v)v$ where v^* is `Conjugate[v]`.

Projection of two three-dimensional Integer vectors:

```
>> Projection[{5, 6, 7}, {1, 0, 0}]  
{5,0,0}
```

Projection of two two-dimensional Integer vectors:

```
>> Projection[{2, 3}, {1, 2}]  
{8/5, 16/5}
```

Projection of a machine-precision vector onto another:

```
>> Projection[{1.3, 2.1, 3.1}, {-0.3, 4.2, 5.3}]  
{ -0.162767, 2.27874, 2.87556}
```

Projection of two complex vectors:

```
>> Projection[{3 + I, 2, 2 - I}, {2, 4, 5 I}]  
{2/5 - 16I/45, 4/5 - 32I/45, 8/9 + I}
```

Project a symbol vector onto a numeric vector:

```
>> Projection[{a, b, c}, {1, 1, 1}]  
{(a+b+c)/3, (a+b+c)/3, (a+b+c)/3}
```

The projection of vector u onto vector v is in the direction of v :

```
>> {u, v} = RandomReal[1, {2, 6}];  
  
>> Abs[VectorAngle[Projection[u, v], v]] < 0. + 10^-7  
True
```

UnitVector

Unit vector (WMA)

```
UnitVector[n, k]  
    returns the  $n$ -dimensional unit vector with a 1 in position  $k$ .  
UnitVector[k]  
    is equivalent to UnitVector[2,  $k$ ].
```

```
>> UnitVector[2]  
{0,1}  
  
>> UnitVector[4, 3]  
{0,0,1,0}
```

VectorAngle

WMA link

```
VectorAngle[u, v]  
    gives the angles between vectors  $u$  and  $v$ 
```

```
>> VectorAngle[{1, 0}, {0, 1}]
 $\frac{\pi}{2}$ 
>> VectorAngle[{1, 2}, {3, 1}]
 $\frac{\pi}{4}$ 
>> VectorAngle[{1, 1, 0}, {1, 0, 1}]
 $\frac{\pi}{3}$ 
```

41. Options Management

A number of functions have various options which control the behavior or the default behavior that function. Default options can be queried or set.

WMA link

Contents

All	490	None	492	OptionValue	493
Default	491	NotOptionQ	492	Options	494
FilterRules	491	OptionQ	493	SetOptions	495

All

WMA link

All

is an option value for a number of functions indicating to include everything.

In list functions, it indicates all levels of the list.

For example, in Part of section 33, All, extracts into a first column vector the first element of each of the list elements:

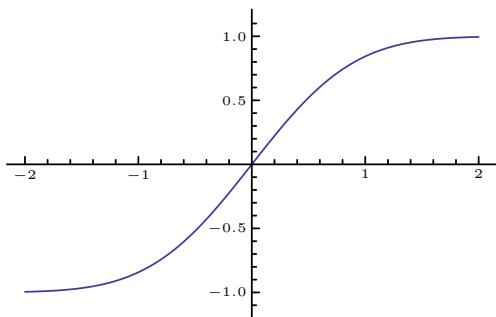
```
>> {{1, 3}, {5, 7}}[[All, 1]]  
{1,5}
```

While in Take of section 33, All extracts as a column matrix the first element as a list for each of the list elements:

```
>> Take[{{1, 3}, {5, 7}}, All, {1}]  
{ {1}, {5} }
```

In Plot of section 24 </url>, setting the Mesh of section 13 option to All will show the specific plot points:

```
>> Plot[x^2, {x, -1, 1}, MaxRecursion->5, Mesh->All]
```



Default

WMA link

```
Default[f]
  gives the default value for an omitted parameter of f.
Default[f, k]
  gives the default value for a parameter on the kth position.
Default[f, k, n]
  gives the default value for the kth parameter out of n.
```

Assign values to Default to specify default values.

```
>> Default[f] = 1
1
>> f[x_.] := x ^ 2
>> f[]
1
```

Default values are stored in DefaultValues:

```
>> DefaultValues[f]
{HoldPattern[Default[f]] :> 1}
```

You can use patterns for *k* and *n*:

```
>> Default[h, k_, n_] := {k, n}
```

Note that the position of a parameter is relative to the pattern, not the matching expression:

```
>> h[] /. h[___, ___, x_, y_, ___] -> {x, y}
{{3, 5}, {4, 5}}
```

FilterRules

WMA link

```
FilterRules[rules, pattern]
  gives those rules that have a left side that matches pattern.
FilterRules[rules, {pattern1, pattern2, ...}]
  gives those rules that have a left side that match at least one of pattern1, pattern2, ...
```

```
>> FilterRules[{x -> 100, y -> 1000}, x]
{x -> 100}
>> FilterRules[{x -> 100, y -> 1000, z -> 10000}, {a, b, x, z}]
{x -> 100, z -> 10000}
```

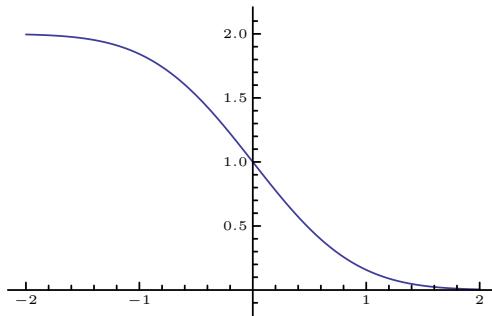
None

WMA link

```
None
is a setting value for many options.
```

Plot3D shows the mesh grid between computed points by default. This is the Mesh option value. However, you hide the mesh by setting the Mesh option value to None:

```
>> Plot3D[{x^2 + y^2, -x^2 - y^2}, {x, -2, 2}, {y, -2, 2}, BoxRatios->Automatic, Mesh->None]
```



NotOptionQ

WMA link

NotOptionQ[expr]

returns True if *expr* does not have the form of a valid option specification.

```
>> NotOptionQ[x]
True
>> NotOptionQ[2]
True
>> NotOptionQ["abc"]
True
>> NotOptionQ[a -> True]
False
```

OptionQ

WMA link

OptionQ[expr]

returns True if *expr* has the form of a valid option specification.

Examples of option specifications:

```
>> OptionQ[a -> True]
True
>> OptionQ[a :> True]
True
>> OptionQ[{a -> True}]
True
```

```

>> OptionQ[{a :> True}]
True

Options lists are flattened when are applied, so
>> OptionQ[{a -> True, {b->1, "c"->2}}]
True

>> OptionQ[{a -> True, {b->1, c}}]
False

>> OptionQ[{a -> True, F[b->1,c->2]}]
False

OptionQ returns False if its argument is not a valid option specification:
>> OptionQ[x]
False

```

OptionValue

WMA link

OptionValue[name] gives the value of the option <i>name</i> as specified in a call to a function with OptionsPattern.
OptionValue[f, name] recover the value of the option <i>name</i> associated to the symbol <i>f</i> .
OptionValue[f, optvals, name] recover the value of the option <i>name</i> associated to the symbol <i>f</i> , extracting the values from <i>optvals</i> if available.
OptionValue[..., list] recover the value of the options in <i>list</i> .

```

>> f[a->3] /. f[OptionsPattern[{ }]] -> {OptionValue[a]}
{3}

```

Unavailable options generate a message:

```

>> f[a->3] /. f[OptionsPattern[{ }]] -> {OptionValue[b]}
Option name b not found.
{b}

```

The argument of OptionValue must be a symbol:

```

>> f[a->3] /. f[OptionsPattern[{ }]] -> {OptionValue[a+b]}
Argument a + b at position 1 is expected to be a symbol.
{OptionValue[a + b]}

```

However, it can be evaluated dynamically:

```

>> f[a->5] /. f[OptionsPattern[{ }]] -> {OptionValue[Symbol["a"]]}
{5}

```

Options

WMA link

`Options[f]`
gives a list of optional arguments to f and their default values.

You can assign values to `Options` to specify options.

```
>> Options[f] = {n -> 2}
{n->2}

>> Options[f]
{n:>2}

>> f[x_, OptionsPattern[f]] := x ^ OptionValue[n]

>> f[x]
x2

>> f[x, n -> 3]
x3
```

Delayed option rules are evaluated just when the corresponding `OptionValue` is called:

```
>> f[a :> Print["value"]] /. f[OptionsPattern[{a}]] :> (OptionValue[a];
Print["between"]; OptionValue[a]);
value
between
value
```

In contrast to that, normal option rules are evaluated immediately:

```
>> f[a -> Print["value"]] /. f[OptionsPattern[{a}]] :> (OptionValue[a];
Print["between"]; OptionValue[a]);
value
between
```

Options must be rules or delayed rules:

```
>> Options[f] = {a}
{a} is not a valid list of option rules.

{a}
```

A single rule need not be given inside a list:

```
>> Options[f] = a -> b
a->b

>> Options[f]
{a:>b}
```

Options can only be assigned to symbols:

```
>> Options[a + b] = {a -> b}
Argument a + b at position 1 is expected to be a symbol.

{a->b}
```

SetOptions

WMA link

`SetOptions[s, name1 -> value1, name2 -> value2, ...]`
sets the specified default options for a symbol s . The entire set of options for s is returned.

One way to find the default options for a symbol is to use `SetOptions` passing no association pairs:

```
>> SetOptions[Plot]
{AspectRatio ->  $\frac{1}{\text{GoldenRatio}}$ , Axes -> True, AxesStyle
 -> {}, Background -> Automatic, Exclusions -> Automatic, ImageSize
 -> Automatic, LabelStyle -> {}, MaxRecursion
 -> Automatic, Mesh -> None, PlotPoints -> None, PlotRange
 -> Automatic, PlotRangePadding -> Automatic, TicksStyle -> {}}
```

42. Physical and Chemical data

Contents

ElementData 497

ElementData

WMA link

```
ElementData["name", "property"]
```

gives the value of the *property* for the chemical specified by *name*.

```
ElementData[n, "property"]
```

gives the value of the *property* for the *n*th chemical element.

```
>> ElementData[74]  
Tungsten  
>> ElementData["He", "AbsoluteBoilingPoint"]  
4.22  
>> ElementData["Carbon", "IonizationEnergies"]  
{1086.5, 2352.6, 4620.5, 6222.7, 37831, 47277.}  
>> ElementData[16, "ElectronConfigurationString"]  
[Ne] 3s2 3p4  
>> ElementData[73, "ElectronConfiguration"]  
{ {2}, {2, 6}, {2, 6, 10}, {2, 6, 10, 14}, {2, 6, 3}, {2} }
```

The number of known elements:

```
>> Length[ElementData[All]]  
118
```

Some properties are not appropriate for certain elements:

```
>> ElementData["He", "ElectroNegativity"]  
Missing [NotApplicable]
```

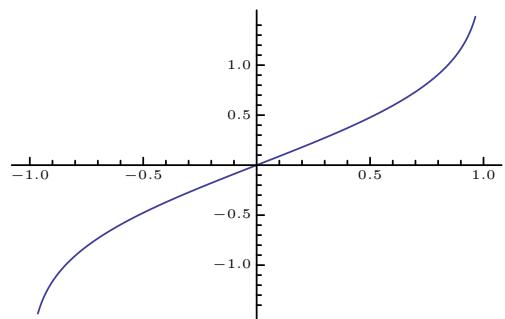
Some data is missing:

```
>> ElementData["Tc", "SpecificHeat"]  
Missing [NotAvailable]
```

All the known properties:

```
>> ElementData["Properties"]  
{Abbreviation, AbsoluteBoilingPoint, AbsoluteMeltingPoint, AtomicNumber, AtomicRadius, AtomicWeight, Block,
```

```
>> ListPlot[Table[ElementData[z, "AtomicWeight"], {z, 118}]]
```



43. Procedural Programming

Procedural programming is a programming paradigm, derived from imperative programming, based on the concept of the procedure call. This term is sometimes compared and contrasted with Functional Programming.

Procedures (a type of routine or subroutine) simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

Procedural functions are integrated into *Mathics3* symbolic programming environment.

Contents

Abort	498	Continue	499	Return	502
Break	498	Do	500	Switch	502
Catch	499	For	500	Throw	502
CompoundExpression (;)	499	If	501	Which	503
		Interrupt	501	While	503

Abort

WMA link

`Abort[]`

aborts an evaluation completely and returns `$Aborted`.

```
>> Print["a"]; Abort[]; Print["b"]
a
$Aborted
```

Break

WMA link

`Break[]`

exits a `For`, `While`, or `Do` loop.

```
>> n = 0;
>> While[True, If[n>10, Break[]]; n=n+1]
>> n
11
```

Catch

WMA link

`Catch[expr]`

returns the argument of the first Throw generated in the evaluation of *expr*.

`Catch[expr, form]`

returns value from the first Throw [*value*, *tag*] for which *form* matches *tag*.

`Catch[expr, form, f]`

returns *f*[*value*, *tag*].

Exit to the enclosing Catch as soon as Throw is evaluated:

```
>> Catch[r; s; Throw[t]; u; v]
      t
```

Define a function that can “throw an exception”:

```
>> f[x_] := If[x > 12, Throw[overflow], x!]
```

The result of Catch is just what is thrown by Throw:

```
>> Catch[f[1] + f[15]]
```

overflow

```
>> Catch[f[1] + f[4]]
```

25

CompoundExpression (;)

WMA link

`CompoundExpression[e1, e2, ...]`

e1; *e2*; ...

evaluates its arguments in turn, returning the last result.

```
>> a; b; c; d
      d
```

If the last argument is omitted, Null is taken:

```
>> a;
```

Continue

WMA link

`Continue[]`

continues with the next iteration in a For, While, or Do loop.

```
>> For[i=1, i<=8, i=i+1, If[Mod[i,2] == 0, Continue[], Print[i]]
      1
      3
      5
      7
```

Do

WMA link

```
Do[expr, {max}]
    evaluates expr max times.
Do[expr, {i, max}]
    evaluates expr max times, substituting i in expr with values from 1 to max.
Do[expr, {i, min, max}]
    starts with i = max.
Do[expr, {i, min, max, step}]
    uses a step size of step.
Do[expr, {i, {i1, i2, ...}}]
    uses values i1, i2, ... for i.
Do[expr, {i, imin, imax}, {j, jmin, jmax}, ...]
    evaluates expr for each j from jmin to jmax, for each i from imin to imax, etc.
```

```
>> Do[Print[i], {i, 2, 4}]
2
3
4

>> Do[Print[{i, j}], {i, 1, 2}, {j, 3, 5}]
{1, 3}
{1, 4}
{1, 5}
{2, 3}
{2, 4}
{2, 5}
```

You can use Break[] and Continue[] inside Do:

```
>> Do[If[i > 10, Break[], If[Mod[i, 2] == 0, Continue[]]; Print[i]], {i,
5, 20}]
5
7
9
```

For

WMA link

```
For[start, test, incr, body]
    evaluates start, and then iteratively body and incr as long as test evaluates to True.
For[start, test, incr]
    evaluates only incr and no body.
For[start, test]
    runs the loop without any body.
```

Compute the factorial of 10 using For:

```
>> n := 1

>> For[i=1, i<=10, i=i+1, n = n * i]

>> n
3628800

>> n == 10!
True
```

If

WMA link

```
If [cond, pos, neg]
    returns pos if cond evaluates to True, and neg if it evaluates to False.
If [cond, pos, neg, other]
    returns other if cond evaluates to neither True nor False.
If [cond, pos]
    returns Null if cond evaluates to False.
```

```
>> If[1<2, a, b]
      a
```

If the second branch is not specified, Null is taken:

```
>> If[1<2, a]
      a

>> If[False, a] //FullForm
      Null
```

You might use comments (* and *) to make the branches of If more readable:

```
>> If[a, (*then*)b, (*else*)c];
```

Interrupt

WMA link

```
Interrupt[]
    Interrupt an evaluation and returns $Aborted.
```

```
>> Print["a"]; Interrupt[]; Print["b"]
      a
      $Aborted
```

Return

WMA link

```
Return[expr]
    aborts a function call and returns expr.
```

```
>> f[x_] := (If[x < 0, Return[0]]; x)

>> f[-1]
      0

>> Do[If[i > 3, Return[]]; Print[i], {i, 10}]
      1
      2
      3
```

Return only exits from the innermost control flow construct.

```
>> g[x_] := (Do[If[x < 0, Return[0]], {i, 2, 1, 0, -1}]]; x)
```

```
>> g[-1]
-1
```

Switch

WMA link

```
Switch[expr, pattern1, value1, pattern2, value2, ...]
yields the first value for which expr matches the corresponding pattern.
```

```
>> Switch[2, 1, x, 2, y, 3, z]
y
>> Switch[5, 1, x, 2, y]
Switch [5,1,x,2,y]
>> Switch[5, 1, x, 2, a, _, b]
b
>> Switch[2, 1]
Switch called with 2 arguments. Switch must be called with an odd
number of arguments.
Switch [2,1]
```

Throw

WMA link

```
Throw['value']
stops evaluation and returns 'value' as the value of the nearest enclosing Catch.
Catch['value', 'tag']
is caught only by 'Catch[expr,form]', where tag matches form.
```

Using Throw can affect the structure of what is returned by a function:

```
>> NestList[#^2 + 1 &, 1, 7]
{1,2,5,26,677,458330,210066388901,44127887745906175987802}

>> Catch[NestList[If[# > 1000, Throw[], #^2 + 1] &, 1, 7]]
458330

>> Throw[1]
Uncaught Hold[Throw[1]] returned to top level.
Hold [Throw[1]]
```

Which

WMA link

```
Which[cond1, expr1, cond2, expr2, ...]
yields expr1 if cond1 evaluates to True, expr2 if cond2 evaluates to True, etc.
```

```

>> n = 5;
>> Which[n == 3, x, n == 5, y]
y
>> f[x_] := Which[x < 0, -x, x == 0, 0, x > 0, x]
>> f[-3]
3

```

If no test yields True, Which returns Null:

```
>> Which[False, a]
```

If a test does not evaluate to True or False, evaluation stops and a Which expression containing the remaining cases is returned:

```

>> Which[False, a, x, b, True, c]
Which[x, b, True, c]

```

Which must be called with an even number of arguments:

```

>> Which[a, b, c]
Which called with 3 arguments.
>> Which[a, b, c]

```

While

[WMA link](#)

<pre>While[test, body]</pre>	evaluates <i>body</i> as long as <i>test</i> evaluates to True.
<pre>While[test]</pre>	runs the loop without any body.

Compute the GCD of two numbers:

```

>> {a, b} = {27, 6};
>> While[b != 0, {a, b} = {b, Mod[a, b]}];
>> a
3

```

44. Rules and Patterns

The concept of transformation rules for arbitrary symbolic patterns is key in *Mathics3*.

Also, functions can get applied or transformed depending on whether or not functions arguments match.

Some examples: » $a + b + c \ /. \ a + b \rightarrow t = c + t$ » $a + 2 + b + c + x * y \ /. \ n_Integer + s_Symbol + rest_ \rightarrow \{n, s, rest\} = \{2, a, b + c + x y\}$ » $f[a, b, c, d] \ /. \ f[first_, rest_] \rightarrow \{first, \{rest\}\} = \{a, \{b, c, d\}\}$

Tests and Conditions: » $f[4] \ /. \ f[x_?(\# > 0 \&)] \rightarrow x^2 = 16$ » $f[4] \ /. \ f[x_] \ /; x > 0 \rightarrow x^2 = 16$

Elements in the beginning of a pattern rather match fewer elements: » $f[a, b, c, d] \ /. \ f[start_, end_] \rightarrow \{\{start\}, \{end\}\} = \{\{a\}, \{b, c, d\}\}$

Optional arguments using `Optional`: » $f[a] \ /. \ f[x_, y_:3] \rightarrow \{x, y\} = \{a, 3\}$

Options using `OptionsPattern` and `OptionValue`: » $f[y, a->3] \ /. \ f[x_, OptionsPattern[\{a->2, b->5\}]] \rightarrow \{x, OptionValue[a], OptionValue[b]\} = \{y, 3, 5\}$

The attributes `Flat`, `Orderless`, and `OneIdentity` affect pattern matching.

Contents

Alternatives ()	504	Longest	508	ReplaceAll (/.)	513
Blank	505	Optional (:).	508	ReplaceList	513
BlankNullSequence	505	OptionsPattern	509	ReplaceRepeated (//.).	514
BlankSequence	506	Pattern	510	Rule (->)	514
Condition (/;).	506	PatternTest (?).	510	RuleDelayed (:>).	514
Dispatch	507	Repeated (.).	511	Shortest	515
Except	507	RepeatedNull (.).	511	Verbatim	515
HoldPattern	507	Replace	512		

Alternatives (|)

WMA link

```
Alternatives[p1, p2, ..., pi]
p1 | p2 | ... | pi
is a pattern that matches any of the patterns 'p1, p2, ..., pi'.
```

```
>> a+b+c+d/.(a|b)->t
c + d + 2t
```

Alternatives can also be used for string expressions

```
>> StringReplace["0123 3210", "1" | "2" -> "X"]
0XX3 3XX0
```

Blank

WMA link

`Blank[]`

`_` represents any single expression in a pattern.

`Blank[h]`

`_h`

represents any expression with head h .

```
>> MatchQ[a + b, _]
```

True

Patterns of the form $_h$ can be used to test the types of objects:

```
>> MatchQ[42, _Integer]
```

True

```
>> MatchQ[1.0, _Integer]
```

False

```
>> {42, 1.0, x} /. {_Integer -> "integer", _Real -> "real"} // InputForm
{"integer", "real", x}
```

`Blank` only matches a single expression:

```
>> MatchQ[f[1, 2], f[_]]
```

False

BlankNullSequence

WMA link

`BlankNullSequence[]`

`---` represents any sequence of expression elements in a pattern, including an empty sequence.

`BlankNullSequence` is like `BlankSequence`, except it can match an empty sequence:

```
>> MatchQ[f[], f[___]]
```

True

BlankSequence

WMA link

`BlankSequence[]`

`--` represents any non-empty sequence of expression elements in a pattern.

`BlankSequence[h]`

`--h`

represents any sequence of elements, all of which have head h .

Use a `BlankSequence` pattern to stand for a non-empty sequence of arguments:

```
>> MatchQ[f[1, 2, 3], f[___]]
```

True

```

>> MatchQ[f[], f[_]]
False

--h will match only if all elements have head h:
>> MatchQ[f[1, 2, 3], f[_Integer]]
True

>> MatchQ[f[1, 2.0, 3], f[_Integer]]
False

```

The value captured by a named BlankSequence pattern is a Sequence object:

```

>> f[1, 2, 3] /. f[x__] -> x
Sequence[1, 2, 3]

```

Condition (/;)

WMA link

```

Condition[pattern, expr]
pattern /; expr
places an additional constraint on pattern that only allows it to match if expr evaluates to True .

```

The controlling expression of a Condition can use variables from the pattern:

```

>> f[3] /. f[x_] /; x>0 -> t
t

>> f[-3] /. f[x_] /; x>0 -> t
f[-3]

```

Condition can be used in an assignment:

```

>> f[x_] := p[x] /; x>0

>> f[3]
p[3]

>> f[-3]
f[-3]

```

Dispatch

WMA link

```

Dispatch[rulelist]
Introduced for compatibility. Currently, it just return rulelist. In the future, it should return an optimized DispatchRules atom, containing an optimized set of rules.

```

```

>> rules = {{a_,b_}->a^b, {1,2}->3., F[x_]->x^2};

>> F[2] /. rules
4

```

```

>> dispatchrules = Dispatch[rules]
Dispatch[{{a_, b_} -> a^b, {1, 2} -> 3., F[x_] -> x^2}]
>> F[2] /. dispatchrules
4

```

Except

WMA link

<pre>Except[c]</pre>	represents a pattern object that matches any expression except those matching <i>c</i> .
<pre>Except[c, p]</pre>	represents a pattern object that matches <i>p</i> but not <i>c</i> .

```

>> Cases[{x, a, b, x, c}, Except[x]]
{a, b, c}
>> Cases[{a, 0, b, 1, c, 2, 3}, Except[1, _Integer]]
{0, 2, 3}

```

Except can also be used for string expressions:

```

>> StringReplace["Hello world!", Except[LetterCharacter] -> ""]
Helloworld

```

HoldPattern

WMA link

<pre>HoldPattern[expr]</pre>	is equivalent to <i>expr</i> for pattern matching, but maintains it in an unevaluated form.
------------------------------	---

```

>> HoldPattern[x + x]
HoldPattern[x + x]
>> x /. HoldPattern[x] -> t
t

```

HoldPattern has attribute HoldAll:

```

>> Attributes[HoldPattern]
{HoldAll, Protected}

```

Longest

WMA link

<pre>Longest[pat]</pre>	is a pattern object that matches the longest sequence consistent with the pattern <i>p</i> .
-------------------------	--

```

>> StringCases["aabaaab", Longest["a" ~~ __ ~~ "b"]]
{aabaaab}

>> StringCases["aabaaab", Longest[RegularExpression["a+b"]]]
{aab, aaab}

```

Optional (:)

WMA link

```

Optional[patt, default]
patt : default
is a pattern which matches patt, which if omitted should be replaced by default.

```

```

>> f[x_, y_:1] := {x, y}

>> f[1, 2]
{1, 2}

>> f[a]
{a, 1}

```

Note that *symb* : *patt* represents a *Pattern* object. However, there is no disambiguity, since *symb* has to be a symbol in this case.

```

>> x:_ // FullForm
Pattern[x,Blank[]]

>> _:d // FullForm
Optional[Blank[],d]

>> x:_+y_:d // FullForm
Pattern[x,Plus[Blank[],Optional[Pattern[y,Blank[]],d]]]

```

s_. is equivalent to *Optional[s_]* and represents an optional parameter which, if omitted, gets its value from *Default*.

```

>> FullForm[s_.]
Optional[Pattern[s,Blank[]]]

>> Default[h, k_] := k

>> h[a] /. h[x_, y_.] -> {x, y}
{a, 2}

```

OptionsPattern

WMA link

`OptionsPattern[f]`

is a pattern that stands for a sequence of options given to a function, with default values taken from `Options[f]`. The options can be of the form `opt->value` or `opt:>value`, and might be in arbitrarily nested lists.

`OptionsPattern[{opt1->value1, ...}]`

takes explicit default values from the given list. The list may also contain symbols *f*, for which `Options[f]` is taken into account; it may be arbitrarily nested. `OptionsPattern[{}]` does not use any default values.

The option values can be accessed using `OptionValue`.

```
>> f[x_, OptionsPattern[{n->2}]] := x ^ OptionValue[n]
```

```
>> f[x]
```

x^2

```
>> f[x, n->3]
```

x^3

Delayed rules as options:

```
>> e = f[x, n:>a]
```

x^a

```
>> a = 5;
```

```
>> e
```

x^5

Options might be given in nested lists:

```
>> f[x, {{n->4}}]
```

x^4

Pattern

WMA link

`Pattern[symb, patt]`

`symb : patt`

assigns the name *symb* to the pattern *patt*.

`symb_head`

is equivalent to `symb : _head` (accordingly with `_` and `__`).

`symb : patt : default`

is a pattern with name *symb* and default value *default*, equivalent to `Optional[patt : symb, default]`.

```
>> FullForm[a_b]
```

`Pattern[a, Blank[b]]`

```
>> FullForm[a:_b]
```

`Optional[Pattern[a, Blank[]], b]`

`Pattern` has attribute `HoldFirst`, so it does not evaluate its name:

```
>> x = 2
```

2

```
>> x_
x_
```

Nested Pattern assign multiple names to the same pattern. Still, the last parameter is the default value.

```
>> f[y] /. f[a:b, _ : d] -> {a, b}
f [y]
```

This is equivalent to:

```
>> f[a] /. f[a:_ : b] -> {a, b}
{a, b}
```

FullForm:

```
>> FullForm[a:b:c:d:e]
Optional[Pattern[a,b],Optional[Pattern[c,d],e]]
>> f[] /. f[a:_ : b] -> {a, b}
{b, b}
```

PatternTest (?)

WMA link

```
PatternTest[pattern, test]
pattern ? test
constrains pattern to match expr only if the evaluation of test[expr] yields True.
```

```
>> MatchQ[3, _Integer?(#>0&)]
True
>> MatchQ[-3, _Integer?(#>0&)]
False
>> MatchQ[3, Pattern[3]]
First element in pattern Pattern[3] is not a valid pattern name.
False
```

Repeated (..)

WMA link

```
Repeated[pattern]
matches one or more occurrences of pattern.
```

```
>> a_Integer.. // FullForm
Repeated[Pattern[a, Blank[Integer]]]
>> 0..1//FullForm
Repeated[0]
```

```

>> { {}, {a}, {a, b}, {a, a, a}, {a, a, a, a} } /. {Repeated[x : a | b,
3]} -> x
{ {}, a, {a, b} , a, {a, a, a, a} }

>> f[x, 0, 0, 0] /. f[x, s:0..] -> s
Sequence[0,0,0]

```

RepeatedNull (. . .)

WMA link

RepeatedNull[*pattern*]
 matches zero or more occurrences of *pattern*.

```

>> a___Integer...//FullForm
RepeatedNull [Pattern [a, BlankNullSequence [Integer]]]

>> f[x] /. f[x, 0...] -> t
t

```

Replace

WMA link

Replace[*expr*, *x* -> *y*]
 yields the result of replacing *expr* with *y* if it matches the pattern *x*.
Replace[*expr*, *x* -> *y*, *levelspec*]
 replaces only subexpressions at levels specified through *levelspec*.
Replace[*expr*, {*x* -> *y*, ...}]
 performs replacement with multiple rules, yielding a single result expression.
Replace[*expr*, {{*a* -> *b*, ...}, {*c* -> *d*, ...}, ...}]
 returns a list containing the result of performing each set of replacements.

```

>> Replace[x, {x -> 2}]
2

```

By default, only the top level is searched for matches

```

>> Replace[1 + x, {x -> 2}]
1 + x
>> Replace[x, {{x -> 1}, {x -> 2}}]
{1,2}

```

Replace stops after the first replacement

```

>> Replace[x, {x -> {}, _List -> y}]
{}

```

Replace replaces the deepest levels first

```

>> Replace[x[1], {x[1] -> y, 1 -> 2}, All]
x[2]

```

By default, heads are not replaced

```
>> Replace[x[x[y]], x -> z, All]
x [x [y]]
```

Heads can be replaced using the Heads option

```
>> Replace[x[x[y]], x -> z, All, Heads -> True]
z [z [y]]
```

Note that heads are handled at the level of elements

```
>> Replace[x[x[y]], x -> z, {1}, Heads -> True]
z [x [y]]
```

You can use Replace as an operator

```
>> Replace[{x_ -> x + 1}] [10]
11
```

ReplaceAll (/.)

WMA link

```
ReplaceAll[expr, x -> y]
expr /. x -> y
    yields the result of replacing all subexpressions of expr matching the pattern x with y.
expr /. {x -> y, ...}
    performs replacement with multiple rules, yielding a single result expression.
expr /. {{a -> b, ...}, {c -> d, ...}, ...}
    returns a list containing the result of performing each set of replacements.
```

```
>> a+b+c /. c->d
a + b + d
>> g[a+b+c,a]/.g[x_+y_,x_]->{x,y}
{a,b + c}
```

If *rules* is a list of lists, a list of all possible respective replacements is returned:

```
>> {a, b} /. {{a->x, b->y}, {a->u, b->v}}
{{x,y},{u,v}}
```

The list can be arbitrarily nested:

```
>> {a, b} /. {{{a->x, b->y}, {a->w, b->z}}, {a->u, b->v}}
{{{x,y},{w,z}},{u,v}}
>> {a, b} /. {{{a->x, b->y}, a->w, b->z}, {a->u, b->v}}
Elements of {{a -> x, b -> y}, a -> w, b -> z} are a mixture of lists
and nonlists.
{{a,b}/.{{a -> x,b -> y},a -> w,b -> z},{u,v}}
```

ReplaceAll also can be used as an operator:

```
>> ReplaceAll[{a -> 1}] [{a, b}]
{1,b}
```

ReplaceAll replaces the shallowest levels first:

```
>> ReplaceAll[x[1], {x[1] -> y, 1 -> 2}]
y
```

ReplaceList

WMA link

`ReplaceList[expr, rules]`

returns a list of all possible results of applying *rules* to *expr*.

Get all subsequences of a list:

```
>> ReplaceList[{a, b, c}, {___, x___, ___} -> {x}]
{{a}, {a,b}, {a,b,c}, {b}, {b,c}, {c}}
```

You can specify the maximum number of items:

```
>> ReplaceList[{a, b, c}, {___, x___, ___} -> {x}, 3]
{{a}, {a,b}, {a,b,c}}
```



```
>> ReplaceList[{a, b, c}, {___, x___, ___} -> {x}, 0]
{}
```

If no rule matches, an empty list is returned:

```
>> ReplaceList[a, b->x]
{}
```

Like in `ReplaceAll`, *rules* can be a nested list:

```
>> ReplaceList[{a, b, c}, {{___, x___, ___} -> {x}}, {{a, b, c} -> t}], 2]
{{{a}, {a,b}}, {t}}
```



```
>> ReplaceList[expr, {}, -1]
Non-negative integer or Infinity expected at position 3.
```

ReplaceList[expr, {}, -1]

Possible matches for a sum:

```
>> ReplaceList[a + b + c, x_ + y_ -> {x, y}]
{{a, b + c}, {b, a + c}, {c, a + b}, {a + b, c}, {a + c, b}, {b + c, a}}
```

ReplaceRepeated (//.)

WMA link

`ReplaceRepeated[expr, x -> y]`

`expr // . x -> y`

repeatedly applies the rule $x \rightarrow y$ to *expr* until the result no longer changes.

```
>> a+b+c // . c->d
a + b + d
>> f = ReplaceRepeated[c->d];
```

```

>> f[a+b+c]
a + b + d
>> Clear[f];

```

Simplification of logarithms:

```

>> logrules = {Log[x_ * y_] :> Log[x] + Log[y], Log[x_ ^ y_] :> y * Log[x]};

>> Log[a * (b * c)^d ^ e * f] // . logrules
Log[a] + Log[f] + (Log[b] + Log[c]) d^e

```

ReplaceAll just performs a single replacement:

```

>> Log[a * (b * c)^d ^ e * f] /. logrules
Log[a] + Log[f (bc)^d^e]

```

Rule (->)

WMA link

```

Rule[x, y]
x -> y
represents a rule replacing x with y.

```

```

>> a+b+c /. c->d
a + b + d
>> {x,x^2,y} /. x->3
{3,9,y}

```

RuleDelayed (:>)

WMA link

```

RuleDelayed[x, y]
x :> y
represents a rule replacing x with y, with y held unevaluated.

```

```

>> Attributes[RuleDelayed]
{HoldRest, Protected, SequenceHold}

```

Shortest

WMA link

```

Shortest[pat]
is a pattern object that matches the shortest sequence consistent with the pattern p.

```

```
>> StringCases["aabaaab", Shortest["a" ~~__ ~~"b"]]
{aab, aaab}

>> StringCases["aabaaab", Shortest[RegularExpression["a+b"]]]
{aab, aaab}
```

Verbatim

WMA link

Verbatim[*expr*]

prevents pattern constructs in *expr* from taking effect, allowing them to match themselves.

Create a pattern matching Blank:

```
>> _ /. Verbatim[_] ->t
      t

>> x /. Verbatim[_] ->t
      x
```

Without Verbatim, Blank has its normal effect:

```
>> x /. _ ->t
      t
```

45. Scoping Constructs

Contents

\$Context	516	Begin	517	EndPackage	518
\$ContextPath	516	BeginPackage	517	Module	519
\$ContextPathStack	516	Block	518	Unique	519
\$ContextStack	517	Contexts	518	With	519
\$ModuleNumber	517	End	518		

\$Context

[WMA link](#)

\$Context
is the current context.

```
>> $Context  
Global'
```

\$ContextPath

[WMA link](#)

\$ContextPath
is the search path for contexts.

```
>> $ContextPath // InputForm  
{“System”, “Global”}
```

\$ContextPathStack

[WMA link](#)

System`Private`\$ContextPathStack
is an internal variable tracking the values of \$ContextPath saved by Begin and BeginPackage.

\$ContextStack

[WMA link](#)

`System`Private`$ContextStack`
is an internal variable tracking the values of `$Context` saved by `Begin` and `BeginPackage`.

\$ModuleNumber

WMA link

`$ModuleNumber`
is the current “serial number” to be used for local module variables.

 \$ModuleNumber is incremented every time `Module` or `Unique` is called. a Mathics session starts with `$ModuleNumber` set to 1. You can reset `$ModuleNumber` to a positive machine integer, but if you do so, naming conflicts may lead to inefficiencies.

Begin

WMA link

`Begin[context]`
temporarily sets the current context to *context*.

```
>> Begin["test`"]
   test'

>> End[]
   test'

>> End[]
   No previous context defined.

Global'
```

BeginPackage

WMA link

`BeginPackage[context]`
starts the package given by *context*.

The *context* argument must be a valid context name. `BeginPackage` changes the values of `$Context` and `$ContextPath`, setting the current context to *context*.

Block

WMA link

`Block[{x, y, ...}, expr]`
temporarily removes the definitions of the given variables, evaluates *expr*, and restores the original definitions afterwards.
`Block[{x=x0, y=y0, ...}, expr]`
assigns temporary values to the variables during the evaluation of *expr*.

```
>> n = 10
10
>> Block[{n = 5}, n ^ 2]
25
>> n
10
```

Values assigned to block variables are evaluated at the beginning of the block. Keep in mind that the result of `Block` is evaluated again, so a returned block variable will get its original value.

```
>> Block[{x = n+2, n}, {x, n}]
{12, 10}
```

If the variable specification is not of the described form, an error message is raised.

```
>> Block[{x + y}, x]
Local variable specification contains x + y, which is not a symbol or
an assignment to a symbol.
x
```

Variable names may not appear more than once:

```
>> Block[{x, x}, x]
Duplicate local variable x found in local variable specification.
x
```

Contexts

WMA link

`Contexts[]`
yields a list of all contexts.

End

WMA link

`End[]`
ends a context started by `Begin`.

EndPackage

WMA link

`EndPackage[]`
marks the end of a package, undoing a previous `BeginPackage`.

After `EndPackage`, the values of `$Context` and `$ContextPath` at the time of the `BeginPackage` call are restored, with the new package's context prepended to `$ContextPath`.

Module

WMA link

`Module[{vars}, expr]`

localizes variables by giving them a temporary name of the form `name$number`, where `number` is the current value of `$ModuleNumber`. Each time a module is evaluated, `$ModuleNumber` is incremented.

Unique

WMA link

`Unique[]`

generates a new symbol and gives a name of the form `$number`.

`Unique[x]`

generates a new symbol and gives a name of the form `x$number`.

`Unique[{x, y, ...}]`

generates a list of new symbols.

`Unique[{"xxx"}]`

generates a new symbol and gives a name of the form `xxxnumber`.

Create a unique symbol with no particular name:

```
>> Unique[]  
$3
```

Create a unique symbol whose name begins with x:

```
>> Unique["x"]  
x4
```

With

WMA link

`With[{x=x0, y=y0, ...}, expr]`

specifies that all occurrences of the symbols `x, y, ...` in `expr` should be replaced by `x0, y0, ...`

46. Solving Recurrence Equations

Contents

RSolve 520

RSolve

WMA link

RSolve[*eqn*, *a*[*n*], *n*]
solves a recurrence equation for the function *a*[*n*].

Solve a difference equation:

```
>> RSolve[a[n] == a[n+1], a[n], n]
{{a[n] -> C[0]}}
```

No boundary conditions gives two general parameters:

```
>> RSolve[{a[n + 2] == a[n]}, a, n]
{{a -> (Function[{n}, C[0] + C[1](-1)^n])}}
```

Include one boundary condition:

```
>> RSolve[{a[n + 2] == a[n], a[0] == 1}, a, n]
{{a -> (Function[{n}, C[0] + (1 - C[0])(-1)^n])}}
```

Get a “pure function” solution for *a* with two boundary conditions:

```
>> RSolve[{a[n + 2] == a[n], a[0] == 1, a[1] == 4}, a, n]
{{a -> (Function[{n}, 5/2 - 3(-1)^n/2])}}
```

47. Sparse Array Functions

Contents

SparseArray 521

SparseArray

WMA link

SparseArray[*rules*]

Builds a sparse array according to the list of *rules*.

SparseArray[*rules*, *dims*]

Builds a sparse array of dimensions *dims* according to the *rules*.

SparseArray[*list*]

Builds a sparse representation of *list*.

```
>> SparseArray[{{1, 2} -> 1, {2, 1} -> 1}]  
SparseArray[Automatic, {2, 2}, 0, {{1, 2} -> 1, {2, 1} -> 1}]  
  
>> SparseArray[{{1, 2} -> 1, {2, 1} -> 1}, {3, 3}]  
SparseArray[Automatic, {3, 3}, 0, {{1, 2} -> 1, {2, 1} -> 1}]  
  
>> M=SparseArray[{{0, a}, {b, 0}}]  
SparseArray[Automatic, {2, 2}, 0, {{1, 2} -> a, {2, 1} -> b}]  
  
>> M //Normal  
{0, a}, {b, 0}
```

48. Special Functions

There are a number of functions found in mathematical physics and found in standard handbooks. One thing to note is that the technical literature often contains several conflicting definitions. So beware and check for conformance with the Mathics documentation.

A number of special functions can be evaluated for arbitrary complex values of their arguments. However defining relations may apply only for some special choices of arguments. Here, the full function corresponds to an extension or “analytic continuation” of the defining relation.

For example, integral representations of functions are only valid when the integral exists, but the functions can usually be defined by analytic continuation.

Contents

Bessel and Related Functions	522	SphericalHankelH2	531	Factorial (!)	539
AiryAi	523	StruveH	532	Factorial2 (!!)	540
AiryAiPrime	523	StruveL	532	Gamma	540
AiryAiZero	524	WeberE	533	LogGamma	541
AiryBi	524	Elliptic Integrals	533	Pochhammer	542
AiryBiPrime	524	EllipticE	533	PolyGamma	542
AiryBiZero	525	EllipticF	534	StieltjesGamma	542
AngerJ	525	EllipticK	534	Orthogonal Polynomials	542
BesselII	526	EllipticPi	535	ChebyshevT	542
BesselJ	526	Error Function and Related Functions	535	ChebyshevU	543
BesselJZero	526	Erf	535	GegenbauerC	543
BesselK	527	Erfc	536	HermiteH	543
BesselY	527	FresnelC	536	JacobiP	544
BesselYZero	527	FresnelS	536	LaguerreL	544
HankelH1	528	InverseErf	537	LegendreP	545
HankelH2	528	InverseErfc	537	LegendreQ	545
KelvinBei	528	Exponential Integral and Special Functions	537	SphericalHarmonicY	545
KelvinBer	529	ExpIntegralE	537	Zeta Functions and Polylogarithms	545
KelvinKei	530	ExpIntegralEi	538	LerchPhi	546
KelvinKer	530	ProductLog	538	Zeta	546
SphericalBesselJ	530	Gamma and Related Functions	538		
SphericalBesselY	531	Beta	539		
SphericalHankelH1	531				

Bessel and Related Functions

Bessel and Related Function

AiryAi

Airy function of the first kind (SymPy, WMA)

AiryAi[x]

returns the Airy function $\text{Ai}(x)$.

Exact values:

```
>> AiryAi[0]
```

$$\frac{3^{\frac{1}{3}}}{3\text{Gamma}\left[\frac{2}{3}\right]}$$

AiryAi can be evaluated numerically:

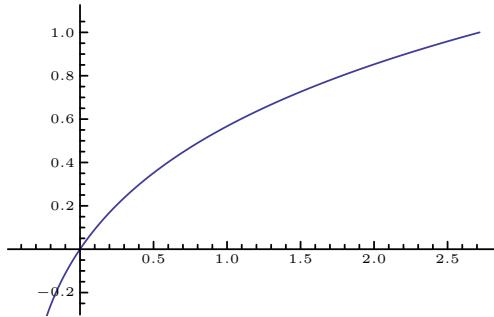
```
>> AiryAi[0.5]
```

0.231694

```
>> AiryAi[0.5 + I]
```

$0.157118 - 0.24104I$

```
>> Plot[AiryAi[x], {x, -10, 10}]
```



AiryAiPrime

Derivative of Airy function (Sympy, WMA link)

AiryAiPrime[x]

returns the derivative of the Airy function $\text{AiryAi}[x]$.

Exact values:

```
>> AiryAiPrime[0]
```

$$-\frac{3^{\frac{2}{3}}}{3\text{Gamma}\left[\frac{1}{3}\right]}$$

Numeric evaluation:

```
>> AiryAiPrime[0.5]
```

-0.224911

AiryAiZero

WMA link

AiryAiZero[k]

returns the k th zero of the Airy function $\text{Ai}(z)$.

```
>> N[AiryAiZero[1]]
- 2.33811
```

AiryBi

WMA link

AiryBi[x]
returns the Airy function of the second kind $\text{Bi}(x)$.

Exact values:

```
>> AiryBi[0]
```

$$\frac{3^{\frac{5}{6}}}{3\Gamma\left[\frac{2}{3}\right]}$$

Numeric evaluation:

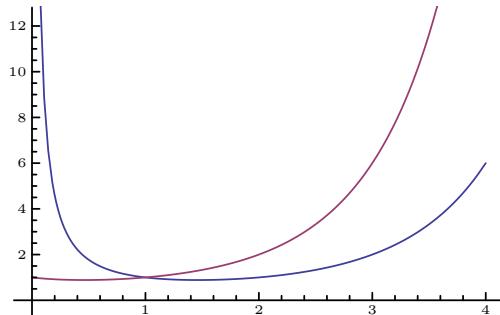
```
>> AiryBi[0.5]
```

$$0.854277$$

```
>> AiryBi[0.5 + I]
```

$$0.688145 + 0.370815I$$

```
>> Plot[AiryBi[x], {x, -10, 2}]
```



AiryBiPrime

WMA link

AiryBiPrime[x]
returns the derivative of the Airy function of the second kind $\text{AiryBi}[x]$.

Exact values:

```
>> AiryBiPrime[0]
```

$$\frac{3^{\frac{1}{6}}}{\Gamma\left[\frac{1}{3}\right]}$$

Numeric evaluation:

```
>> AiryBiPrime[0.5]
```

$$0.544573$$

AiryBiZero

WMA link

AiryBiZero [k]

returns the k th zero of the Airy function $\text{Bi}(z)$.

```
>> N[AiryBiZero[1]]  
- 1.17371
```

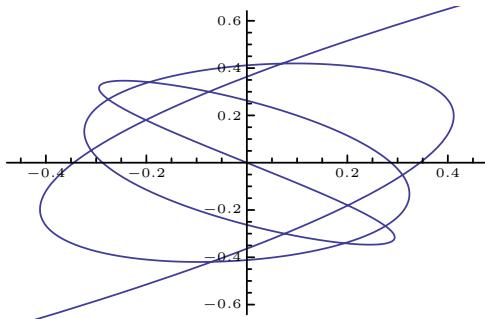
AngerJ

Anger function (mpmath, WMA)

AngerJ [n , z]

returns the Anger function $J_n(z)$.

```
>> AngerJ[1.5, 3.5]  
0.294479  
>> Plot[AngerJ[1, x], {x, -10, 10}]
```



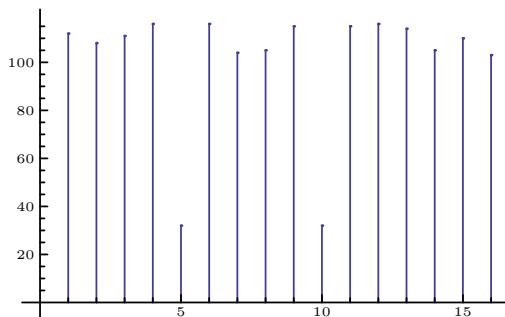
BesselI

Modified Bessel function of the first kind (Sympy, WMA)

BesselI [n , z]

returns the modified Bessel function of the first kind $I_n(z)$.

```
>> BesselI[1.5, 4]  
8.17263  
>> Plot[BesselI[0, x], {x, 0, 5}]
```



BesselJ

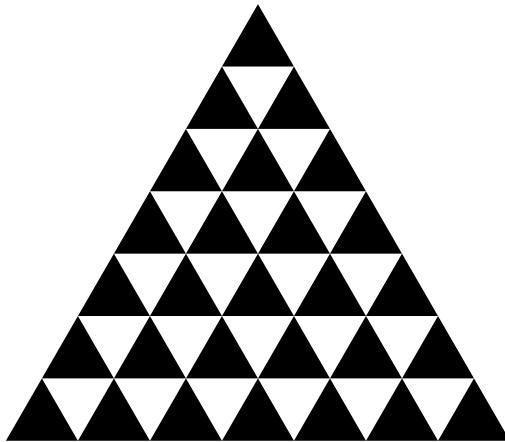
Bessel function of the first kind (SymPy, WMA)

```
BesselJ[n, z]
    returns the Bessel function of the first kind J_n(z).
```

```
>> BesselJ[0, 5.2]
- 0.11029

>> D[BesselJ[n, z], z]
- BesselJ[1 + n, z] + BesselJ[-1 + n, z]
  2
  2

>> Plot[BesselJ[0, x], {x, 0, 10}]
```



BesselJZero

WMA link

```
BesselJZero[n, k]
    returns the kth zero of the Bessel function of the first kind J_n(z).
```

```
>> N[BesselJZero[0, 1]]
2.40483

>> N[BesselJZero[0, 1], 10]
2.404825558
```

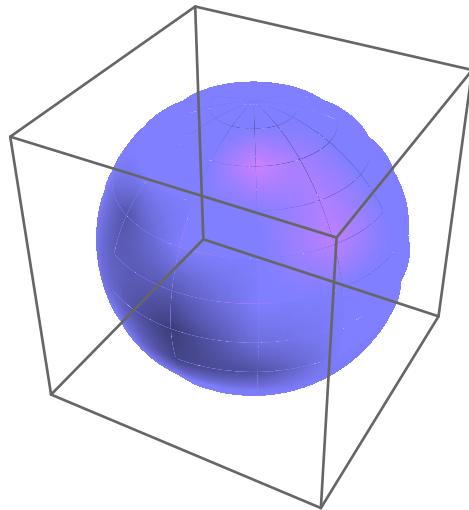
BesselK

Modified Bessel function of the second kind (SymPy, WMA)

```
BesselK[n, z]
    returns the modified Bessel function of the second kind K_n(z).
```

```
>> BesselK[1.5, 4]
0.014347
```

```
>> Plot[BesselK[0, x], {x, 0, 5}]
```



BesselY

Bessel function of the second kind (SymPy, WMA)

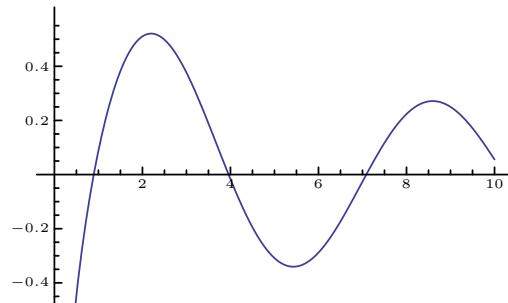
```
BesselY[n, z]
```

returns the Bessel function of the second kind $Y_n(z)$.

```
>> BesselY[1.5, 4]
```

0.367112

```
>> Plot[BesselY[0, x], {x, 0, 10}]
```



BesselYZero

WMA link

```
BesselYZero[n, k]
```

returns the k th zero of the Bessel function of the second kind $Y_n(z)$.

```
>> N[BesselYZero[0, 1]]
```

0.893577

```
>> N[BesselYZero[0, 1], 10]
```

0.8935769663

HankelH1

WMA link

```
HankelH1[n, z]
    returns the Hankel function of the first kind H_n^1(z).
```

```
>> HankelH1[1.5, 4]
0.185286 + 0.367112I
```

HankelH2

WMA link

```
HankelH2[n, z]
    returns the Hankel function of the second kind H_n^2(z).
```

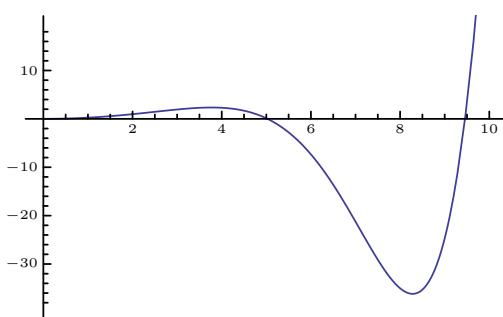
```
>> HankelH2[1.5, 4]
0.185286 - 0.367112I
```

KelvinBei

Kelvin function bei (mpmath, WMA)

```
KelvinBei[z]
    returns the Kelvin function bei(z).
KelvinBei[n, z]
    returns the Kelvin function bei_n(z).
```

```
>> KelvinBei[0.5]
0.0624932
>> KelvinBei[1.5 + I]
0.326323 + 0.755606I
>> KelvinBei[0.5, 0.25]
0.370153
>> Plot[KelvinBei[x], {x, 0, 10}]
```



KelvinBer

Kelvin function ber (mpmath, WMA)

```

KelvinBer[z]
    returns the Kelvin function  $\text{ber}(z)$ .
KelvinBer[n, z]
    returns the Kelvin function  $\text{ber}_n(z)$ .

```

```

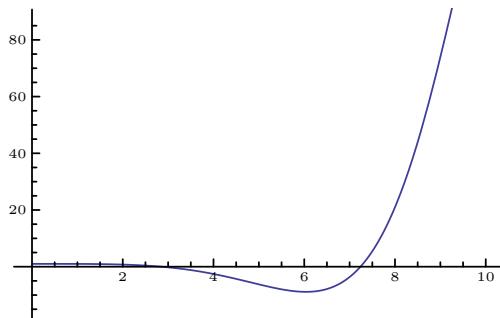
>> KelvinBer[0.5]
0.999023

>> KelvinBer[1.5 + I]
1.1162 - 0.117944I

>> KelvinBer[0.5, 0.25]
0.148824

>> Plot[KelvinBer[x], {x, 0, 10}]

```



KelvinKei

Kelvin function kei (mpmath, WMA)

```

KelvinKei[z]
    returns the Kelvin function  $\text{kei}(z)$ .
KelvinKei[n, z]
    returns the Kelvin function  $\text{kei}_n(z)$ .

```

```

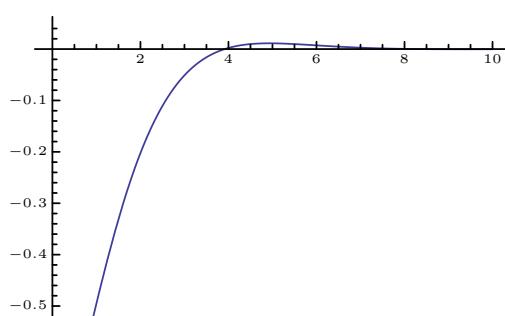
>> KelvinKei[0.5]
- 0.671582

>> KelvinKei[1.5 + I]
- 0.248994 + 0.303326I

>> KelvinKei[0.5, 0.25]
- 2.0517

>> Plot[KelvinKei[x], {x, 0, 10}]

```



KelvinKer

Kelvin function ker (mpmath, WMA)

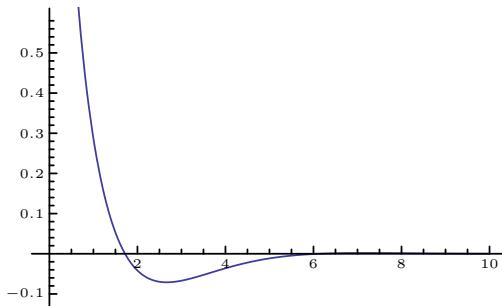
```
KelvinKer[z]
    returns the Kelvin function  $\text{ker}(z)$ .
KelvinKer[n, z]
    returns the Kelvin function  $\text{ker}_n(z)$ .
```

```
>> KelvinKer[0.5]
0.855906

>> KelvinKer[1.5 + I]
-0.167162 - 0.184404I

>> KelvinKer[0.5, 0.25]
0.450023

>> Plot[KelvinKer[x], {x, 0, 10}]
```



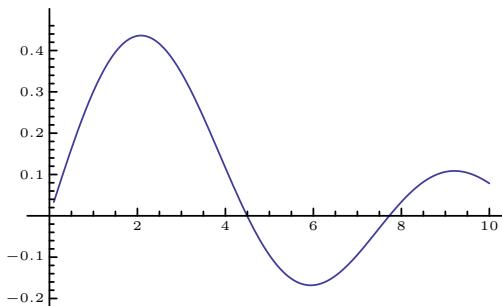
SphericalBesselJ

Spherical Bessel function of the first kind (Sympy, WMA)

```
SphericalBesselJ[n, z]
    returns the spherical Bessel function of the first kind  $Y_n(z)$ .
```

```
>> SphericalBesselJ[1, 5.2]
-0.122771

>> Plot[SphericalBesselJ[1, x], {x, 0.1, 10}]
```

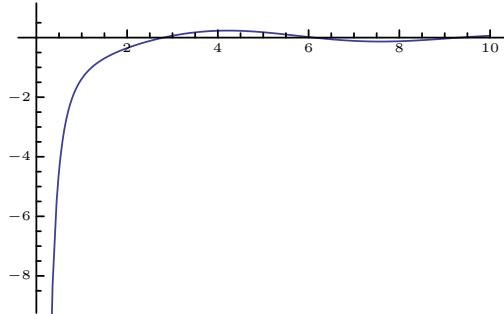


SphericalBesselY

Spherical Bessel function of the first kind (Sympy, WMA)

```
SphericalBesselY[n, z]
    returns the spherical Bessel function of the second kind  $Y_n(z)$ .
```

```
>> SphericalBesselY[1, 5.5]
0.104853
>> Plot[SphericalBesselY[1, x], {x, 0, 10}]
```



SphericalHankelH1

Spherical Bessel function of the first kind (WMA link)

```
SphericalHankelH1[n, z]
    returns the spherical Hankel function of the first kind  $h_n^{(1)}(z)$ .
```

```
>> SphericalHankelH1[3, 1.5]
0.0283246 - 3.78927I
```

SphericalHankelH2

Spherical Bessel function of the second kind (WMA link)

```
SphericalHankelH1[n, z]
    returns the spherical Hankel function of the second kind  $h_n^{(2)}(z)$ .
```

```
>> SphericalHankelH2[3, 1.5]
0.0283246 + 3.78927I
```

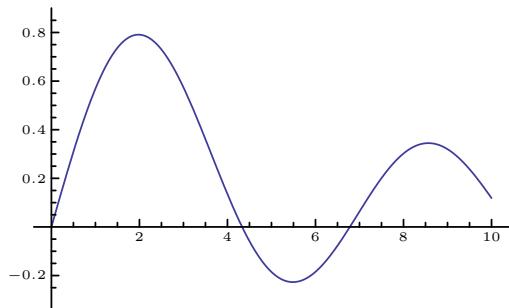
StruveH

Struve functions H (WMA)

```
StruveH[n, z]
    returns the Struve function  $H_n(z)$ .
```

```
>> StruveH[1.5, 3.5]
1.13192
```

```
>> Plot[StruveH[0, x], {x, 0, 10}]
```



StruveL

Modified Struve functions L

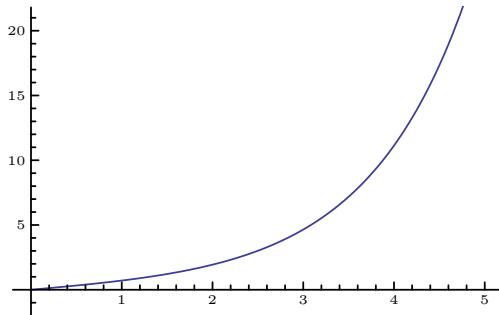
StruveL[n, z]

returns the modified Struve function $L_n(z)$.

```
>> StruveL[1.5, 3.5]
```

4.41126

```
>> Plot[StruveL[0, x], {x, 0, 5}]
```



WeberE

WMA link

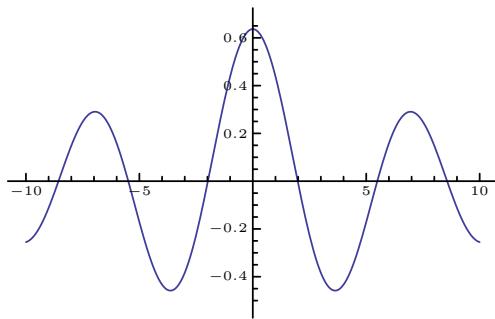
WeberE[n, z]

returns the Weber function $E_n(z)$.

```
>> WeberE[1.5, 3.5]
```

-0.397256

```
>> Plot[WeberE[1, x], {x, -10, 10}]
```



Elliptic Integrals

Elliptic Integrals

In integral calculus, an elliptic integral is one of a number of related functions defined as the value of certain integral. Their name originates from their originally arising in connection with the problem of finding the arc length of an ellipse.

These functions often are used in cryptography to encode and decode messages.

EllipticE

Elliptic complete elliptic integral of the second kind (SymPy, WMA)

EllipticE[m]

computes the complete elliptic integral $E(m)$.

EllipticE[phi|m]

computes the complete elliptic integral of the second kind $E(m \mid \phi)$.

Elliptic curves give $\pi / 2$ when evaluated at zero:

```
>> EllipticE[0]
```

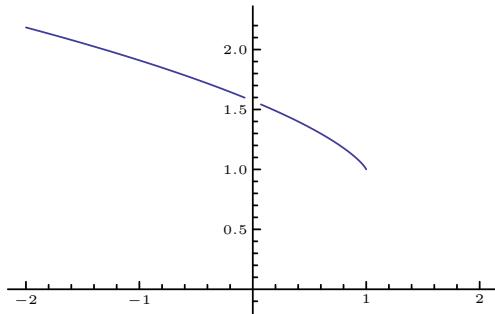
$$\frac{\pi}{2}$$

```
>> EllipticE[0.3, 0.8]
```

$$0.296426$$

Plot over a reals centered around 0:

```
>> Plot[EllipticE[m], {m, -2, 2}]
```



EllipticF

Complete elliptic integral of the first kind (SymPy, WMA)

```
EllipticF[phi, m]
computes the elliptic integral of the first kind  $F(\phi | m)$ .
```

```
>> EllipticF[0.3, 0.8]
0.303652
```

EllipticF is zero when the first argument is zero:

```
>> EllipticF[0, 0.8]
0
```

EllipticK

Complete elliptic integral of the first kind (SymPy, WMA)

```
EllipticK[m]
computes the elliptic integral of the first kind  $K(m)$ .
```

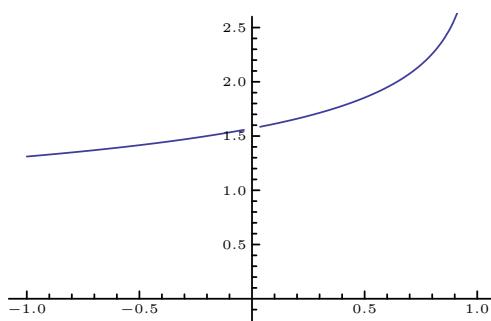
```
>> EllipticK[0.5]
1.85407
```

Elliptic curves give $\pi / 2$ when evaluated at zero:

```
>> EllipticK[0]
 $\frac{\pi}{2}$ 
```

Plot over a reals around 0:

```
>> Plot[EllipticK[n], {n, -1, 1}]
```



EllipticPi

Complete elliptic integral of the third kind (SymPy, WMA)

```
EllipticPi[n, m]
computes the elliptic integral of the third kind  $Pi(m)$ .
```

```
>> EllipticPi[0.4, 0.6]
2.89281
```

Elliptic curves give $\pi / 2$ when evaluated at zero:

```
>> EllipticPi[0, 0]

$$\frac{\pi}{2}$$

```

Error Function and Related Functions

Error Function and Related Function

Erf

Error function (SymPy, WMA)

```
Erf[z]
    returns the error function of z.
Erf[z0, z1]
    returns the result of Erf[z1] - Erf[z0].
```

$\text{Erf}[x]$ is an odd function:

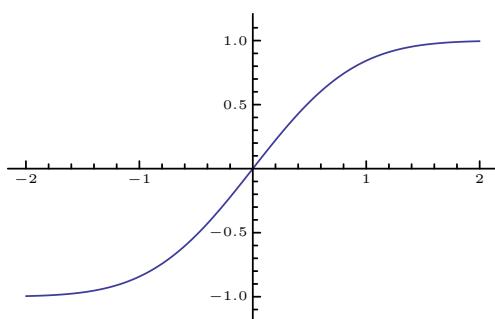
```
>> Erf[-x]
-Erf[x]

>> Erf[1.0]
0.842701

>> Erf[0]
0

>> {Erf[0, x], Erf[x, 0]}
{Erf[x], -Erf[x]}

>> Plot[Erf[x], {x, -2, 2}]
```



Erfc

Complementary Error function (SymPy, WMA)

```
Erfc[z]
    returns the complementary error function of z.
```

```
>> Erfc[-x] / 2

$$\frac{2 - \text{Erfc}[x]}{2}$$

```

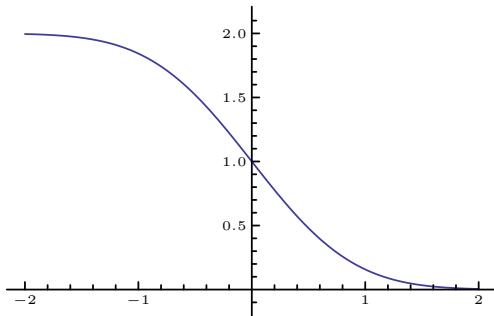
```

>> Erfc[1.0]
0.157299

>> Erfc[0]
1

>> Plot[Erfc[x], {x, -2, 2}]

```



FresnelC

Fresnel integral (mpmath, WMA)

FresnelC[z]
is the Fresnel C integral $C(z)$.

```

>> FresnelC[{0, Infinity}]
 $\left\{0, \frac{1}{2}\right\}$ 

>> Integrate[Cos[x^2 Pi/2], {x, 0, z}]
FresnelC[z]

```

FresnelS

Fresnel integral (mpmath, WMA)

FresnelS[z]
is the Fresnel S integral $S(z)$.

```

>> FresnelS[{0, Infinity}]
 $\left\{0, \frac{1}{2}\right\}$ 

>> Integrate[Sin[x^2 Pi/2], {x, 0, z}]
FresnelS[z]

```

InverseErf

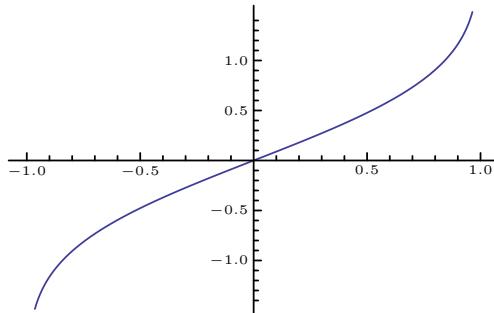
Inverse error function (SymPy, WMA)

InverseErf [z]

returns the inverse error function of z .

```
>> InverseErf /@ {-1, 0, 1}
{-\infty, 0, \infty}

>> Plot[InverseErf[x], {x, -1, 1}]
```



`InverseErf [z]` only returns numeric values for $-1 \leq z \leq 1$:

```
>> InverseErf /@ {0.9, 1.0, 1.1}
{1.16309, \infty, InverseErf[1.1]}
```

InverseErfc

Complementary error function (SymPy, WMA)

InverseErfc [z]

returns the inverse complementary error function of z .

```
>> InverseErfc /@ {0, 1, 2}
{\infty, 0, -\infty}
```

Exponential Integral and Special Functions

Exponential Integral and Special Function

ExplIntegralE

WMA link

ExpIntegralE[n, z]

returns the exponential integral function $E_n(z)$.

```
>> ExpIntegralE[2.0, 2.0]
0.0375343
```

ExplIntegralEi

WMA link

ExpIntegralEi[z]
returns the exponential integral function \$Ei(z)\$.

```
>> ExpIntegralEi[2.0]  
4.95423
```

ProductLog

WMA link

ProductLog[z]
returns the value of the Lambert W function at z .

The defining equation:

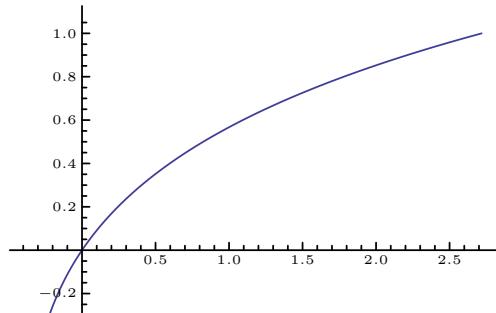
```
>> z == ProductLog[z] * E ^ ProductLog[z]  
True
```

Some special values:

```
>> ProductLog[0]  
0  
>> ProductLog[E]  
1
```

The graph of ProductLog:

```
>> Plot[ProductLog[x], {x, -1/E, E}]
```



Gamma and Related Functions

Gamma and Related Function

Beta

Euler beta function (SymPy, WMA)

Beta[a, b]
is the Euler's Beta function.
Beta[z, a, b]
gives the incomplete Beta function.

The

Beta function satisfies the property $\text{Beta}[x, y] = \text{Integrate}[t^{x-1}(1-t)^{y-1}, \{t, 0, 1\}] = \text{Gamma}[a]\text{Gamma}[b]/\text{Gamma}[a+b]$

```

>> Beta[2, 3]

$$\frac{1}{12}$$

>> 12* Beta[1., 2, 3]
1.

```

Factorial (!)

Factorial (SymPy, mpmath, WMA)

```

Factorial[n]
n!
computes the factorial of n.

```

```

>> 20!
2432902008176640000

```

Factorial handles numeric (real and complex) values using the gamma function:

```

>> 10.5!
1.18994*^7
>> (-3.0+1.5*I)!
0.0427943 - 0.00461565I

```

However, the value at poles is ComplexInfinity:

```

>> (-1.)!
ComplexInfinity

```

Factorial has the same operator (!) as Not, but with higher precedence:

```

>> !a! //FullForm
Not[Factorial[a]]

```

Factorial2 (!!)

WMA link

```

Factorial2[n]
n!!
computes the double factorial of n.

```

The

double factorial or semifactorial of a number n , is the product of all the integers from 1 up to n that have the same parity (odd or even) as n .

```

>> 5!!
15.
>> Factorial2[-3]
-1.

```

Factorial2 accepts Integers, Rationals, Reals, or Complex Numbers:

```

>> I!! + 1
3.71713 + 0.279527I

```

Irrationals can be handled by using numeric approximation:

```
>> N[Pi!! , 6]
3.35237
```

Gamma

Gamma function (SymPy, mpmath, WMA)

The gamma function is one commonly used extension of the factorial function applied to complex numbers, and is defined for all complex numbers except the non-positive integers.

Gamma[z] is the gamma function on the complex number z . Gamma[z, x] is the upper incomplete gamma function. Gamma[z, x0, x1] is equivalent to $\text{Gamma}[z, x0] - \text{Gamma}[z, x1]$.
--

$\text{Gamma}[z]$ is equivalent to $(z - 1)!$:

```
>> Simplify[Gamma[z] - (z - 1)!]
0
```

Exact arguments:

```
>> Gamma[8]
5040

>> Gamma[1/2]
Sqrt[Pi]

>> Gamma[1, x]
E^-x

>> Gamma[0, x]
ExpIntegralE[1, x]
```

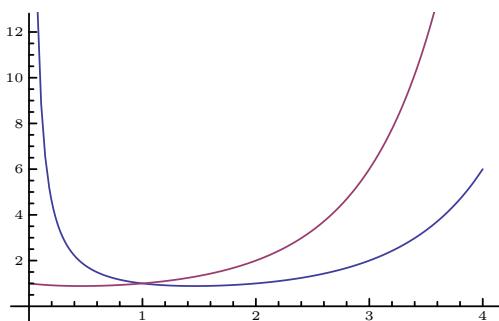
Numeric arguments:

```
>> Gamma[123.78]
4.21078^204

>> Gamma[1. + I]
0.498016 - 0.15495I
```

Both Gamma and Factorial functions are continuous:

```
>> Plot[{Gamma[x], x!}, {x, 0, 4}]
```



LogGamma

log-gamma function (SymPy, WMA)

```
LogGamma[z]
    is the logarithm of the gamma function on the complex number z.
```

```
>> LogGamma[3]
    Log[2]
```

LogGamma[z] has different analytical structure than Log[Gamma[z]]

```
>> LogGamma[-2.+3 I]
    - 6.77652 - 4.56879I

>> Log[Gamma[-2.+3 I]]
    - 6.77652 + 1.71439I
```

LogGamma also can be evaluated for large arguments, for which Gamma produces Overflow:

```
>> LogGamma[1.*^20]
    4.50517*^21

>> Log[Gamma[1.*^20]]
    Overflow occurred in computation.

Overflow[]
```

Pochhammer

Rising factorial (SymPy, WMA)

The Pochhammer symbol or rising factorial often appears in series expansions for hypergeometric functions.

The Pochhammer symbol has a definite value even when the gamma functions which appear in its definition are infinite.

```
Pochhammer[a, n]
    is the Pochhammer symbol  $a_n$ .
```

Product of the first 3 numbers:

```
>> Pochhammer[1, 3]
    6
```

Pochhammer[1, n] is the same as Pochhammer[2, n-1] since 1 is a multiplicative identity.

```
>> Pochhammer[1, 3] == Pochhammer[2, 2]
    True
```

Although sometimes Pochhammer[0, n] is taken to be 1, in Mathics it is 0:

```
>> Pochhammer[0, n]
    0
```

Pochhammer uses Gamma for non-Integer values of n:

```
>> Pochhammer[1, 3.001]
    6.00754

>> Pochhammer[1, 3.001] == Pochhammer[2, 2.001]
    True
```

```
>> Pochhammer[1.001, 3] == 1.001 2.001 3.001
True
```

PolyGamma

Polygamma function (SymPy, WMA)

PolyGamma is a meromorphic function on the complex numbers and is defined as a derivative of the logarithm of the gamma function.

PolyGamma[z]
returns the digamma function.
PolyGamma[n,z]
gives the n^{th} derivative of the digamma function.

```
>> PolyGamma[5]

$$\frac{25}{12} - \text{EulerGamma}$$

>> PolyGamma[3, 5]

$$-\frac{22369}{3456} + \frac{\pi^4}{15}$$

```

StieltjesGamma

Stieltjes constants (SymPy, WMA)

StieltjesGamma[n]
returns the Stieltjes constant for n .
StieltjesGamma[n, a]
gives the generalized Stieltjes constant of its parameters

Orthogonal Polynomials

Orthogonal Polynomial

ChebyshevT

Chebyshev polynomial of the first kind (Sympy, WMA)

ChebyshevT[n, x]
returns the Chebyshev polynomial of the first kind $T_n(x)$.

```
>> ChebyshevT[8, x]

$$1 - 32x^2 + 160x^4 - 256x^6 + 128x^8$$

>> ChebyshevT[1 - I, 0.5]
0.800143 + 1.08198I
```

ChebyshevU

Chebyshev polynomial of the second kind (SymPy, WMA)

```
ChebyshevU[n, x]
    returns the Chebyshev polynomial of the second kind U_n(x).
```

```
>> ChebyshevU[8, x]
1 - 40x^2 + 240x^4 - 448x^6 + 256x^8
>> ChebyshevU[1 - I, 0.5]
1.60029 + 0.721322I
```

GegenbauerC

Gegenbauer polynomials (SymPy, WMA)

```
GegenbauerC[n, m, x]
    returns the Gegenbauer polynomial C_n^(m)(x).
```

```
>> GegenbauerC[6, 1, x]
-1 + 24x^2 - 80x^4 + 64x^6
>> GegenbauerC[4 - I, 1 + 2 I, 0.7]
-3.2621 - 24.9739I
```

HermiteH

Hermite polynomial (SymPy, WMA)

```
HermiteH[n, x]
    returns the Hermite polynomial H_n(x).
```

```
>> HermiteH[8, x]
1680 - 13440x^2 + 13440x^4 - 3584x^6 + 256x^8
>> HermiteH[3, 1 + I]
-28 + 4I
>> HermiteH[4.2, 2]
77.5291
```

JacobiP

Jacobi polynomials (SymPy, WMA)

```
JacobiP[n, a, b, x]
    returns the Jacobi polynomial P_n^(a,b)(x).
```

```

>> JacobiP[1, a, b, z]

$$\frac{a}{2} - \frac{b}{2} + z \left( 1 + \frac{a}{2} + \frac{b}{2} \right)$$

>> JacobiP[3.5 + I, 3, 2, 4 - I]
1410.02 + 5797.3I

```

LaguerreL

Laguerre polynomials (SymPy, WMA)

```

LaguerreL[n, x]
    returns the Laguerre polynomial L_n(x).
LaguerreL[n, a, x]
    returns the generalised Laguerre polynomial L^a_n(x).

```

```

>> LaguerreL[8, x]

$$1 - 8x + 14x^2 - \frac{28x^3}{3} + \frac{35x^4}{12} - \frac{7x^5}{15} + \frac{7x^6}{180} - \frac{x^7}{630} + \frac{x^8}{40320}$$

>> LaguerreL[3/2, 1.7]
- 0.947134
>> LaguerreL[5, 2, x]

$$21 - 35x + \frac{35x^2}{2} - \frac{7x^3}{2} + \frac{7x^4}{24} - \frac{x^5}{120}$$


```

LegendreP

Legendre polynomials (SymPy, WMA)

```

LegendreP[n, x]
    returns the Legendre polynomial P_n(x).
LegendreP[n, m, x]
    returns the associated Legendre polynomial P^m_n(x).

```

```

>> LegendreP[4, x]

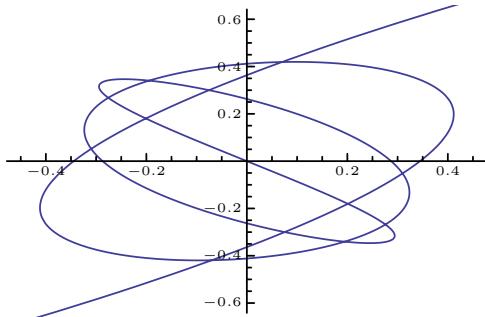
$$\frac{3}{8} - \frac{15x^2}{4} + \frac{35x^4}{8}$$

>> LegendreP[5/2, 1.5]
4.17762
>> LegendreP[1.75, 1.4, 0.53]
- 1.32619
>> LegendreP[1.6, 3.1, 1.5]
- 0.303998 - 1.91937I

```

LegendreP can be used to draw generalized Lissajous figures:

```
>> ParametricPlot[ {LegendreP[7, x], LegendreP[5, x]}, {x, -1, 1}]
```



LegendreQ

Legendre functions of the second kind (mpmath, WMA)

```
LegendreQ[n, x]
```

returns the Legendre function of the second kind $Q_n(x)$.

```
LegendreQ[n, m, x]
```

returns the associated Legendre function of the second $Q^m_n(x)$.

```
>> LegendreQ[5/2, 1.5]
```

$0.036211 - 6.56219I$

```
>> LegendreQ[1.75, 1.4, 0.53]
```

2.05499

```
>> LegendreQ[1.6, 3.1, 1.5]
```

$-1.71931 - 7.70273I$

SphericalHarmonicY

Spherical Harmonic [https](https://reference.wolfram.com/language/ref/SphericalHarmonicY.html) (mpmath, WMA)

```
SphericalHarmonicY[l, m, theta, phi]
```

returns the spherical harmonic function $Y_l^m(\theta, \phi)$.

```
>> SphericalHarmonicY[3/4, 0.5, Pi/5, Pi/3]
```

$0.254247 + 0.14679I$

```
>> SphericalHarmonicY[3, 1, theta, phi]
```

$$\frac{\sqrt{21} \left(1 - 5 \cos[\theta]^2\right) E^{I \phi} \sin[\theta]}{8 \sqrt{\pi}}$$

Zeta Functions and Polylogarithms

Zeta Functions and Polylogarithm

LerchPhi

WMA link

```
LerchPhi[z,s,a]  
gives the Lerch transcendent  
Phi(z,s,a).
```

```
>> LerchPhi[2, 3, -1.5]  
19.3893 - 2.1346I  
>> LerchPhi[1, 2, 1/4]  
17.1973
```

Zeta

WMA link

```
Zeta[z]  
returns the Riemann zeta function of z.
```

```
>> Zeta[2]  

$$\frac{\pi^2}{6}$$
  
>> Zeta[-2.5 + I]  
0.0235936 + 0.0014078I
```

49. Strings and Characters

Contents

Character Codes	547	StringInsert	552	EndOfLine	558
FromCharacterCode	548	StringJoin (<>) . . .	552	EndOfString	558
ToCharacterCode .	548	StringLength	553	LetterCharacter	558
Characters in Strings	549	StringPosition	553	StartOfLine	559
CharacterRange .	549	StringReplace	554	StartOfString	559
Characters	549	StringReverse	554	StringCases	560
DigitQ	549	StringRiffle	555	StringExpression	
LetterQ	550	StringSplit	555	(~)	560
LowerCaseQ	550	StringTake	556	StringFreeQ	561
ToLowerCase	550	StringTrim	556	StringMatchQ	561
ToUpperCase	550	Regular Expressions	557	WhitespaceChar-	
UpperCaseQ	551	RegularExpression	557	acter	561
Operations on Strings	551	String Patterns	557	WordBoundary	562
StringDrop	551	DigitCharacter . . .	557	WordCharacter	562

Character Codes

Character Code

FromCharacterCode

WMA link

```
FromCharacterCode[n]
    returns the character corresponding to Unicode codepoint n.
FromCharacterCode[{n1, n2, ...}]
    returns a string with characters corresponding to n_i.
FromCharacterCode[{{n11, n12, ...}, {n21, n22, ...}, ...}]
    returns a list of strings.
```

```
>> FromCharacterCode[100]
d

>> FromCharacterCode[228, "ISO8859-1"]
ä

>> FromCharacterCode[{100, 101, 102}]
def

>> ToCharacterCode[%]
{100,101,102}
```

```

>> FromCharacterCode[{{97, 98, 99}, {100, 101, 102}}]
{abc,def}

>> ToCharacterCode["abc 123"] // FromCharacterCode
abc 123

```

ToCharacterCode

WMA link

```

ToCharacterCode["string"]
converts the string to a list of character codes (Unicode codepoints).
ToCharacterCode[{ "string1", "string2", ...}]
converts a list of strings to character codes.

```

```

>> ToCharacterCode["abc"]
{97,98,99}

>> FromCharacterCode[%]
abc

>> ToCharacterCode["\[Alpha]\\[Beta]\\[Gamma]"]
{945,946,947}

>> ToCharacterCode["ä", "UTF8"]
{195,164}

>> ToCharacterCode["ä", "ISO8859-1"]
{228}

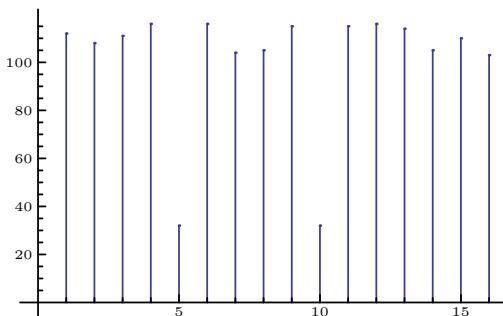
>> ToCharacterCode[{"ab", "c"}]
{{97,98},{99}}

>> ToCharacterCode[{"ab", x}]
String or list of strings expected at position 1 in
ToCharacterCode[{ab, x}].

ToCharacterCode [{ab,x}]

>> ListPlot[ToCharacterCode["plot this string"], Filling -> Axis]

```



Characters in Strings

Characters in String

CharacterRange

WMA link

```
CharacterRange["a", "b"]  
    returns a list of the Unicode characters from a to b inclusive.
```

```
>> CharacterRange["a", "e"]  
{a,b,c,d,e}  
  
>> CharacterRange["b", "a"]  
{}
```

Characters

WMA link

```
Characters["string"]  
    returns a list of the characters in string.
```

```
>> Characters["abc"]  
{a,b,c}
```

DigitQ

WMA link

```
DigitQ[string]  
    yields True if all the characters in the string are digits, and yields False otherwise.
```

```
>> DigitQ["9"]  
True  
  
>> DigitQ["a"]  
False  
  
>> DigitQ["0100110101100001011101000110100011010010110001101110011"]  
True  
  
>> DigitQ["-123456789"]  
False
```

LetterQ

WMA link

```
LetterQ[string]  
    yields True if all the characters in the string are letters, and yields False otherwise.
```

```
>> LetterQ["m"]
True

>> LetterQ["9"]
False

>> LetterQ["Mathics"]
True

>> LetterQ["Welcome to Mathics"]
False
```

LowerCaseQ

WMA link

```
LowerCaseQ[s]
    returns True if s consists wholly of lower case characters.
```

```
>> LowerCaseQ["abc"]
True
```

An empty string returns True.

```
>> LowerCaseQ[]
True
```

ToLowerCase

WMA link

```
ToLowerCase[s]
    returns s in all lower case.
```

```
>> ToLowerCase["New York"]
new york
```

ToUpperCase

WMA link

```
ToUpperCase[s]
    returns s in all upper case.
```

```
>> ToUpperCase["New York"]
NEW YORK
```

UpperCaseQ

WMA link

```
UpperCaseQ[s]
    returns True if s consists wholly of upper case characters.
```

```
>>UpperCaseQ["ABC"]
True
An empty string returns True.
>>UpperCaseQ[""]
True
```

Operations on Strings

Operations on String

StringDrop

WMA link

```
StringDrop["string", n]
gives string with the first n characters dropped.
StringDrop["string", -n]
gives string with the last n characters dropped.
StringDrop["string", {n}]
gives string with the nth character dropped.
StringDrop["string", {m, n}]
gives string with the characters m through n dropped.
```

```
>> StringDrop["abcde", 2]
cde
>> StringDrop["abcde", -2]
abc
>> StringDrop["abcde", {2}]
acde
>> StringDrop["abcde", {2,3}]
ade
>> StringDrop["abcd", {3,2}]
abcd
>> StringDrop["abcd", 0]
abcd
```

StringInsert

WMA link

```

StringInsert["string`", "snew", n]
    yields a string with snew inserted starting at position n in string.
StringInsert["string`", "snew", -n]
    inserts a at position n from the end of "string".
StringInsert["string`", "snew", {n_1, n_2, ...}]
    inserts a copy of snew at each position ni in string; the ni are taken before any insertion is done.
StringInsert[{s_1, s_2, ...}, "snew", n]
    gives the list of results for each of the si.

```

```

>> StringInsert["noting", "h", 4]
nothing

>> StringInsert["note", "d", -1]
noted

>> StringInsert["here", "t", -5]
there

>> StringInsert["adac", "he", {1, 5}]
headache

>> StringInsert[{"something", "sometimes"}, " ", 5]
{some thing, some times}

>> StringInsert["1234567890123456", ".", Range[-16, -4, 3]]
1.234.567.890.123.456

```

StringJoin (<>)

WMA link

```

StringJoin["s1`", "s2", ...]
    returns the concatenation of the strings s1, s2, ...

```

```

>> StringJoin["a", "b", "c"]
abc

>> "a" <> "b" <> "c" // InputForm
"abc"

StringJoin flattens lists out:
>> StringJoin[{"a", "b"}] // InputForm
"ab"

>> Print[StringJoin[{"Hello", " ", {"world"}}, "!"]]
Hello world!

```

StringLength

WMA link

```
StringLength["string"]
gives the length of string.
```

```
>> StringLength["abc"]
3

StringLength is listable:
>> StringLength[{"a", "bc"}]
{1,2}

>> StringLength[x]
String expected.
StringLength[x]
```

StringPosition

WMA link

```
StringPosition["string", patt]
gives a list of starting and ending positions where patt matches "string".
StringPosition["string", patt, n]
returns the first n matches only.
StringPosition["string", {patt1, patt2, ...}, n]
matches multiple patterns.
StringPosition[{s1, s2, ...}, patt]
returns a list of matches for multiple strings.
```

```
>> StringPosition["123ABCxyABCzzzABCABC", "ABC"]
{{4,6},{9,11},{15,17},{18,20}}

>> StringPosition["123ABCxyABCzzzABCABC", "ABC", 2]
{{4,6},{9,11}}
```

StringPosition can be useful for searching through text.

```
>> data = Import["ExampleData/EinsteinSzilLetter.txt", CharacterEncoding ->"UTF8"];
```

```
>> StringPosition[data, "uranium"]
{{299,305},{870,876},{1538,1544},{1671,1677},{2300,2306},{2784,2790},{3093,3099}}
```

StringReplace

WMA link

```
StringReplace["string", "a" -> "b"]
replaces each occurrence of old with new in string.
StringReplace["string", {"s1" -> "sp1", "s2" -> "sp2"}]
performs multiple replacements of each si by the corresponding spi in string.
StringReplace["string", srules, n]
only performs the first n replacements.
StringReplace[{"string1", "string2", ...}, srules]
performs the replacements specified by srules on a list of strings.
```

`StringReplace` replaces all occurrences of one substring with another:

```
>> StringReplace["xyxyxyyxxxxxy", "xy" -> "A"]
AAAyxxAyA
```

Multiple replacements can be supplied:

```
>> StringReplace["xyzwxyzwxxyzxyzw", {"xyz" -> "A", "w" -> "BCD"}]
ABCDABCDxAABCD
```

Only replace the first 2 occurrences:

```
>> StringReplace["xyxyxyyxxxxxy", "xy" -> "A", 2]
AAxyyxxxxxy
```

Also works for multiple rules:

```
>> StringReplace["abba", {"a" -> "A", "b" -> "B"}, 2]
ABba
```

`StringReplace` acts on lists of strings too:

```
>> StringReplace[{"xyxyxxy", "yxyxyxxxxxy"}, "xy" -> "A"]
{AAxA, yAAxxAyA}
```

`StringReplace` also can be used as an operator:

```
>> StringReplace["y" -> "ies"] ["city"]
cities
```

StringReverse

WMA link

```
StringReverse["string"]
reverses the order of the characters in "string".
```

```
>> StringReverse["live"]
evil
```

StringRiffle

WMA link

```
StringRiffle[{s1, s2, s3, ...}]
returns a new string by concatenating all the si, with spaces inserted between them.
StringRiffle[list, sep]
inserts the separator sep between all elements in list.
StringRiffle[list, {"left", "sep", "right"}]
use left and right as delimiters after concatenation.
```

```
>> StringRiffle[{"a", "b", "c", "d", "e"}]
a b c d e
>> StringRiffle[{"a", "b", "c", "d", "e"}, ", "]
a, b, c, d, e
```

```
>> StringRiffle[{"a", "b", "c", "d", "e"}, {"(", " ", ")"}]
(a b c d e)
```

StringSplit

WMA link

```
StringSplit[s]
splits the string s at whitespace, discarding the whitespace and returning a list of strings.
StringSplit[s, pattern]
splits s into substrings separated by delimiters matching the string expression pattern.
StringSplit[s, {p1, p2, ...}]
splits s at any of the pi patterns.
StringSplit[{s1, s2, ...}, {d1, d2, ...}]
returns a list with the result of applying the function to each element.
```

```
>> StringSplit["abc,123", ","]
{abc,123}
```

By default any number of whitespace characters are used to at a delimiter:

```
>> StringSplit[" abc 123 "]
{abc,123}
```

However if you want instead to use only a *single* character for each delimiter, use `WhiteSpaceCharacter`:

```
>> StringSplit[" abc 123 ", WhiteSpaceCharacter]
{,,abc,,,123,,}

>> StringSplit["abc,123.456", {",", ".", ".}]
{abc,123,456}

>> StringSplit["a b c", RegularExpression["+ +"]]
{a,b,c}

>> StringSplit[{"a b", "c d"}, RegularExpression["+ +"]]
{{a,b},{c,d}}

>> StringSplit["x", "x"]
{}
```

Split using a delimiter that has nonzero list of 12's

```
>> StringSplit["12312123", "12" ..]
{3,3}
```

StringTake

WMA link

```

StringTake["string", n]
    gives the first n characters in string.
StringTake["string", -n]
    gives the last n characters in string.
StringTake["string", {n}]
    gives the nth character in string.
StringTake["string", {m, n}]
    gives characters m through n in string.
StringTake["string", {m, n, s}]
    gives characters m through n in steps of s.
StringTake[{s1, s2, ...} spec]
    gives the list of results for each of the si.

```

```

>> StringTake["abcde", 2]
ab
>> StringTake["abcde", 0]

>> StringTake["abcde", -2]
de
>> StringTake["abcde", {2}]
b
>> StringTake["abcd", {2,3}]
bc
>> StringTake["abcdefg", {1, 5, 2}]
ace

```

Take the last 2 characters from several strings:

```

>> StringTake[{"abcdef", "stuv", "xyzw"}, -2]
{ef, uv, zw}

```

StringTake also supports standard sequence specifications

```

>> StringTake["abcdef", All]
abcdef

```

StringTrim

WMA link

```

StringTrim[s]
    returns a version of s with whitespace removed from start and end.

```

```

>> StringJoin["a", StringTrim[" \tb\n "], "c"]
abc
>> StringTrim["ababaxababyaabab", RegularExpression["(ab)+"]]
axababya

```

Regular Expressions

Regular Expression

RegularExpression

WMA link

RegularExpression[“regex”]
represents the regex specified by the string \$“regex”\$.

```
>> StringSplit["1.23, 4.56 7.89", RegularExpression["(\s|,)+"]]
{1.23,4.56,7.89}
```

String Patterns

String Pattern

DigitCharacter

WMA link

DigitCharacter
represents the digits 0-9.

```
>> StringMatchQ["1", DigitCharacter]
True
>> StringMatchQ["a", DigitCharacter]
False
>> StringMatchQ["12", DigitCharacter]
False
>> StringMatchQ["123245", DigitCharacter..]
True
```

EndOfLine

WMA link

EndOfLine
represents the end of a line in a string.

```
>> StringReplace["aba\nbba\na\nab", "a" ~~EndOfLine -> "c"]
abc
bbc
c
ab
```

```
>> StringSplit["abc\ndef\nhij", EndOfLine]
{abc,
 def,
 hij}
```

EndOfString

WMA link

EndOfString
represents the end of a string.

Test whether strings end with "e":

```
>> StringMatchQ[#, __ ~~ "e" ~~ EndOfString] &/@ {"apple", "banana", "artichoke"}
{True, False, True}

>> StringReplace["aab\nabb", "b" ~~ EndOfString -> "c"]
aab
abc
```

LetterCharacter

WMA link

LetterCharacter
represents letters.

```
>> StringMatchQ[#, LetterCharacter] & /@ {"a", "1", "A", " ", ".}
{True, False, True, False, False}
```

LetterCharacter also matches unicode characters.

```
>> StringMatchQ["\[Lambda]", LetterCharacter]
True
```

StartOfLine

WMA link

StartOfLine
represents the start of a line in a string.

```
>> StringReplace["aba\nbba\nna\nab", StartOfLine ~~ "a" -> "c"]
cba
bba
c
cb
```

```
>> StringSplit["abc\ndef\nhij", StartOfLine]
{abc
 ,def
 ,hij}
```

StartOfString

WMA link

StartOfString
represents the start of a string.

Test whether strings start with "a":

```
>> StringMatchQ[#, StartOfString ~~ "a" ~~ __] &/@ {"apple", "banana", "artichoke"}
{True, False, True}

>> StringReplace["aba\nabb", StartOfString ~~ "a" -> "c"]
cba
abb
```

StringCases

WMA link

StringCases ["*string*", *pattern*]
gives all occurrences of *pattern* in *string*.
StringReplace ["*string*", *pattern* -> *form*]
gives all instances of *form* that stem from occurrences of *pattern* in *string*.
StringCases ["*string*", {*pattern*₁, *pattern*₂, ...}]
gives all occurrences of *pattern*₁, *pattern*₂,
StringReplace ["*string*", *pattern*, *n*]
gives only the first *n* occurrences.
StringReplace [{*string*₁ ``, *string*₂, ...}, *pattern*]
gives occurrences in *string*₁, *string*₂, ...

```
>> StringCases["axbaxxb", "a" ~~ x_ ~~ "b"]
{axb}

>> StringCases["axbaxxb", "a" ~~ x__ ~~ "b"]
{axbaxxb}

>> StringCases["axbaxxb", Shortest["a" ~~ x__ ~~ "b"]]
{axb, axxb}

>> StringCases["-abc- def -uvw- xyz", Shortest["-" ~~ x__ ~~ "-"] -> x]
{abc, uvw}

>> StringCases["-öhi- -abc- -.-", "-" ~~ x : WordCharacter .. ~~ "-" -> x]
{öhi, abc}
```

```

>> StringCases["abc-abc xyz-uvw", Shortest[x : WordCharacter .. ~~"-" ~~
x_] -> x]
{abc}

>> StringCases["abba", {"a" -> 10, "b" -> 20}, 2]
{10,20}

>> StringCases["a#ä_123", WordCharacter]
{a,ä,1,2,3}

>> StringCases["a#ä_123", LetterCharacter]
{a,ä}

```

StringExpression (~~)

WMA link

StringExpression[*s*_1, *s*_2, ...]
represents a sequence of strings and symbolic string objects *s*_*i*.

```

>> "a" ~~ "b" // FullForm
"ab"

```

StringFreeQ

WMA link

StringFreeQ["*string*", *patt*]
returns True if no substring in *string* matches the string expression *patt*, and returns False otherwise.
StringFreeQ[{"*s*1", "*s*2", ...}, *patt*]
returns the list of results for each element of string list.
StringFreeQ[{"*string*", {*p*1, *p*2, ...}]
returns True if no substring matches any of the *p**i*.
StringFreeQ[*patt*]
represents an operator form of StringFreeQ that can be applied to an expression.

```

>> StringFreeQ["mathics", "m" ~~__ ~~"s"]
False

>> StringFreeQ["mathics", "a" ~~__ ~~"m"]
True

>> StringFreeQ["Mathics", "MA" , IgnoreCase -> True]
False

>> StringFreeQ[{"g", "a", "laxy", "universe", "sun"}, "u"]
{True, True, True, False, False}

>> StringFreeQ["e" ~~__ ~~"u"] /@ {"The Sun", "Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune"}
{False, False, False, True, True, True, True, False}

```

```
>> StringFreeQ[{"A", "Galaxy", "Far", "Far", "Away"}, {"F" ~~__ ~~"r", "aw" ~~__}, IgnoreCase -> True]
{True, True, False, False, False}
```

StringMatchQ

WMA link

`StringMatchQ['string', pattern]`
checks if "string" matches *pattern*

```
>> StringMatchQ["abc", "abc"]
True

>> StringMatchQ["abc", "abd"]
False

>> StringMatchQ["15a94xcZ6", (DigitCharacter | LetterCharacter)..]
True
```

Use `StringMatchQ` as an operator

```
>> StringMatchQ[LetterCharacter] ["a"]
True
```

WhitespaceCharacter

WMA link

`WhitespaceCharacter`
represents a single whitespace character.

```
>> StringMatchQ["\n", WhitespaceCharacter]
True

>> StringSplit["a\nb\r\nc\rd", WhitespaceCharacter]
{a,b,,c,d}
```

For sequences of whitespace characters use `Whitespace`:

```
>> StringMatchQ[" \n", WhitespaceCharacter]
False

>> StringMatchQ[" \n", Whitespace]
True
```

WordBoundary

WMA link

`WordBoundary`
represents the boundary between words.

```
>> StringReplace["apple banana orange artichoke", "e" ~~WordBoundary ->  
"E"]  
applE banana orangE artichokE
```

WordCharacter

WMA link

WordCharacter

represents a single letter or digit character.

```
>> StringMatchQ[#, WordCharacter] &/@ {"1", "a", "A", ",", " " }  
{True, True, True, False, False}
```

Test whether a string is alphanumeric:

```
>> StringMatchQ["abc123DEF", WordCharacter..]  
True  
>> StringMatchQ["$b;123", WordCharacter..]  
False
```

50. Tensors

A tensor is an algebraic object that describes a (multilinear) relationship between sets of algebraic objects related to a vector space. Objects that tensors may map between include vectors and scalars, and even other tensors.

There are many types of tensors, including scalars and vectors (which are the simplest tensors), dual vectors, multilinear maps between vector spaces, and even some operations such as the dot product. Tensors are defined independent of any basis, although they are often referred to by their components in a basis related to a particular coordinate system.

Mathics3 represents tensors of vectors and matrices as lists; tensors of any rank can be handled.

Contents

ArrayDepth	563	Outer	565	TransformationFunction	566
Dimensions	564	RotationTransform . .	566	TranslationTransform .	567
Dot (.)	564	ScalingTransform . . .	566	Transpose	567
Inner	565	ShearingTransform . .	566		

ArrayDepth

WMA link

ArrayDepth [*a*]

returns the depth of the non-ragged array *a*, defined as `Length[Dimensions[a]]`.

```
>> ArrayDepth[{{a,b},{c,d}}]  
2  
>> ArrayDepth[x]  
0
```

Dimensions

WMA

Dimensions [*expr*]

returns a list of the dimensions of the expression *expr*.

A vector of length 3:

```
>> Dimensions[{a, b, c}]  
{3}
```

A 3x2 matrix:

```
>> Dimensions[{{a, b}, {c, d}, {e, f}}]  
{3,2}
```

Ragged arrays are not taken into account:

```
>> Dimensions[{{a, b}, {b, c}, {c, d, e}}]  
{3}
```

The expression can have any head:

```
>> Dimensions[f[f[a, b, c]]]  
{1,3}
```

Dot (.)

Dot product (WMA link)

```
Dot[x, y]  
x . y  
computes the vector dot product or matrix product  $x \cdot y$ .
```

Scalar product of vectors:

```
>> {a, b, c} . {x, y, z}  
ax + by + cz
```

Product of matrices and vectors:

```
>> {{a, b}, {c, d}} . {x, y}  
{ax + by, cx + dy}
```

Matrix product:

```
>> {{a, b}, {c, d}} . {{r, s}, {t, u}}  
{ar + bt, as + bu}, {cr + dt, cs + du}  
  
>> a . b  
a.b
```

Inner

WMA link

```
Inner[f, x, y, g]  
computes a generalised inner product of  $x$  and  $y$ , using a multiplication function  $f$  and an addition function  $g$ .
```

```
>> Inner[f, {a, b}, {x, y}, g]  
g [f [a, x], f [b, y]]
```

Inner can be used to compute a dot product:

```
>> Inner[Times, {a, b}, {c, d}, Plus] == {a, b} . {c, d}  
True
```

The inner product of two boolean matrices:

```
>> Inner[And, {{False, False}, {False, True}}, {{True, False}, {True, True}}, Or]  
{ {False, False}, {True, True}}
```

Inner works with tensors of any depth:

```
>> Inner[f, {{a, b}}, {{c, d}}], {{1}, {2}}, g
{{g[f[a, 1], f[b, 2]]}, {g[f[c, 1], f[d, 2]]}}
```

Outer

Outer product (WMA link)

```
Outer[f, x, y]
```

computes a generalised outer product of x and y , using the function f in place of multiplication.

```
>> Outer[f, {a, b}, {1, 2, 3}]
{{f[a, 1], f[a, 2], f[a, 3]}, {f[b, 1], f[b, 2], f[b, 3]}}
```

Outer product of two matrices:

```
>> Outer[Times, {a, b}, {c, d}, {{1, 2}, {3, 4}}]
{{{a, 2a}, {3a, 4a}}, {{b, 2b}, {3b, 4b}}, {{c, 2c}, {3c, 4c}}, {{d, 2d}, {3d, 4d}}}
```

Outer of multiple lists:

```
>> Outer[f, {a, b}, {x, y, z}, {1, 2}]
{{{f[a, x, 1], f[a, x, 2]}, {f[a, y, 1], f[a, y, 2]}, {f[a, z, 1], f[a, z, 2]}}, {{f[b, x, 1], f[b, x, 2]}, {f[b, y, 1], f[b, y, 2]}, {f[b, z, 1], f[b, z, 2]}}}
```

Arrays can be ragged:

```
>> Outer[Times, {1, 2}, {a, b}, {c, d, e}]
{{{a, b}, {c, d, e}}, {{2a, 2b}, {2c, 2d, 2e}}}
```

Word combinations:

```
>> Outer[StringJoin, {"", "re", "un"}, {"cover", "draw", "wind"}, {"", "ing", "s"}] // InputForm
{{{"cover", "covering", "covers"}, {"draw", "drawing", "draws"}, {"wind", "winding", "winds"}}, {{"recover", "recovering", "recovers"}, {"redraw", "redrawing", "redraws"}, {"rewind", "rewinding", "rewinds"}}, {{"uncover", "uncovering", "uncovers"}, {"undraw", "undrawing", "undraws"}, {"unwind", "unwinding", "unwinds"}}}
```

Compositions of trigonometric functions:

```
>> trigs = Outer[Composition, {Sin, Cos, Tan}, {ArcSin, ArcCos, ArcTan}]
{{Composition[Sin, ArcSin], Composition[Sin, ArcCos], Composition[
Sin, ArcTan]}, {Composition[Cos, ArcSin], Composition[
Cos, ArcCos], Composition[Cos, ArcTan]}, {Composition[
Tan, ArcSin], Composition[Tan, ArcCos], Composition[Tan, ArcTan]}}
```

Evaluate at 0:

```
>> Map[#[0] &, trigs, {2}]
{{0, 1, 0}, {1, 0, 1}, {0, ComplexInfinity, 0}}
```

RotationTransform

WMA link

```
RotationTransform[phi]
  gives a rotation by phi.
RotationTransform[phi, p]
  gives a rotation by phi around the point p.
```

ScalingTransform

WMA link

```
ScalingTransform[v]
  gives a scaling transform of v. v may be a scalar or a vector.
ScalingTransform[phi, p]
  gives a scaling transform of v that is centered at the point p.
```

ShearingTransform

WMA link

```
ShearingTransform[phi, {1, 0}, {0, 1}]
  gives a horizontal shear by the angle phi.
ShearingTransform[phi, {0, 1}, {1, 0}]
  gives a vertical shear by the angle phi.
ShearingTransform[phi, u, u, p]
  gives a shear centered at the point p.
```

TransformationFunction

WMA link

```
TransformationFunction[m]
  represents a transformation.
```

```
>> RotationTransform[Pi].TranslationTransform[{1, -1}]
TransformationFunction [{ {-1, 0, -1}, {0, -1, 1}, {0, 0, 1}}]

>> TranslationTransform[{1, -1}].RotationTransform[Pi]
TransformationFunction [{ {-1, 0, 1}, {0, -1, -1}, {0, 0, 1}}]
```

TranslationTransform

WMA link

```
TranslationTransform[v]
  gives a TransformationFunction that translates points by vector v.
```

```

>> t = TranslationTransform[{x0, y0}]
TransformationFunction [ {{1, 0, x0}, {0, 1, y0}, {0, 0, 1}}]

>> t[{x, y}]
{x + x0, y + y0}

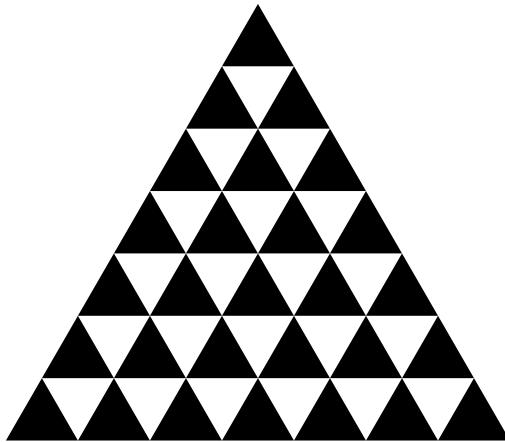
```

From Creating a Sierpinsky gasket with the missing triangles filled in:

```

>> Show[Graphics[Table[Polygon[TranslationTransform[{Sqrt[3] (i - j/2),
3 j/2}] /@ {{Sqrt[3]/2, -1/2}, {0, 1}, {-Sqrt[3]/2, -1/2}}], {i, 7},
{j, i}]]]

```



Transpose

Transpose (WMA)

```

Transpose[m]
transposes rows and columns in the matrix m.

```

```

>> square = {{1, 2, 3}, {4, 5, 6}}; Transpose[square]
{{1, 4}, {2, 5}, {3, 6}}

>> MatrixForm[%]

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}

>> matrix = {{1, 2}, {3, 4}, {5, 6}}; MatrixForm[Transpose[matrix]]

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$$$

```

Transpose is its own inverse. Transposing a matrix twice will give you back the same thing you started out with:

```

>> Transpose[Transpose[matrix]] == matrix
True

```

51. Testing Expressions

There are a number of functions for testing Expressions.

Functions that “ask a question” have names that end in “Q”. They return True for an explicit answer, and False otherwise.

Contents

Equality and Inequality	568	DisjointQ	577	Not (!)	583
BooleanQ	569	IntersectingQ	577	Or ()	583
Equal (==)	570	LevelQ	578	True	583
Greater (>)	571	MatrixQ	578	Xor (xor)	584
GreaterEqual (>=) .	571	MemberQ	579	Numerical Properties .	584
Inequality	571	NotListQ	579	CoprimeQ	584
Less (<)	572	SubsetQ	580	EvenQ	585
LessEqual (<=) . .	572	VectorQ	580	IntegerQ	585
Max	573	Logical Combinations .	580	MachineNumberQ	585
Min	573	AllTrue	580	Negative	586
SameQ (==>)	574	And (&&)	580	NonNegative	586
TrueQ	574	AnyTrue	581	NonPositive	586
Unequal (!=)	575	Equivalent		NumberQ	586
UnsameQ (=!=) . .	576	(\!\equiv) . . .	581	NumericQ	587
Expression Tests . . .	576	False	581	OddQ	587
ListQ	576	Implies (⇒)	582	Positive	588
MatchQ	576	Nand	582	PossibleZeroQ . .	588
List-Oriented Tests . .	576	NoneTrue	582	PrimeQ	589
ArrayQ	577	Nor	583		

Equality and Inequality

Equality and Inequality

BooleanQ

WMA link

`BooleanQ[expr]`
returns True if *expr* is either True or False.

```
>> BooleanQ[True]
True
>> BooleanQ[False]
True
>> BooleanQ[a]
False
```

```
>> BooleanQ[1 < 2]
True
```

Equal (==)

WMA link

```
Equal[x, y]
x == y
    is True if x and y are known to be equal, or False if x and y are known to be unequal, in which
    case case, Not[x == y] will be True.
Commutative properties apply, so if x == y then y == x.
For any expression x and y, Equal[x, y] == Not[Unequal[x, y]].
For any expression SameQ[x, y] implies Equal[x, y].
x == y == z == ...
    express a chain of equalities.
```

Numerical Equalities:

```
>> 1 == 1.
True

>> 5/3 == 3/2
False
```

Comparisons are done using the lower precision:

```
>> N[E, 100] == N[E, 150]
True
```

Compare an exact numeric expression and its corresponding approximate number:

```
>> Pi == N[Pi, 20]
True
```

Symbolic constants are compared numerically:

```
>> Pi == 3.14
False
```

Compare two exact numeric expressions; a numeric test may suffice to disprove equality:

```
>> Pi ^ E == E ^ Pi
False
```

Compare an exact expression against an approximate real number:

```
>> Pi == 3.1415^4
True
```

Real values are considered equal if they only differ in their last digits:

```
>> 0.739085133215160642 == 0.739085133215160641
True

>> 0.73908513321516064200000000 == 0.73908513321516064100000000
False
```

Numeric evaluation using Equal:

```
>> {Mod[6, 2] == 0, Mod[6, 4] == 0}
{True, False}
```

String equalities:

```

>> Equal["11", "11"]
True

>> Equal["121", "11"]
False

```

When we have symbols without values, the values are equal only if the symbols are equal:

```

>> Clear[a, b]; a == b
a==b

>> a == a
True

>> a = b; a == b
True

```

Comparison to mismatched types is False:

```

>> Equal[11, "11"]
False

```

Lists are compared based on their elements:

```

>> {{1}, {2}} == {{1}, {2}}
True

>> {1, 2} == {1, 2, 3}
False

```

For chains of equalities, the comparison is done amongst all the pairs. The evaluation is successful only if the equality is satisfied over all the pairs:

```

>> g[1] == g[1] == g[1]
True

>> g[1] == g[1] == g[r]
g[1]==g[1]==g[r]

```

Equality can also be combined with other inequality expressions, like:

```

>> g[1] == g[2] != g[3]
g[1]==g[2]&&g[2]!=g[3]

>> g[1] == g[2] <= g[3]
g[1]==g[2]&&g[2]<=g[3]

```

Equal with no parameter or an empty list is True:

```

>> Equal[] == True
True

```

Equal on one parameter or list element is also True

```

>> {Equal[x], Equal[1], Equal["a"]}
{True, True, True}

```

This degenerate behavior is the same for Unequal; empty or single-element lists are both Equal and Unequal.

Greater ($>$)

WMA link

`Greater[x, y]` or $x > y$
yields True if x is known to be greater than y .

Symbolic constants are compared numerically:

```
>> E > 1  
True
```

Greater operator can be chained:

```
>> a > b > c //FullForm  
Greater[a, b, c]  
  
>> 3 > 2 > 1  
True
```

GreaterEqual (\geq)

WMA link

`GreaterEqual[x, y]`
 $x \geq y$ or $x \geq y$
yields True if x is known to be greater than or equal to y .

Inequality

WMA link

`Inequality`
is the head of expressions involving different inequality operators (at least temporarily). Thus, it is possible to write chains of inequalities.

```
>> a < b <= c  
a < b&&b<=c  
  
>> Inequality[a, Greater, b, LessEqual, c]  
a > b&&b<=c  
  
>> 1 < 2 <= 3  
True  
  
>> 1 < 2 > 0  
True  
  
>> 1 < 2 < -1  
False
```

Less ($<$)

WMA link

Less[x , y] or $x < y$
yields True if x is known to be less than y .

```
>> 1 < 0  
False
```

LessEqual operator can be chained:

```
>> 2/18 < 1/5 < Pi/10  
True
```

Using less on an undefined symbol value:

```
>> 1 < 3 < x < 2  
1 < 3 < x < 2
```

LessEqual (\leq)

WMA link

LessEqual[x , y , ...] or $x \leq y$ or $x \leq y$
yields True if x is known to be less than or equal to y .

LessEqual operator can be chained:

```
>> LessEqual[1, 3, 3, 2]  
False  
>> 1 <= 3 <= 3  
True
```

Max

WMA link

Max[e_1 , e_2 , ..., e_i]
returns the expression with the greatest value among the e_i .

Maximum of a series of values:

```
>> Max[4, -8, 1]  
4  
>> Max[E - Pi, Pi, E + Pi, 2 E]  
E + \pi
```

Max flattens lists in its arguments:

```
>> Max[{1,2},3,{ -3,3.5,-Infinity},{{1/2}}]  
3.5
```

Max with symbolic arguments remains in symbolic form:

```
>> Max[x, y]  
Max[x,y]  
>> Max[5, x, -3, y, 40]  
Max[40,x,y]
```

With no arguments, Max gives `-Infinity`:

```
>> Max []
      -∞
```

Max does not compare strings or symbols:

```
>> Max[-1.37, 2, "a", b]
      Max[2, a, b]
```

Min

WMA link

```
Min[e_1, e_2, ..., e_i]
```

returns the expression with the lowest value among the e_i .

Minimum of a series of values:

```
>> Min[4, -8, 1]
      -8

>> Min[E - Pi, Pi, E + Pi, 2 E]
      E - π
```

Min flattens lists in its arguments:

```
>> Min[{1,2},3,{ -3,3.5,-Infinity},{{1/2}}]
      -∞
```

Min with symbolic arguments remains in symbolic form:

```
>> Min[x, y]
      Min[x, y]

>> Min[5, x, -3, y, 40]
      Min[-3, x, y]
```

With no arguments, Min gives `Infinity`:

```
>> Min []
      ∞
```

SameQ (==)

WMA link

```
SameQ[x, y]
```

$x == y$

returns True if x and y are structurally identical. Commutative properties apply, so if $x == y$ then $y == x$.

- SameQ requires exact correspondence between expressions, except that it still considers Real numbers equal if they differ in their last binary digit.
- $e1 == e2 == e3$ gives True if all the e_i 's are identical.
- SameQ[] and SameQ[*expr*] always yield True.

Any object is the same as itself:

```
>> a == a
      True
```

Degenerate cases of `SameQ` showing off how you can chain `==`:

```
>> SameQ[a] === SameQ[] === True
      True
```

Unlike `Equal`, `SameQ` only yields `True` if x and y have the same type:

```
>> {1==1., 1==1.}
      {True, False}

>> 2./9. === .2222222222222222`15.9546
      True
```

The comparison consider just the lowest precision

```
>> .2222222`6 === .2222`3
      True
```

Notice the extra decimal in the rhs. Because the internal representation, `.2222`3` is not equivalent to `.222222`6`:

```
>> .2222222`6 === .2222`3
      False
```

`15.9546` is the value of `$MaxPrecision`

TrueQ

WMA link

`TrueQ[expr]`
returns `True` if and only if $expr$ is `True`.

```
>> TrueQ[True]
      True

>> TrueQ[False]
      False

>> TrueQ[a]
      False
```

Unequal (`!=`)

WMA link

`Unequal[x, y]` or $x \neq y$ or $x \neq y$
is `False` if x and y are known to be equal, or `True` if x and y are known to be unequal.
Commutative properties apply so if $x \neq y$ then $y \neq x$.
For any expression x and y , `Unequal[x, y] == Not[Equal[x, y]]`.

```
>> 1 != 1.
      False
```

Comparisons can be chained:

```
>> 1 != 2 != 3
      True

>> 1 != 2 != x
      1!=2!=x
```

Strings are allowed:

```
>> Unequal["11", "11"]
False
```

Comparison to mismatched types is True:

```
>> Unequal[11, "11"]
True
```

Lists are compared based on their elements:

```
>> {1} != {2}
True

>> {1, 2} != {1, 2}
False

>> {a} != {a}
False

>> "a" != "b"
True

>> "a" != "a"
False
```

Unequal using an empty parameter or list, or a list with one element is True. This is the same as 'Equal'.

```
>> {Unequal[], Unequal[x], Unequal[1]}
{True, True, True}
```

UnsameQ ($=!=$)

WMA link

```
UnsameQ[x, y]
x =!= y
    returns True if x and y are not structurally identical. Commutative properties apply, so if x =!= y, then y =!= x.
```

```
>> a =!= a
False

>> 1 =!= 1.
True
```

UnsameQ accepts any number of arguments and returns True if all expressions are structurally distinct:

```
>> 1 =!= 2 =!= 3 =!= 4
True
```

UnsameQ returns False if any expression is identical to another:

```
>> 1 =!= 2 =!= 1 =!= 4
False
```

UnsameQ[] and UnsameQ[expr] return True:

```
>> UnsameQ[]
True
```

```
>> UnsameQ[expr]
True
```

Expression Tests

Expression Test

ListQ

WMA link

```
ListQ[expr]
tests whether expr is a List.
```

```
>> ListQ[{1, 2, 3}]
True
>> ListQ[{{1, 2}, {3, 4}}]
True
>> ListQ[x]
False
```

MatchQ

WMA link

```
MatchQ[expr, form]
tests whether expr matches form.
```

```
>> MatchQ[123, _Integer]
True
>> MatchQ[123, _Real]
False
>> MatchQ[_Integer][123]
True
>> MatchQ[3, Pattern[3]]
First element in pattern Pattern[3] is not a valid pattern name.
False
```

List-Oriented Tests

List-Oriented Test

ArrayQ

WMA

```

ArrayQ[expr]
  tests whether expr is a full array.
ArrayQ[expr, pattern]
  also tests whether the array depth of expr matches pattern.
ArrayQ[expr, pattern, test]
  furthermore tests whether test yields True for all elements of expr. ArrayQ[expr] is equivalent to ArrayQ[expr, _, True&].

```

```

>> ArrayQ[a]
False

>> ArrayQ[{a}]
True

>> ArrayQ[{{a}}, {{b,c}}]
False

>> ArrayQ[{{a, b}, {c, d}}, 2, SymbolQ]
True

```

DisjointQ

WMA link

```

DisjointQ[a, b]
  gives True if a and b are disjoint, or False if a and b have any common elements.

```

IntersectingQ

WMA link

```

IntersectingQ[a, b]
  gives True if there are any common elements in $a and $b, or False if $a and $b are disjoint.

```

LevelQ

```

LevelQ[expr]
  tests whether expr is a valid level specification. This function is primarily used in function patterns for specifying type of a parameter.

```

```

>> LevelQ[2]
True

>> LevelQ[{2, 4}]
True

>> LevelQ[Infinity]
True

>> LevelQ[a + b]
False

```

We will define MyMap with the “level” parameter as a synonym for the Builtin Map equivalent:

```
>> MyMap[f_, expr_, Pattern[levelspec, _?LevelQ]] := Map[f, expr,
  levelspec]

>> MyMap[f, {{a, b}, {c, d}}, {2}]
 {{f[a], f[b]}, {f[c], f[d]}}
```



```
>> Map[f, {{a, b}, {c, d}}, {2}]
 {{f[a], f[b]}, {f[c], f[d]}}
```

But notice that when we pass an invalid level specification, MyMap does not match and therefore does not pass the arguments through to Map. So we do not see the error message that Map would normally produce

```
>> Map[f, {{a, b}, {c, d}}, x]
 Level specification x is not of the form n, {n}, or {m, n}.
 Map[f, {{a, b}, {c, d}}, x]

>> MyMap[f, {{a, b}, {c, d}}, {1, 2, 3}]
 MyMap[f, {{a, b}, {c, d}}, {1, 2, 3}]
```

MatrixQ

WMA link

`MatrixQ[m]`

gives True if m is a list of equal-length lists.

`MatrixQ[m, f]`

gives True only if $f[x]$ returns True for when applied to element x of the matrix m .

```
>> MatrixQ[{{1, 3}, {4.0, 3/2}}, NumberQ]
 True
```

These are not matrices:

```
>> MatrixQ[{{1}, {1, 2}}] (* first row should have length two *)
 False

>> MatrixQ[Array[a, {1, 1, 2}]]
 False
```

Supply a test function parameter to generalize and specialize:

```
>> MatrixQ[{{1, 2}, {3, 4 + 5}}, Positive]
 True

>> MatrixQ[{{1, 2 I}, {3, 4 + 5}}, Positive]
 False
```

MemberQ

WMA link

`MemberQ[list, pattern]`

returns True if $pattern$ matches any element of $list$, or False otherwise.

```

>> MemberQ[{a, b, c}, b]
True

>> MemberQ[{a, b, c}, d]
False

>> MemberQ[{"a", b, f[x]}, _?NumericQ]
False

>> MemberQ[_List][{{}}]
True

```

NotListQ

`NotListQ[expr]`

returns `True` if `expr` is not a list. This function is primarily used in function patterns for specifying type of a parameter.

Consider this definition for taking the deriviate `Sin` of a function:

```
>> MyD[Sin[f_],x_?NotListQ] := D[f,x]*Cos[f]
```

=

We use "MyD" above to distinguish it from the Builtin `D`. Now let's try it:

```
>> MyD[Sin[2 x], x]
2Cos[2x]
```

And compare it with the Builtin derivative function `D`:

```
>> D[Sin[2 x], x]
2Cos[2x]
```

Note however the pattern only matches if the `x` parameter is not a list:

```
>> MyD[{Sin[2], Sin[4]}, {1, 2}]
MyD[{Sin[2], Sin[4]}, {1, 2}]
```

SubsetQ

WMA link

`SubsetQ[list1, list2]`

returns `True` if `list2` is a subset of `list1`, and `False` otherwise.

```
>> SubsetQ[{1, 2, 3}, {3, 1}]
True
```

The empty list is a subset of every list:

```
>> SubsetQ[{}, {}]
True

>> SubsetQ[{1, 2, 3}, {}]
True
```

Every list is a subset of itself:

```
>> SubsetQ[{1, 2, 3}, {1, 2, 3}]
True
```

VectorQ

WMA link

VectorQ[*v*]
returns True if *v* is a list of elements which are not themselves lists.
VectorQ[*v*, *f*]
returns True if *v* is a vector and *f*[*x*] returns True for each element *x* of *v*.

```
>> VectorQ[{a, b, c}]
True
```

Logical Combinations

Logical Combination

AllTrue

WMA link

AllTrue[*expr*1, *expr*2, ...], *test*]
returns True if all applications of *test* to *expr*1, *expr*2, ... evaluate to True.
AllTrue[*list*, *test*, *level*]
returns True if all applications of *test* to items of *list* at *level* evaluate to True.
AllTrue[*test*]
gives an operator that may be applied to expressions.

```
>> AllTrue[{2, 4, 6}, EvenQ]
True
>> AllTrue[{2, 4, 7}, EvenQ]
False
```

And (&&)

WMA link

And[*expr*1, *expr*2, ...]
*expr*1 && *expr*2 && ...
evaluates each expression in turn, returning False as soon as an expression evaluates to False. If all expressions evaluate to True, And returns True.

```
>> True && True && False
False
```

If an expression does not evaluate to True or False, And returns a result in symbolic form:

```
>> a && b && True && c
a&&b&&c
```

AnyTrue

WMA link

```
AnyTrue[{expr1, expr2, ...}, test]
    returns True if any application of test to expr1, expr2, ... evaluates to True.
AnyTrue[list, test, level]
    returns True if any application of test to items of list at level evaluates to True.
AnyTrue[test]
    gives an operator that may be applied to expressions.
```

```
>> AnyTrue[{1, 3, 5}, EvenQ]
False
>> AnyTrue[{1, 4, 5}, EvenQ]
True
```

Equivalent (\Equiv)

WMA link

```
Equivalent[expr1, expr2, ...]
expr1
Equiv expr2
Equiv ...
    is equivalent to (expr1 && expr2 && ...) || (!expr1 && !expr2 && ...)
```

```
>> Equivalent[True, True, False]
False
```

If all expressions do not evaluate to True or False, Equivalent returns a result in symbolic form:

```
>> Equivalent[a, b, c]
a\[Equivalent]b\[Equivalent]c
```

Otherwise, Equivalent returns a result in DNF

```
>> Equivalent[a, b, True, c]
a&&b&&c
```

False

WMA link

```
False
represents the Boolean false value.
```

Implies (\Rightarrow)

WMA link

```
Implies[expr1, expr2]
```

```
expr1  $\Rightarrow$  expr2
```

evaluates each expression in turn, returning True as soon as the first expression evaluates to False. If the first expression evaluates to True, Implies returns the second expression.

```
>> Implies[False, a]
```

```
True
```

```
>> Implies[True, a]
```

```
a
```

If an expression does not evaluate to True or False, Implies returns a result in symbolic form:

```
>> Implies[a, Implies[b, Implies[True, c]]]
```

```
aImpliesbImpliesc
```

Nand

WMA link

```
Nand[expr1, expr2, ...]
```

```
expr1  $\overline{\wedge}$  expr2  $\overline{\wedge}$  ...
```

Implements the logical NAND function. The same as Not [And [expr1, expr2, ...]]

```
>> Nand[True, False]
```

```
True
```

NoneTrue

WMA link

```
NoneTrue[{expr1, expr2, ...}, test]
```

returns True if no application of test to expr1, expr2, ... evaluates to True.

```
NoneTrue[list, test, level]
```

returns True if no application of test to items of list at level evaluates to True.

```
NoneTrue[test]
```

gives an operator that may be applied to expressions.

```
>> NoneTrue[{1, 3, 5}, EvenQ]
```

```
True
```

```
>> NoneTrue[{1, 4, 5}, EvenQ]
```

```
False
```

Nor

WMA link

```
Nor[expr1, expr2, ...]
```

```
expr1  $\overline{\vee}$  expr2  $\overline{\vee}$  ...
```

Implements the logical NOR function. The same as Not [Or [expr1, expr2, ...]]

```
>> Nor[True, False]
False
```

Not (!)

WMA link

```
Not [expr]
!expr
negates the logical expression expr.
```

```
>> !True
False
>> !False
True
>> !b
!b
```

Or (||)

WMA link

```
Or [expr1, expr2, ...]
expr1 || expr2 || ...
evaluates each expression in turn, returning True as soon as an expression evaluates to True .
If all expressions evaluate to False , Or returns False.
```

```
>> False || True
True
```

If an expression does not evaluate to True or False , Or returns a result in symbolic form:

```
>> a || False || b
a||b
```

True

WMA link

```
True
represents the Boolean true value.
```

Xor (xor)

WMA link

```
Xor [expr1, expr2, ...]
expr1 ⊕ expr2 ⊕ ...
evaluates each expression in turn, returning True as soon as not all expressions evaluate to
the same value. If all expressions evaluate to the same value, Xor returns False.
```

```
>> Xor[False, True]
True
>> Xor[True, True]
False
```

If an expression does not evaluate to `True` or `False`, `Xor` returns a result in symbolic form:

```
>> Xor[a, False, b]
a\[Xor]b
```

Numerical Properties

Numerical Propertie

CoprimeQ

WMA link

`CoprimeQ[x, y]`

tests whether x and y are coprime by computing their greatest common divisor.

```
>> CoprimeQ[7, 9]
True
>> CoprimeQ[-4, 9]
True
>> CoprimeQ[12, 15]
False
```

`CoprimeQ` also works for complex numbers

```
>> CoprimeQ[1+2I, 1-I]
True
>> CoprimeQ[4+2I, 6+3I]
True
>> CoprimeQ[2, 3, 5]
True
>> CoprimeQ[2, 4, 5]
False
```

EvenQ

WMA link

`EvenQ[x]`

returns `True` if x is even, and `False` otherwise.

```
>> EvenQ[4]
True
```

```
>> EvenQ[-3]
False
>> EvenQ[n]
False
```

IntegerQ

WMA link

```
IntegerQ[expr]
returns True if expr is an integer, and False otherwise.
```

```
>> IntegerQ[3]
True
>> IntegerQ[Pi]
False
```

MachineNumberQ

WMA link

```
MachineNumberQ[expr]
returns True if expr is a machine-precision real or complex number.
```

```
= True
>> MachineNumberQ[3.14159265358979324]
False
>> MachineNumberQ[1.5 + 2.3 I]
True
>> MachineNumberQ[2.71828182845904524 + 3.14159265358979324 I]
False
```

Negative

WMA link

```
Negative[x]
returns True if x is a negative real number.
```

```
>> Negative[0]
False
>> Negative[-3]
True
>> Negative[10/7]
False
```

```
>> Negative[1+2I]
False
>> Negative[a + b]
Negative[a + b]
```

NonNegative

WMA link

NonNegative[*x*]
returns True if *x* is a positive real number or zero.

```
>> {Positive[0], NonNegative[0]}
{False, True}
```

NonPositive

WMA link

NonPositive[*x*]
returns True if *x* is a negative real number or zero.

```
>> {Negative[0], NonPositive[0]}
{False, True}
```

NumberQ

WMA link

NumberQ[*expr*]
returns True if *expr* is an explicit number, and False otherwise.

```
>> NumberQ[3+I]
True
>> NumberQ[5!]
True
>> NumberQ[Pi]
False
```

NumericQ

WMA link

NumericQ[*expr*]
tests whether *expr* represents a numeric quantity.

```
>> NumericQ[2]
True
```

```

>> NumericQ[Sqrt[Pi]]
True

>> NumberQ[Sqrt[Pi]]
False

```

It is possible to set that a symbol is numeric or not by assign a boolean value to "NumericQ"

```

>> NumericQ[a]=True
True

>> NumericQ[a]
True

>> NumericQ[Sin[a]]
True

```

Clear and ClearAll do not restore the default value.

```

>> Clear[a]; NumericQ[a]
True

>> ClearAll[a]; NumericQ[a]
True

>> NumericQ[a]=False; NumericQ[a]
False

```

NumericQ can only set to True or False

```

>> NumericQ[a] = 37
Cannot set NumericQ[a] to 37; the lhs argument must be a symbol and
the rhs must be True or False.
37

```

OddQ

WMA link

```

OddQ[x]
    returns True if x is odd, and False otherwise.

```

```

>> OddQ[-3]
True

>> OddQ[0]
False

```

Positive

WMA link

```

Positive[x]
    returns True if x is a positive real number.

```

```

>> Positive[1]
True

```

`Positive` returns `False` if x is zero or a complex number:

```
>> Positive[0]
False

>> Positive[1 + 2 I]
False
```

PossibleZeroQ

WMA link

`PossibleZeroQ[expr]`

returns `True` if basic symbolic and numerical methods suggest that `expr` has value zero, and `False` otherwise.

Test whether a numeric expression is zero:

```
>> PossibleZeroQ[E^(I Pi/4) - (-1)^(1/4)]
True
```

The determination is approximate.

Test whether a symbolic expression is likely to be identically zero:

```
>> PossibleZeroQ[(x + 1)(x - 1) - x^2 + 1]
True

>> PossibleZeroQ[(E + Pi)^2 - E^2 - Pi^2 - 2 E Pi]
True
```

Show that a numeric expression is nonzero:

```
>> PossibleZeroQ[E^Pi - Pi^E]
False

>> PossibleZeroQ[1/x + 1/y - (x + y)/(x y)]
True
```

Decide that a numeric expression is zero, based on approximate computations:

```
>> PossibleZeroQ[2^(2 I) - 2^(-2 I) - 2 I Sin[Log[4]]]
True

>> PossibleZeroQ[Sqrt[x^2] - x]
False
```

PrimeQ

WMA link

`PrimeQ[n]`

returns `True` if n is a prime number.

For very large numbers, `PrimeQ` uses probabilistic prime testing, so it might be wrong sometimes (a number might be composite even though `PrimeQ` says it is prime). The algorithm might be changed in the future.

```
>> PrimeQ[2]
True
```

```
>> PrimeQ[-3]
True

>> PrimeQ[137]
True

>> PrimeQ[2 ^ 127 - 1]
True

All prime numbers between 1 and 100:
>> Select[Range[100], PrimeQ]
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97}

PrimeQ has attribute Listable:
>> PrimeQ[Range[20]]
{False,True,True,False,True,False,True,False,False,True,True,False,True,False,True,False,True,Fa
```

52. The Main Loop

An interactive session operates a loop, called the “main loop” in this way:

- read input
- process input
- format and print results
- repeat

As part of this loop, various global objects in this section are consulted.

There are a variety of “hooks” that allow you to insert functions to be applied to the expressions at various stages in the main loop.

If you assign a function to the global variable \$PreRead it will be applied with the input that is read in the first step listed above.

Similarly, if you assign a function to global variable \$Pre, it will be applied with the input before processing the input, the second step listed above.

Contents

\$HistoryLength	591	\$Pre	592	\$SyntaxHandler	592
\$Line	591	\$PrePrint	592	In	593
\$Post	591	\$PreRead	592	Out	593

\$HistoryLength

WMA

\$HistoryLength
specifies the maximum number of In and Out entries.

```
>> $HistoryLength  
100  
>> $HistoryLength = 1;  
  
>> 42  
42  
>> %  
42  
>> %%  
%3  
  
>> $HistoryLength = 0;  
  
>> 42  
42
```

```
>> %
%7
```

\$Line

WMA

```
$Line
holds the current input line number.
```

```
>> $Line
1

>> $Line
2

>> $Line = 12;

>> 2 * 5
10

>> Out[13]
10

>> $Line = -1;
Non-negative integer expected.
```

\$Post

WMA

```
$Post
is a global variable whose value, if set, is applied to every output expression.
```

\$Pre

WMA

```
$Pre
is a global variable whose value, if set, is applied to every input expression.
```

Set *Timing* as the \$Pre function, stores the elapsed time in a variable, stores just the result in Out[\$Line] and print a formatted version showing the elapsed time

```
>> $Pre := (Print["[Processing input...]"]; #1) &

>> $Post := (Print["[Storing result...]"]; #1) &
[Processing input...]
[Storing result...]

>> $PrePrint := (Print["The result is:"], {TimeUsed[], #1}) &
[Processing input...]
[Storing result...]
```

```
>> 2 + 2
[Processing input...]
[Storing result...]
The result is:
{389.354, 4}

>> $Pre = .; $Post = .; $PrePrint = .; $ElapsedTime = .;
[Processing input...]

>> 2 + 2
4
```

\$PrePrint

WMA

\$PrePrint

is a global variable whose value, if set, is applied to every output expression before it is printed.

\$PreRead

WMA

\$PreRead

is a global variable whose value, if set, is applied to the text or box form of every input expression before it is fed to the parser.

(Not implemented yet)

\$SyntaxHandler

WMA

\$SyntaxHandler

is a global variable whose value, if set, is applied to any input string that is found to contain a syntax error.

(Not implemented yet)

In

WMA

In[*k*]

gives the *k*th line of input.

```
>> x = 1
1
>> x = x + 1
2
```

```

>> Do[In[2], {3}]

>> x
5

>> In[-1]
5

>> Definition[In]
Attributes[In] = {Listable, Protected}
In[6] = Definition[In]
In[5] = In[-1]
In[4] = x
In[3] = Do[In[2], {3}]
In[2] = x = x + 1
In[1] = x = 1

```

Out

WMA

<pre> Out[k] %k gives the result of the <i>k</i>th input line. %, %%, etc. gives the result of the previous input line, of the line before the previous input line, etc. </pre>

```

>> 42
42

>> %
42

>> 43;
43

>> %
43

>> 44
44

>> %1
42

>> %%
44

>> Hold[Out[-1]]
Hold[%]

>> Hold[%4]
Hold[%4]

>> Out[0]
Out[0]

```

53. Tracing Built-in Functions

Built-in Function Tracing provides one high-level way understand what is getting evaluated and where the time is spent in evaluation.

With this, it may be possible for both users and implementers to follow how Mathics3 arrives at its results, or guide how to speed up expression evaluation.

Contents

\$TraceBuiltins	594	ClearTrace	596	TraceBuiltins	597
\$TraceEvaluation	595	PrintTrace	596	TraceEvaluation	598

\$TraceBuiltins

`$TraceBuiltins`

A Boolean Built-in variable when True collects function evaluation statistics.

Setting this variable True will enable statistics collection for Built-in functions that are evaluated. In contrast to `TraceBuiltins[]` statistics are accumulated and over several inputs, and are not shown after each input is evaluated. By default this setting is False.

```
>> $TraceBuiltins = True  
True
```

Tracing is enabled, so the expressions entered and evaluated will have statistics collected for the evaluations.

```
>> x  
x
```

To print the statistics collected, use `PrintTrace[]` :

```
>> PrintTrace[]
```

To clear statistics collected use `ClearTrace[]` :

```
>> ClearTrace[]
```

`$TraceBuiltins` cannot be set to a non-boolean value.

```
>> $TraceBuiltins = x  
x should be True or False.  
x
```

\$TraceEvaluation

`$TraceEvaluation`

A Boolean variable which when set True traces Expression evaluation calls and returns.

三

`$TraceEvaluation = False`

```
>>> $TraceEvaluation = False
Evaluating: System`Set[System`$TraceEvaluation, System`False]
Evaluating: System`Set
Evaluating: System`False

False

>>> $TraceEvaluation
False

>>> a + a
2a

$TraceEvaluation cannot be set to a no
```

```
>> $TraceEvaluation = x
```

x should be True or False.
 x

ClearTrace

ClearTrace[]
Clear the statistics collected for Built-in Functions

First, set up Builtin-function tracing:

```
>> $TraceBuiltins = True  
True
```

Dump Builtin-Function statistics gathered in running that assignment:

```
>> PrintTrace[]  
  
>> ClearTrace[]
```

PrintTrace

PrintTrace[]

Print statistics collected for Built-in Functions

Sort Options:

- count
- name
- time

Note that in a browser the information only appears in a console.

If \$TraceBuiltins was never set to True , this will print an empty list.

```
>> PrintTrace[]  
  
>> $TraceBuiltins = True  
True  
  
>> PrintTrace[SortBy -> "time"]
```

TraceBuiltins

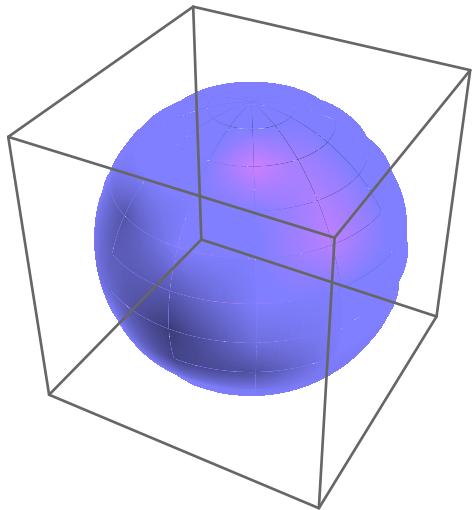
TraceBuiltins[*expr*]

Evaluate *expr* and then print a list of the Built-in Functions called in evaluating *expr* along with the number of times is each called, and combined elapsed time in milliseconds spent in each.

Sort Options:

- count
- name
- time

```
>> TraceBuiltins[Graphics3D[Tetrahedron[]]]
```



By default, the output is sorted by the number of calls of the builtin from highest to lowest:

```
>> TraceBuiltins[Times[x, x], SortBy->"count"]
```

$$x^2$$

You can have results ordered by name, or time.

Trace an expression and list the result by time from highest to lowest.

```
>> TraceBuiltins[Times[x, x], SortBy->"time"]
```

$$x^2$$

TraceEvaluation

TraceEvaluation[*expr*]

Evaluate *expr* and print each step of the evaluation.

```
>> TraceEvaluation[(x + x)^2]
```

```
Evaluating: System`Power[System`Plus[Global`x, Global`x], 2]
Evaluating: System`Power
Evaluating: System`Plus[Global`x, Global`x]
Evaluating: System`Plus
Evaluating: Global`x
Evaluating: Global`x
-> System`Times[2, Global`x]
Evaluating: System`Times
Evaluating: Global`x
Evaluating: System`Power[Global`x, 2]
Evaluating: System`Power
Evaluating: Global`x
Evaluating: Global`x
-> System`Times[4, System`Power[Global`x, 2]]
Evaluating: System`Times
Evaluating: System`Power[Global`x, 2]
```

$$4x^2$$

```

>> TraceEvaluation[(x + x)^2, ShowTimeBySteps->True]
0.000375271
Evaluating: System`Power[System`Plus[Global`x, Global`x], 2]
0.00413585
Evaluating: System`Power
0.00782394
Evaluating: System`Plus[Global`x, Global`x]
0.0114872
Evaluating: System`Plus
0.0151558
Evaluating: Global`x
0.0188198
Evaluating: Global`x
-> System`Times[2, Global`x]
0.024493
Evaluating: System`Times
0.0282567
Evaluating: Global`x
0.0326107
Evaluating: System`Power[Global`x, 2]
0.0362749
Evaluating: System`Power
0.0399296
Evaluating: Global`x
0.0439322
Evaluating: Global`x
-> System`Times[4, System`Power[Global`x, 2]]
0.0496171
Evaluating: System`Times
0.0533211
Evaluating: System`Power[Global`x, 2]

```

$$4x^2$$

54. Units and Quantities

Contents

KnownUnitQ	599	QuantityMagnitude	600	QuantityUnit	600
Quantity	599	QuantityQ	600	UnitConvert	601

KnownUnitQ

WMA link

`KnownUnitQ[unit]`

returns True if *unit* is a canonical unit, and False otherwise.

```
>> KnownUnitQ["Feet"]
True
>> KnownUnitQ["Foo"]
False
```

Quantity

WMA link

`Quantity[magnitude, unit]`

represents a quantity with size *magnitude* and unit specified by *unit*.

`Quantity[unit]`

assumes the magnitude of the specified *unit* to be 1.

```
>> Quantity["Kilogram"]
1kilogram
>> Quantity[10, "Meters"]
10meter
>> Quantity[{10,20}, "Meters"]
{10meter,20meter}
```

QuantityMagnitude

WMA link

```
QuantityMagnitude[quantity]
  gives the amount of the specified quantity.
QuantityMagnitude[quantity, unit]
  gives the value corresponding to quantity when converted to unit.
```

```
>> QuantityMagnitude[Quantity["Kilogram"]]
1
>> QuantityMagnitude[Quantity[10, "Meters"]]
10
>> QuantityMagnitude[Quantity[{10,20}, "Meters"]]
{10,20}
```

QuantityQ

WMA link

```
QuantityQ[expr]
  return True if expr is a valid Association object, and False otherwise.
```

```
>> QuantityQ[Quantity[3, "Meters"]]
True
>> QuantityQ[Quantity[3, "Maters"]]
Unable to interpret unit specification Maters.
False
```

QuantityUnit

WMA link

```
QuantityUnit[quantity]
  returns the unit associated with the specified quantity.
```

```
>> QuantityUnit[Quantity["Kilogram"]]
kilogram
>> QuantityUnit[Quantity[10, "Meters"]]
meter
>> QuantityUnit[Quantity[{10,20}, "Meters"]]
{meter,meter}
```

UnitConvert

WMA link

`UnitConvert [quantity, targetunit]`

converts the specified *quantity* to the specified *targetunit*.

`UnitConvert [quantity]`

converts the specified *quantity* to its "SIBase" units.

Convert from miles to kilometers:

```
>> UnitConvert [Quantity[5.2, "miles"], "kilometers"]
```

8.36859kilometer

Convert a Quantity object to the appropriate SI base units:

```
>> UnitConvert [Quantity[3.8, "Pounds"]]
```

1.72365kilogram

Part III.

Mathics3 Modules

1. Graphs - Vertices and Edges

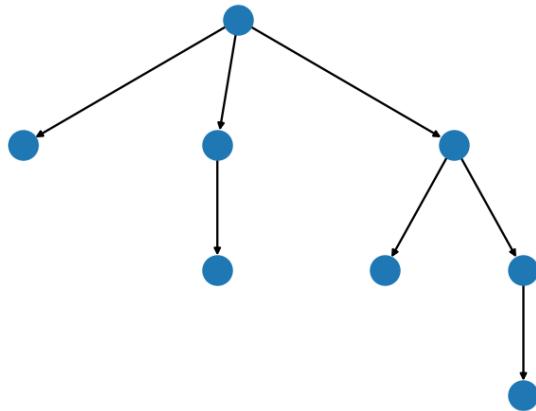
A Graph is a tuple of a set of Nodes and Edges.

Mathics3 Module that provides functions and variables for working with graphs.

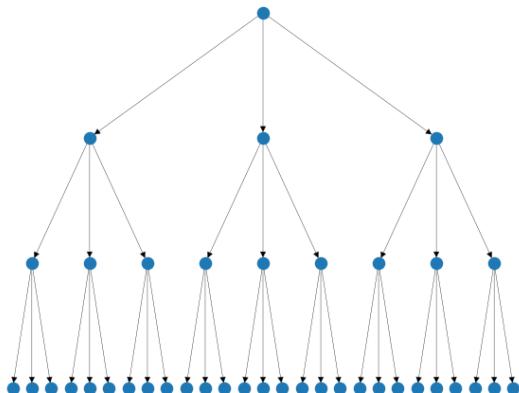
Examples:

```
>> LoadModule["pymathics.graph"]
pymathics.graph

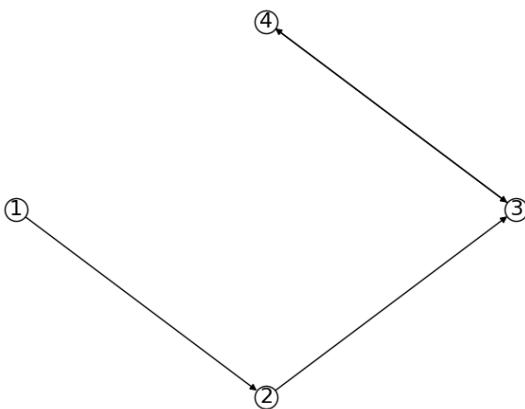
>> BinomialTree[3, DirectedEdges->True]
```



```
>> BalancedTree[3, 3]
```



```
>> g = Graph[{1 -> 2, 2 -> 3, 3 <-> 4}, VertexLabels->True]
```



```
>> ConnectedComponents[g]
{{3,4},{2},{1}}
>> WeaklyConnectedComponents[g]
{{1,2,3,4}}
>> GraphDistance[g, 1, 4]
3
>> GraphDistance[g, 3, 2]
∞
```

NetworkX does the heavy lifting here.

Contents

Centralities	605	FindVertexCut	612	GraphDistance	622
BetweennessCentrality	606	Graph	613	VertexCount	623
ClosenessCentrality	607	HighlightGraph	613	VertexDegree	623
DegreeCentrality	607	Property	614	Graph Operations and Modifications	623
Eigenvector-Centrality	608	PropertyValue	614	FindSpanningTree	623
HITSCentrality	608	UndirectedEdge	614	Graph Properties and Measurements	623
KatzCentrality	609	VertexAdd	615	AcyclicGraphQ	624
PageRankCentrality	610	VertexConnectivity	616	ConnectedGraphQ	624
Core routines for working with Graphs.	610	VertexDelete	617	DirectedGraphQ	625
AdjacencyList	610	VertexIndex	617	LoopFreeGraphQ	625
DirectedEdge	610	VertexList	618	MixedGraphQ	625
EdgeConnectivity	610	Curated Graphs	618	MultigraphQ	626
EdgeDelete	611	GraphData	618	PathGraphQ	626
EdgeIndex	611	Graph Components and Connectivity	618	PlanarGraphQ	626
EdgeList	611	ConnectedComponents	620	SimpleGraphQ	627
EdgeRules	611	WeaklyConnectedComponents	621	Parametric Graphs	627
FindShortestPath	612	Graph Measures and Metrics	621	BalancedTree	627
		EdgeCount	621	BarbellGraph	628
				BinomialTree	628

CompleteGraph	629	KaryTree	633	Random Graphs	635
CompleteKaryTree	630	LadderGraph	633	RandomGraph	635
CycleGraph	630	PathGraph	634	Trees	635
GraphAtlas	631	RandomTree	634	TreeGraph	636
HknHararyGraph	631	StarGraph	635	TreeGraphQ	636
HmnHararyGraph	632				

Centralities

Centralities

Centralities

Routines to evaluate centralities of a graph.

In graph theory and network analysis, the centrality is a ranking between pairs of node according some metric.

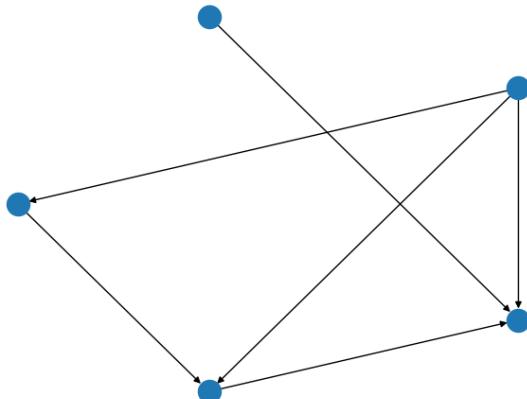
BetweennessCentrality

Betweenness centrality (NetworkX, WMA)

BetweennessCentrality[g]

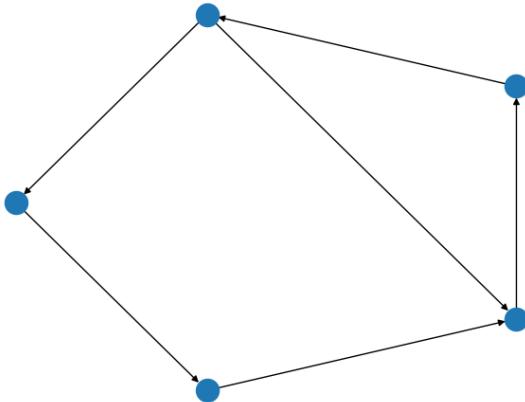
gives a list of betweenness centralities for the vertices in a Graph or a list of edges g.

```
>> g = Graph[{a -> b, b -> c, d -> c, d -> a, e -> c, d -> b}]
```



```
>> BetweennessCentrality[g]
```

```
>> g = Graph[{a -> b, b -> c, c -> d, d -> e, e -> c, e -> a}]
```



```
>> BetweennessCentrality[g]
```

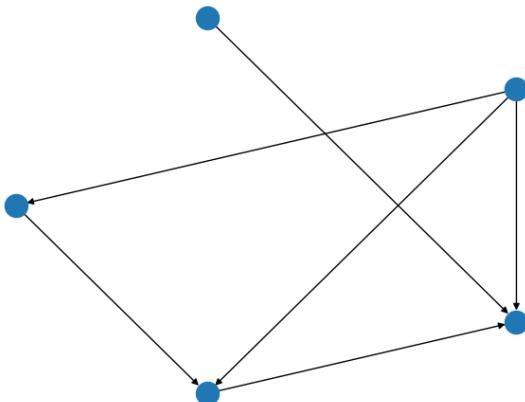
ClosenessCentrality

Betweenness centrality (NetworkX, WMA)

```
ClosenessCentrality[g]
```

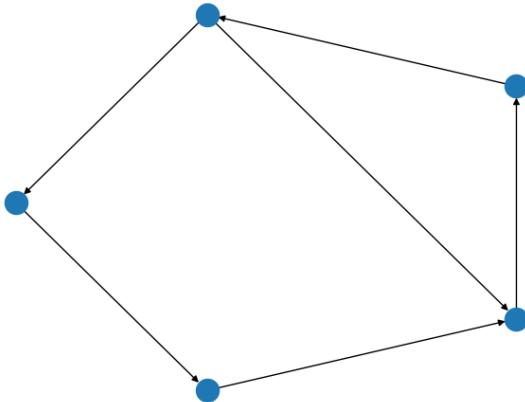
gives a list of closeness centralities for the vertices in a Graph or a list of edges g.

```
>> g = Graph[{a -> b, b -> c, d -> c, d -> a, e -> c, d -> b}]
```



```
>> ClosenessCentrality[g]
```

```
>> g = Graph[{a -> b, b -> c, c -> d, d -> e, e -> c, e -> a}]
```



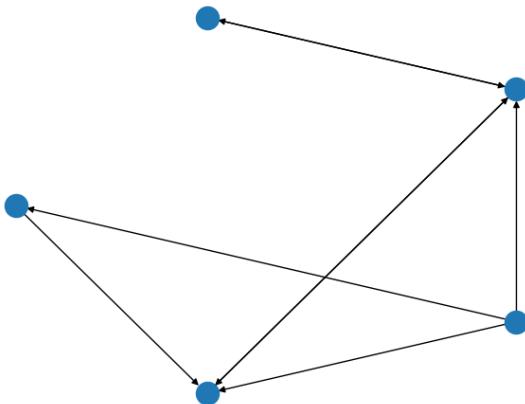
```
>> ClosenessCentrality[g]
```

DegreeCentrality

Degree centrality (NetworkX, WMA)

```
DegreeCentrality[g]  
gives a list of degree centralities for the vertices in a Graph or a list of edges g.
```

```
>> g = Graph[{a -> b, b <-> c, d -> c, d -> a, e <-> c, d -> b}]
```



```
>> DegreeCentrality[g]  
{2,4,3,5,2}  
>> DegreeCentrality[g, "In"]  
{1,3,0,3,1}  
>> DegreeCentrality[g, "Out"]  
{1,1,3,2,1}
```

EigenvectorCentrality

Eigenvector Centrality (NetworkX,WMA)

```
EigenvectorCentrality[g]
    gives a list of eigenvector centralities for the vertices in the graph g.
EigenvectorCentrality[g, "In"]
    gives a list of eigenvector in-centralities for the vertices in the graph g.
EigenvectorCentrality[g, "Out"]
    gives a list of eigenvector out-centralities for the vertices in the graph g.
```

```
>> g = Graph[{a -> b, b -> c, c -> d, d -> e, e -> c, e -> a}];
EigenvectorCentrality[g, "In"]
{0.16238,0.136013,0.276307,0.23144,0.193859}

>> EigenvectorCentrality[g, "Out"]
{0.136013,0.16238,0.193859,0.23144,0.276307}

>> g = Graph[{a <-> b, b <-> c, c <-> d, d <-> e, e <-> c, e <-> a}];
EigenvectorCentrality[g]
{0.162435,0.162435,0.240597,0.193937,0.240597}

>> g = Graph[{a <-> b, b <-> c, a <-> c, d <-> e, e <-> f, f <-> d, e
<-> d}]; EigenvectorCentrality[g]
{0.166667,0.166667,0.166667,0.183013,0.183013,0.133975}

>> g = Graph[{a -> b, b -> c, c -> d, b -> e, a -> e, c -> a}];
EigenvectorCentrality[g]
{0.333333,0.333333,0.333333,0.,0.}
```

HITSCentrality

NetworkX, WMA

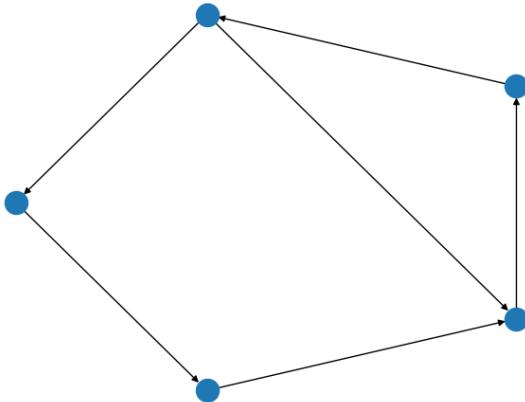
```
HITS Centrality[g]
    gives a list of authority and hub centralities for the vertices in the graph g.
```

KatzCentrality

Katz Centrality (NetworkX, WMA)

```
KatzCentrality[g, alpha]
    gives a list of Katz centralities for the vertices in the graph g and weight alpha.
KatzCentrality[g, alpha, beta]
    gives a list of Katz centralities for the vertices in the graph g and weight alpha and initial
    centralities beta.
```

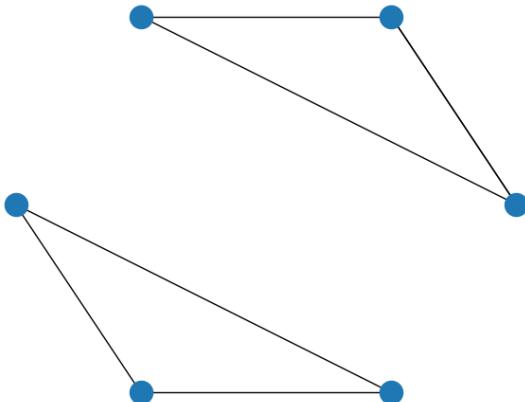
```
>> g = Graph[{a -> b, b -> c, c -> d, d -> e, e -> c, e -> a}]
```



```
>> KatzCentrality[g, 0.2]
```

```
{1.25202, 1.2504, 1.5021, 1.30042, 1.26008}
```

```
>> g = Graph[{a <-> b, b <-> c, a <-> c, d <-> e, e <-> f, f <-> d, e <-> d}]
```



```
>> KatzCentrality[g, 0.1]
```

```
{1.25, 1.25, 1.25, 1.41026, 1.41026, 1.28205}
```

PageRankCentrality

Pagerank Centrality (NetworkX, WMA)

```
PageRankCentrality[g, alpha]
```

gives a list of page rank centralities for the vertices in the graph g and weight α .

```
PageRankCentrality[g, alpha, beta]
```

gives a list of page rank centralities for the vertices in the graph g and weight α and initial centralities β .

```
>> g = Graph[{a -> d, b -> c, d -> c, d -> a, e -> c, d -> c}];  
PageRankCentrality[g, 0.2]
```

```
{0.184502, 0.207565, 0.170664, 0.266605, 0.170664}
```

Core routines for working with Graphs.

Core routines for working with Graphs.

AdjacencyList

Adjacency list (NetworkX, WMA)

```
AdjacencyList[graph, v]
    gives a list of vertices adjacent to v in a Graph or a list of edges g.
AdjacencyList[graph, pattern]
    gives a list of vertices adjacent to vertices matching pattern.
```

```
>> AdjacencyList[{1 -> 2, 2 -> 3}, 3]
{2}

>> AdjacencyList[{1 -> 2, 2 -> 3}, _?EvenQ]
{1,3}

>> AdjacencyList[{x -> 2, x -> 3, x -> 4, 2 -> 10, 2 -> 11, 4 -> 20, 4
-> 21, 10 -> 100}, 10, 2]
{2,11,100,x}
```

DirectedEdge

Edge of a Directed graph (NetworkX, WMA)

```
DirectedEdge[u, v]
    create a directed edge from u to v.
```

EdgeConnectivity

Edge connectivity (NetworkX, WMA)

```
EdgeConnectivity[g]
    gives the edge connectivity of the graph g.
```

```
>> EdgeConnectivity[{1 <-> 2, 2 <-> 3}]
1

>> EdgeConnectivity[{1 -> 2, 2 -> 3}]
0

>> EdgeConnectivity[{1 -> 2, 2 -> 3, 3 -> 1}]
1

>> EdgeConnectivity[{1 <-> 2, 2 <-> 3, 1 <-> 3}]
2

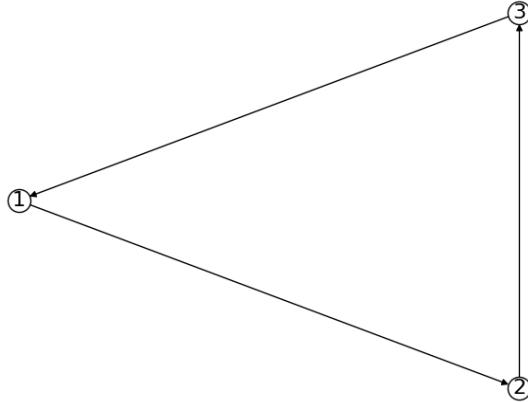
>> EdgeConnectivity[{1 <-> 2, 3 <-> 4}]
0
```

EdgeDelete

Delete an Edge (WMA)

```
EdgeDelete[g, edge]  
remove the edge edge.
```

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 -> 1}, VertexLabels->True]
```



```
>> EdgeList[EdgeDelete[g, 2 -> 3]]  
{\{1,2\},\{3,1\}}
```

EdgeIndex

WMA link

```
EdgeIndex[graph, edge]  
gives the position of the edge in the list of edges associated to the graph.
```

EdgeList

WMA link

```
EdgeList[g]  
gives the list of edges that defines g
```

EdgeRules

WMA link

```
EdgeRules[g]  
gives the list of edge rules for the graph g.
```

FindShortestPath

Shortest path problem (NetworkX, WMA)

```
FindShortestPath[g, src, tgt]
```

List the vertices in the shortest path connecting the source *src* with the target *tgt* in the graph *g*.

```
>> FindShortestPath[{1 <-> 2, 2 <-> 3, 3 <-> 4, 2 <-> 4, 4 -> 5}, 1, 5]
{1,2,4,5}

>> FindShortestPath[{1 <-> 2, 2 <-> 3, 3 <-> 4, 4 -> 2, 4 -> 5}, 1, 5]
{1,2,3,4,5}

>> FindShortestPath[{1 <-> 2, 2 <-> 3, 4 -> 3, 4 -> 2, 4 -> 5}, 1, 5]
{}

>> g = Graph[{1 -> 2, 2 -> 3, 1 -> 3}, EdgeWeight -> {0.5, a, 3}];
```

FindVertexCut

Minimum cut (NetworkX, WMA)

```
FindVertexCut [g]
```

finds a set of vertices of minimum cardinality that, if removed, renders *g* disconnected.

```
FindVertexCut [g, s, t]
```

finds a vertex cut that disconnects all paths from *s* to *t*.

```
>> g = Graph[{1 -> 2, 2 -> 3}]; FindVertexCut[g]
{}

>> g = Graph[{1 <-> 2, 2 <-> 3}]; FindVertexCut[g]
{2}

>> g = Graph[{1 <-> x, x <-> 2, 1 <-> y, y <-> 2, x <-> y}];
FindVertexCut[g]
{x,y}
```

Graph

Graph (WMA)

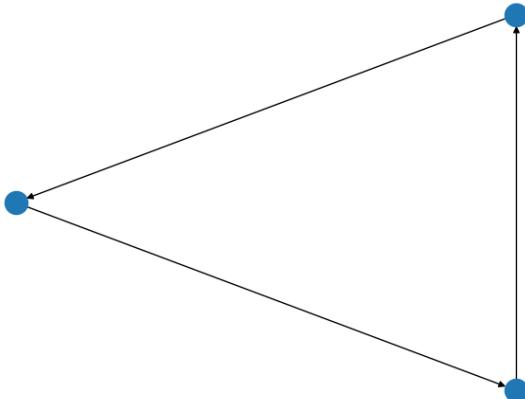
```
Graph[{$e1, $e2, ...}]
```

returns a graph with edges *e_j*.

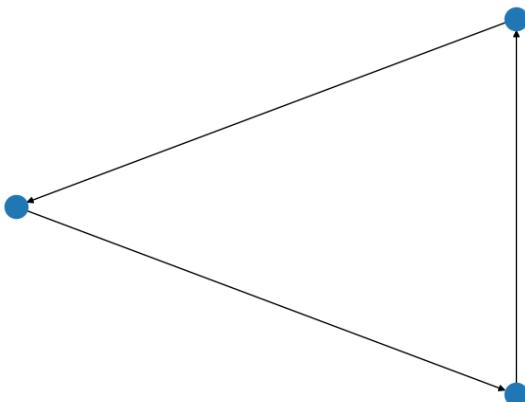
```
Graph[{v1, v2, ...}, {$e1, $e2, ...}]
```

returns a graph with vertices *v_i* and edges *e_j*.

```
>> Graph[{1->2, 2->3, 3->1}]
```



```
#» Graph[{1->2, 2->3, 3->1}, EdgeStyle -> {Red, Blue, Green}] # = -Graph-
>> Graph[{1->2, Property[2->3, EdgeStyle -> Thick], 3->1}]
```



```
# » Graph[{1->2, 2->3, 3->1}, VertexStyle -> {1 -> Green, 3 -> Blue}] # = -Graph-
>> Graph[x]
Graph[x]
>> Graph[{1}]
Graph [{1}]
>> Graph[{{1 -> 2}}]
Graph [{ {1-> 2}}]
```

HighlightGraph

WMA link

```
HighlightGraph[graph, what]
highlight in graph the elements enumerated in what by adding style marks.
```

Property

WMA link

```
Property[item, {name, val}]  
associate a property called name with value val to item.
```

PropertyValue

WMA link

```
PropertyValue[{obj, item}, name]  
gives the value of a property associated with the name name for item in the object obj.
```

```
>> g = Graph[{a <-> b, Property[b <-> c, SomeKey -> 123}];  
>> PropertyValue[{g, b <-> c}, SomeKey]  
123  
>> PropertyValue[{g, b <-> c}, SomeUnknownKey]  
$Failed
```

UndirectedEdge

WMA link

```
UndirectedEdge[u, v]  
create an undirected edge between u and v.
```

```
>> a <-> b  
UndirectedEdge[a, b]  
>> (a <-> b)<-> c  
UndirectedEdge[UndirectedEdge[a, b], c]  
>> a <-> (b <-> c)  
UndirectedEdge[a, UndirectedEdge[b, c]]
```

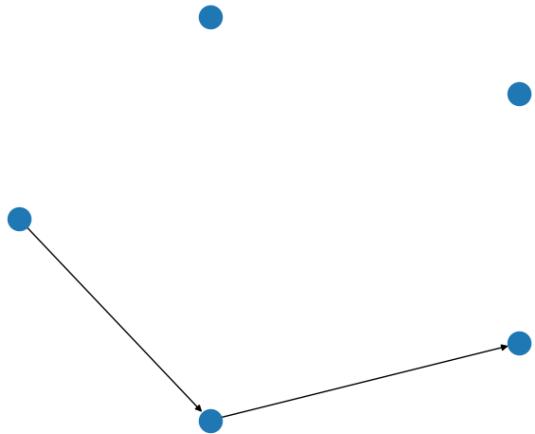
VertexAdd

WMA link

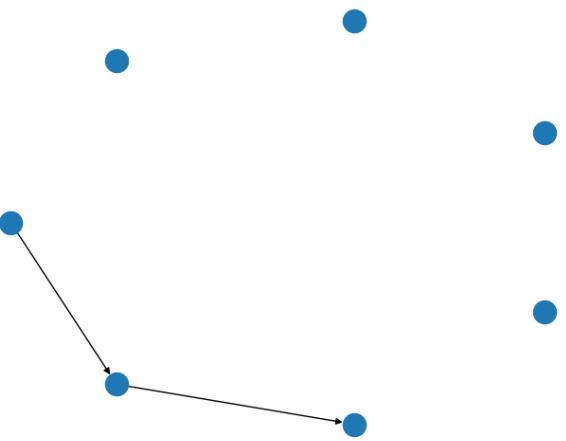
```
VertexAdd[g, ver]  
create a new graph from g, by adding the vertex ver.
```

```
>> g1 = Graph[{1 -> 2, 2 -> 3}];
```

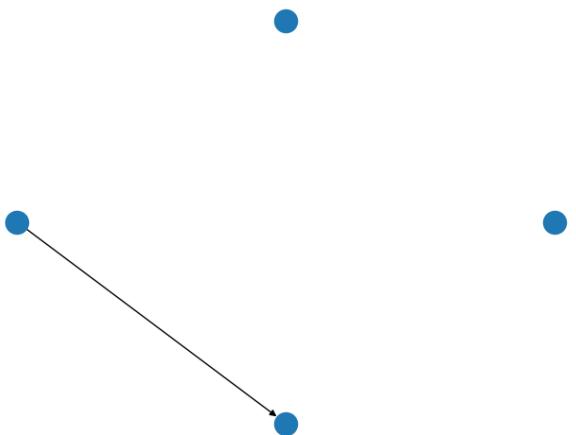
```
>> g2 = VertexAdd[g1, 4]
```



```
>> g3 = VertexAdd[g2, {5, 10}]
```



```
>> VertexAdd[{a -> b}, c]
```



VertexConnectivity

WMA link

```
VertexConnectivity[g]
gives the vertex connectivity of the graph g.
```

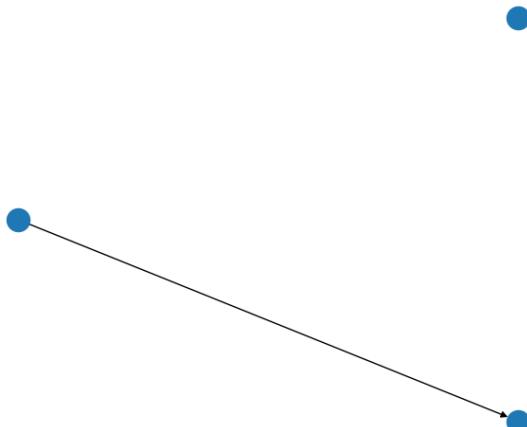
```
>> VertexConnectivity[{1 <-> 2, 2 <-> 3}]
1
>> VertexConnectivity[{1 -> 2, 2 -> 3}]
0
>> VertexConnectivity[{1 -> 2, 2 -> 3, 3 -> 1}]
1
>> VertexConnectivity[{1 <-> 2, 2 <-> 3, 1 <-> 3}]
2
>> VertexConnectivity[{1 <-> 2, 3 <-> 4}]
0
```

VertexDelete

WMA link

```
VertexDelete[g, vert]
remove the vertex vert and their associated edges.
```

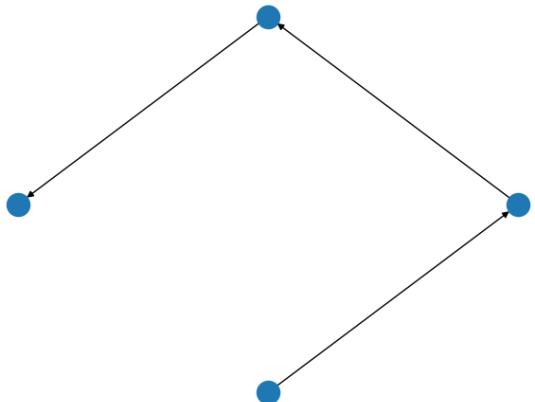
```
>> g1 = Graph[{1 -> 2, 2 -> 3, 3 -> 4}];
>> VertexDelete[g1, 3]
```



```
>> VertexDelete[{a -> b, b -> c, c -> d, d -> a}, {a, c}]
```



```
>> VertexDelete[{1 -> 2, 2 -> 3, 3 -> 4, 4 -> 6, 6 -> 8, 8 -> 2}, _?OddQ ]
```



VertexIndex

WMA link

```
VertexIndex[g, v]
```

gives the integer index of the vertex v in the graph g.

```
>> a=.;  
>> VertexIndex[{c <-> d, d <-> a}, a]  
3
```

VertexList

WMA link

```
VertexList [edgelist]
```

list the vertices from a list of directed edges.

```

>> a=.;  

>> VertexList[{1 -> 2, 2 -> 3}]  

{1,2,3}  

>> VertexList[{a -> c, c -> b}]  

{a,c,b}  

>> VertexList[{a -> c, 5 -> b}, _Integer -> 10]  

{10}

```

Curated Graphs

Curated Graph

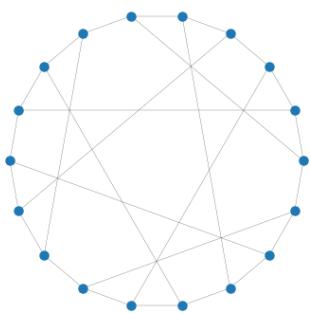
GraphData

WMA link

`GraphData[name]`

Returns a graph with the specified name.

```
>> GraphData["PappusGraph"]
```



Graph Components and Connectivity

Graph Components and Connectivity

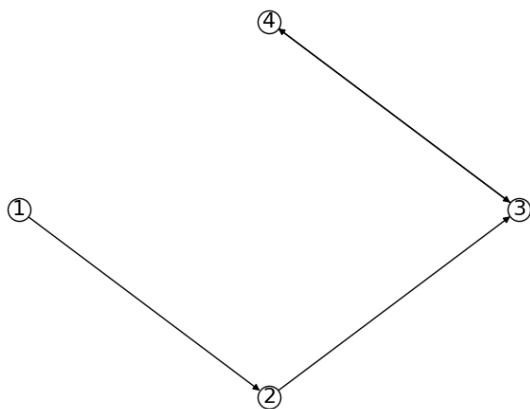
ConnectedComponents

Strongly connected components (NetworkX, WMA)

`ConnectedComponents[g]`

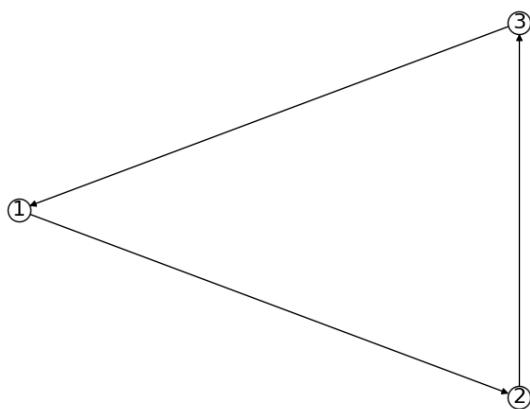
gives the connected components of the graph g .

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 <-> 4}, VertexLabels->True]
```



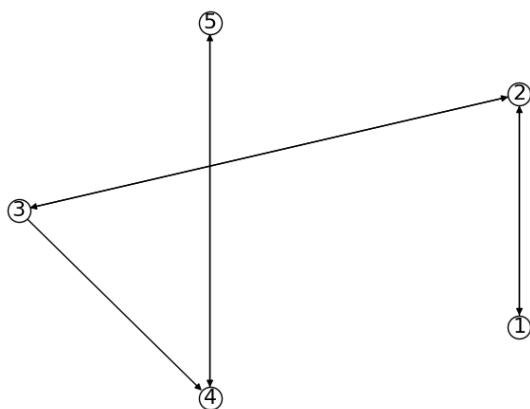
```
>> ConnectedComponents[g]
```

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 -> 1}, VertexLabels->True]
```



```
>> ConnectedComponents[g]
```

```
>> g = Graph[{1 <-> 2, 2 <-> 3, 3 -> 4, 4 <-> 5}, VertexLabels->True]
```



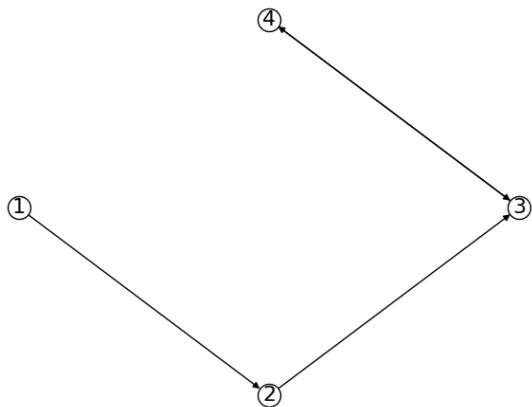
```
>> ConnectedComponents[g]
```

WeaklyConnectedComponents

Weak components (NetworkX, WMA)

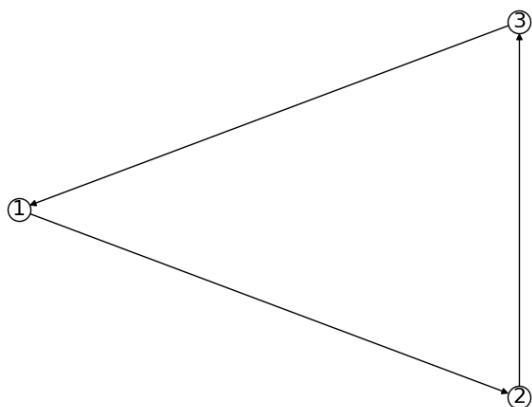
```
WeaklyConnectedComponents[g]
gives the weakly connected components of the graph g.
```

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 <-> 4}, VertexLabels->True]
```



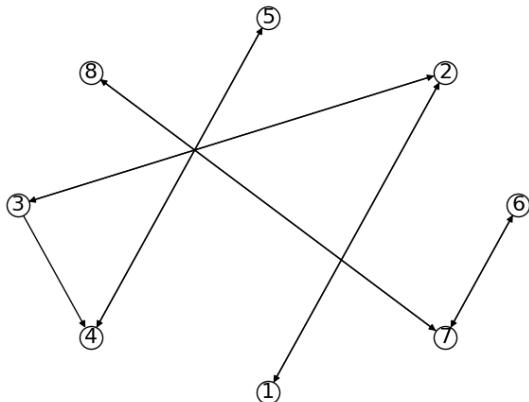
```
>> WeaklyConnectedComponents[g]
```

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 -> 1}, VertexLabels->True]
```



```
>> WeaklyConnectedComponents[g]
```

```
>> g = Graph[{1 <-> 2, 2 <-> 3, 3 -> 4, 4 <-> 5, 6 <-> 7, 7 <-> 8},
  VertexLabels->True]
```



```
>> WeaklyConnectedComponents[g]
```

Graph Measures and Metrics

Graph Measures and Metrics

Measures include basic measures, such as the number of vertices and edges, connectivity, degree measures, centrality, and so on.

EdgeCount

NetworkX, WMA

EdgeCount [<i>g</i>]	returns a count of the number of edges in graph <i>g</i> .
EdgeCount [<i>g</i> , <i>patt</i>]	returns the number of edges that match the pattern <i>patt</i> .
EdgeCount [{ <i>v</i> -> <i>w</i> }, ...], ...]	uses rules <i>v</i> -> <i>w</i> to specify the graph <i>g</i> .

```
>> EdgeCount[{1 -> 2, 2 -> 3}]
2
```

GraphDistance

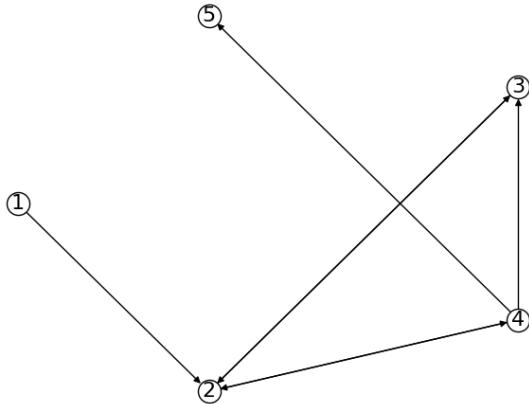
NetworkX, WMA

GraphDistance [<i>g</i> , <i>s</i> , <i>t</i>]	returns the distance from source vertex <i>s</i> to target vertex <i>t</i> in the graph <i>g</i> .
---	--

GraphDistance [<i>g</i> , <i>s</i>]	returns the distance from source vertex <i>s</i> to all vertices in the graph <i>g</i> .
--	--

```
GraphDistance[{v->w, ...}, ...]
use rules v->w to specify the graph g.
```

```
>> g = Graph[{1 -> 2, 2 <-> 3, 4 -> 3, 2 <-> 4, 4 -> 5}, VertexLabels -> True]
```



```
>> GraphDistance[g, 1, 5]
3
>> GraphDistance[g, 4, 2]
1
>> GraphDistance[g, 5, 4]
∞
>> GraphDistance[g, 5]
{∞,∞,∞,∞,0}
>> GraphDistance[g, 3]
{∞,1,2,0,3}
>> GraphDistance[g, 4]
{∞,1,0,1,1}
```

VertexCount

NetworkX, WMA

```
VertexCount[g]
returns a count of the number of vertices in graph g.
VertexCount[g, patt]
returns the number of vertices that match the pattern patt.
VertexCount[{v->$w}, ...], ...
uses rules v->w to specify the graph g.
```

```
>> VertexCount[{1 -> 2, 2 -> 3}]
3
```

```
>> VertexCount[{1 -> x, x -> 3}, _Integer]
2
```

VertexDegree

NetworkX, WMA

```
VertexDegree[g]
    returns a list of the degrees of each of the vertices in graph g.
EdgeCount[g, patt]
    returns the number of edges that match the pattern patt.
EdgeCount[{v->$w}, ...], ...
    uses rules v->w to specify the graph g.
```

```
>> VertexDegree[{1 <-> 2, 2 <-> 3, 2 <-> 4}]
{1,3,1,1}
```

Graph Operations and Modifications

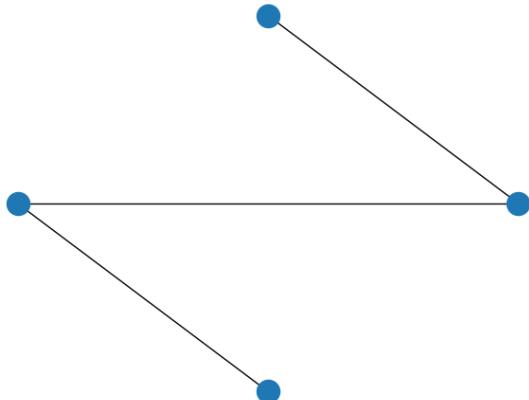
Graph Operations and Modification

FindSpanningTree

Spanning Tree (NetworkX, WMA)

```
FindSpanningTree[g]
    finds a spanning tree of the graph g.
```

```
>> FindSpanningTree[CycleGraph[4]]
```



Graph Properties and Measurements

Graph Properties and Measurement

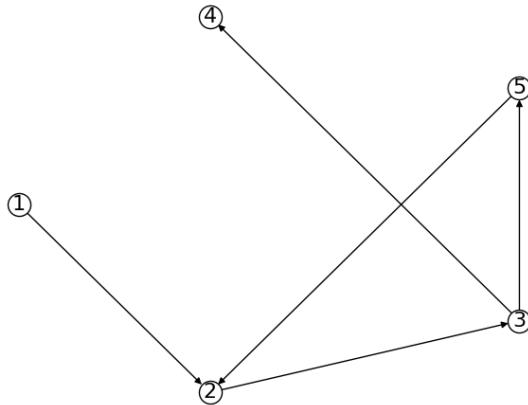
AcyclicGraphQ

Acyclic graph test (NetworkX, WMA)

```
AcyclicGraphQ[graph]
  check if graph is an acyclic graph.
```

Create a directed graph with a cycle in it:

```
>> g = Graph[{1 -> 2, 2 -> 3, 5 -> 2, 3 -> 4, 3 -> 5}, VertexLabels -> True]
```



```
>> AcyclicGraphQ[g]
```

Remove a cycle edge:

```
>> g = EdgeDelete[g, 5 -> 2]; EdgeList[g]
{{1,2},{2,3},{3,4},{3,5}}
>> AcyclicGraphQ[g]
```

ConnectedGraphQ

Connected graph test (NetworkX, WMA)

```
ConnectedGraphQ[graph]
  check if graph is a connected graph.
```

```
>> g = Graph[{1 -> 2, 2 -> 3}]; ConnectedGraphQ[g]
False
>> g = Graph[{1 -> 2, 2 -> 3, 3 -> 1}]; ConnectedGraphQ[g]
True
>> g = Graph[{1 <-> 2, 2 <-> 3}]; ConnectedGraphQ[g]
True
>> g = Graph[{1 <-> 2, 2 <-> 3, 4 <-> 5}]; ConnectedGraphQ[g]
False
```

DirectedGraphQ

Directed graph test (NetworkX, WMA)

`DirectedGraphQ[graph]`

True if *graph* is a Graph and all the edges are directed.

```
>> g = Graph[{1 -> 2, 2 -> 3}]; DirectedGraphQ[g]
True
>> g = Graph[{1 -> 2, 2 <-> 3}]; DirectedGraphQ[g]
False
```

LoopFreeGraphQ

Loop-Free graph test (NetworkX, WMA)

`LoopFreeGraphQ[graph]`

True if *graph* is a Graph and the edges do not close any loop.

```
>> g = Graph[{1 -> 2, 2 -> 3}]; LoopFreeGraphQ[g]
True
>> g = Graph[{1 -> 2, 2 -> 3, 1 -> 1}]; LoopFreeGraphQ[g]
False
```

MixedGraphQ

Mixed Graph test (WMA)

`MixedGraphQ[graph]`

returns True if *graph* is a Graph with both directed and undirected edges, and False otherwise.

```
>> MixedGraphQ[Graph[{1 -> 2, 2 -> 3}]]
False
>> MixedGraphQ[Graph[{1 -> 2, 2 <-> 3}]]
True
>> MixedGraphQ[Graph[{}]]
False
>> MixedGraphQ["abc"]
False
```

MultigraphQ

Multigraph test (NetworkX, WMA)

`MultigraphQ[graph]`

True if *graph* is a Graph and there vertices connected by more than one edge.

```

>> g = Graph[{1 -> 2, 2 -> 3}]; MultigraphQ[g]
False

>> g = Graph[{1 -> 2, 2 -> 3, 1 -> 2}]; MultigraphQ[g]
True

```

PathGraphQ

Path graph test (WMA)

```

LoopFreeGraphQ[graph]
True if graph is a Graph and it becomes disconnected by removing a single edge.

```

```

>> PathGraphQ[Graph[{1 -> 2, 2 -> 3}]]
True

>> PathGraphQ[Graph[{1 -> 2, 2 <-> 3}]]
False

>> PathGraphQ[Graph[{1 -> 2, 3 -> 2}]]
False

>> PathGraphQ[Graph[{1 -> 2, 2 -> 3, 2 -> 4}]]
False

>> PathGraphQ[Graph[{1 -> 2, 3 -> 2, 2 -> 4}]]
False

>> PathGraphQ[Graph[{1 -> 2, 2 -> 3, 2 -> 3}]]
False

```

PlanarGraphQ

Planar Graph test (NetworkX, WMA)

```

PlanarGraphQ[g]
Returns True if g is a planar graph and False otherwise.

```

```

>> PlanarGraphQ[CycleGraph[4]]
True

>> PlanarGraphQ[CompleteGraph[5]]
False

>> PlanarGraphQ["abc"]
Expected a graph at position 1 in PlanarGraphQ[abc].
False

```

SimpleGraphQ

Simple (not multigraph)graph test (WMA)

```
LoopFreeGraphQ[graph]
```

True if *graph* is a Graph, loop-free and each pair of vertices are connected at most by a single edge.

```
>> g = Graph[{1 -> 2, 2 -> 3, 3 <-> 4}]; SimpleGraphQ[g]
>> g = Graph[{1 -> 2, 2 -> 3, 1 -> 1}]; SimpleGraphQ[g]
>> g = Graph[{1 -> 2, 2 -> 3, 1 -> 2}]; SimpleGraphQ[g]
```

Parametric Graphs

Parametric Graph

BalancedTree

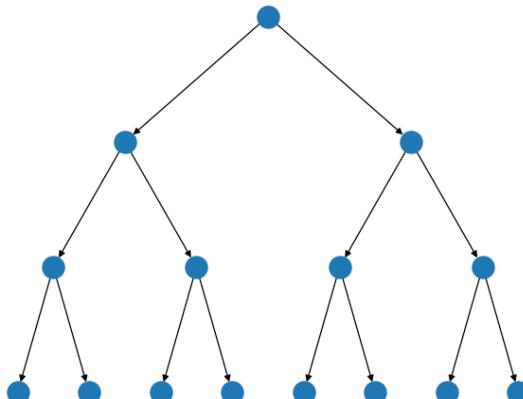
WMA

```
BalancedTree[r, h]
```

Returns the perfectly balanced r -ary tree of height h .

In this tree produced, all non-leaf nodes will have r children and the height of the path from root r to any leaf will be h .

```
>> BalancedTree[2, 3]
```



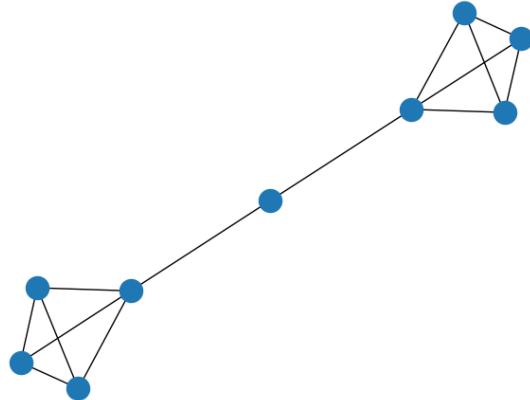
BarbellGraph

Barbell graph (NetworkX, Wolfram MathWorld)

```
BarbellGraph[m1, m2]
```

Barbell Graph: two complete graphs connected by a path.

```
>> BarbellGraph[4, 1]
```



BinomialTree

Binomial tree (NetworkX, WMA)

```
BinomialTree[n]
```

Returns the Binomial Tree of order n .

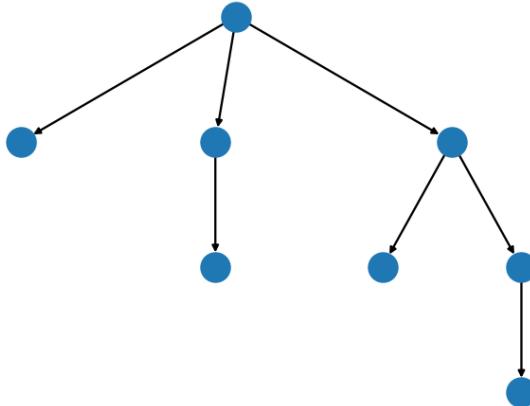
The binomial tree of order n with root R is defined as:

If $k=0$, $B[k] = B[0] = \{R\}$. i.e., the binomial tree of order zero consists of a single node, R .

If $k>0$, $B[k] = \{R, B[0], B[1] \dots B[k]\}$, i.e., the binomial tree of order $k>0$ comprises the root R , and k binomial subtrees, $B[0]$ to $B[k]$.

Binomial trees are the underlying data structure in Binomial heaps.

```
>> BinomialTree[3]
```



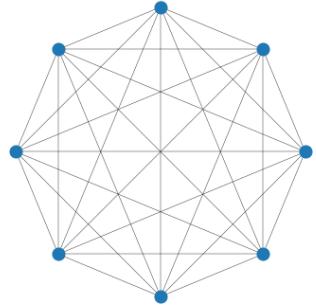
CompleteGraph

Complete Multipartite Graph (NetworkX, WMA)

```
CompleteGraph[n]
```

Returns the complete graph with n vertices, K_n .

```
>> CompleteGraph[8]
```



CompleteKaryTree

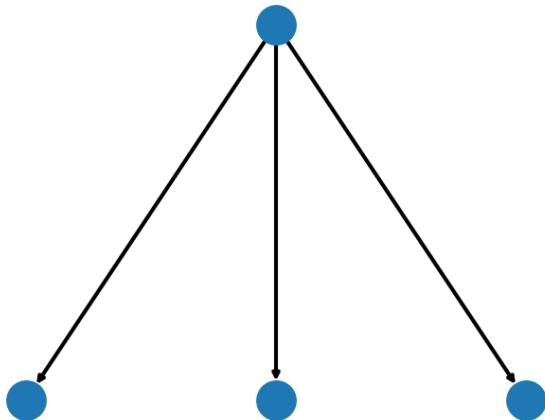
M-ary Tree (NetworkX, WMA)

```
CompleteKaryTree[n, k]
```

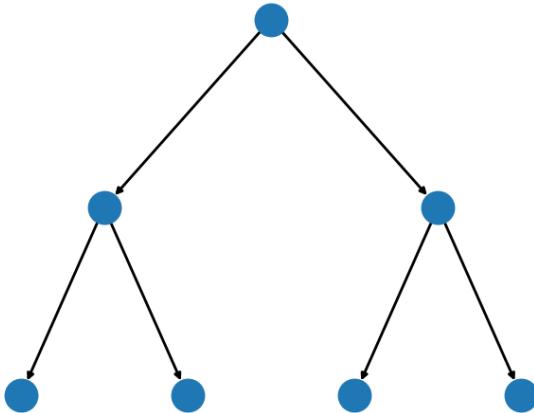
Creates a complete k -ary tree of n levels.

In the returned tree, with n nodes, the from root R to any leaf be k .

```
>> CompleteKaryTree[2, 3]
```



```
>> CompleteKaryTree[3]
```



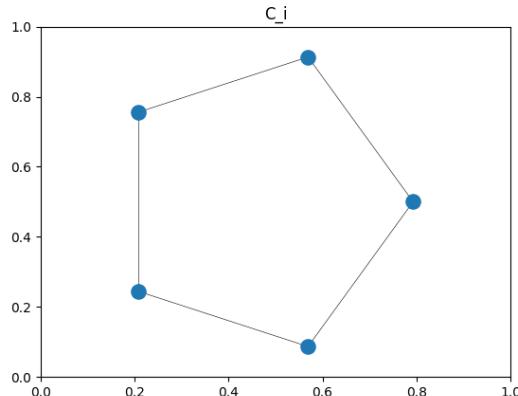
CycleGraph

Cycle Graph (WMA)

```
CycleGraph[n]
```

Returns the cycle graph with n vertices C_n .

```
>> CycleGraph[5, PlotLabel -> "C_i"]
```



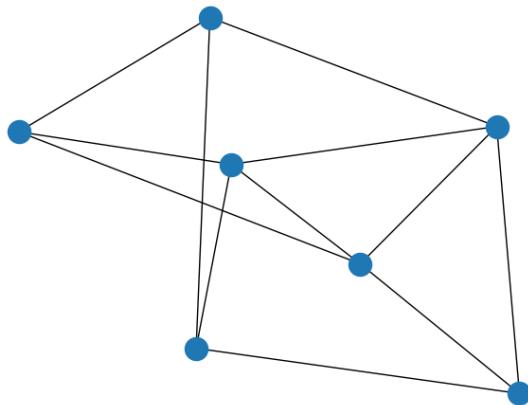
GraphAtlas

NetworkX

```
GraphAtlas[n]
```

Returns graph number i from the NetworkX's Graph Atlas. There are about 1200 of them and get large as i increases.

```
>> GraphAtlas[1000]
```



HknHararyGraph

NetworkX, WMA

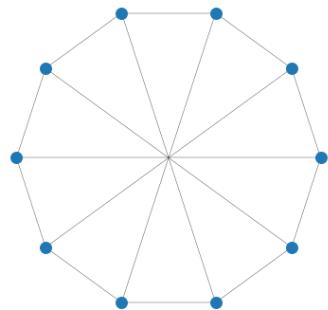
```
HknHararyGraph [k, n]
```

Returns the Harary graph with given node connectivity and node number.

This second generator gives the Harary graph that minimizes the number of edges in the graph with given node connectivity and number of nodes.

Harary, F. The Maximum Connectivity of a Graph. Proc. Nat. Acad. Sci. USA 48, 1142-1146, 1962.

```
>> HknHararyGraph[3, 10]
```



HmnHararyGraph

NetworkX, WMA

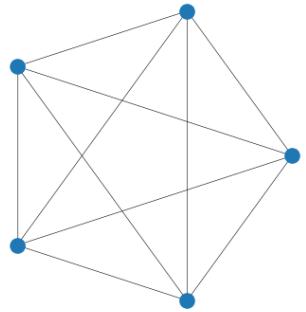
```
HmnHararyGraph [m, n]
```

Returns the Harary graph with given numbers of nodes and edges.

This generator gives the Harary graph that maximizes the node connectivity with given number of nodes and given number of edges.

Harary, F. The Maximum Connectivity of a Graph. Proc. Nat. Acad. Sci. USA 48, 1142-1146, 1962.

```
>> HmnHararyGraph[5, 10]
```



KaryTree

M-ary Tree

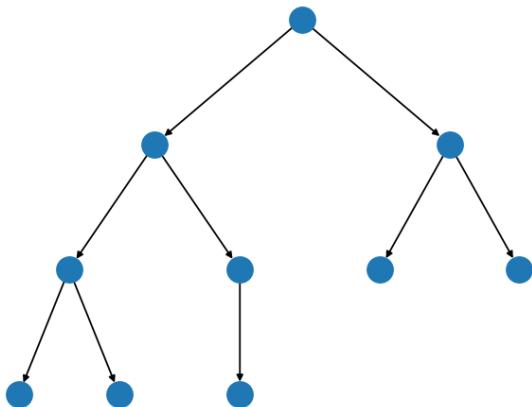
```
KaryTree[r, n]
```

Creates binary tree of n vertices.

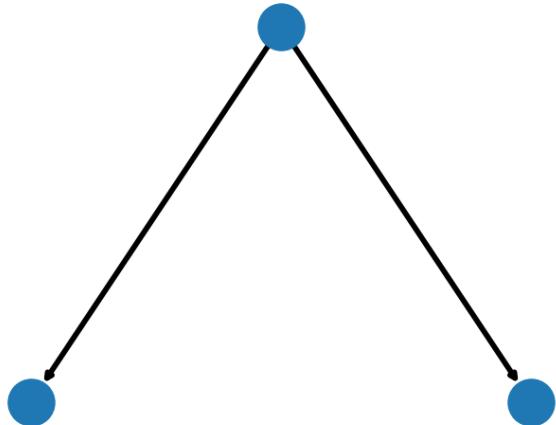
```
KaryTree[n, k]
```

Creates k -ary tree with n vertices.

```
>> KaryTree[10]
```



```
>> KaryTree[3, 10]
```



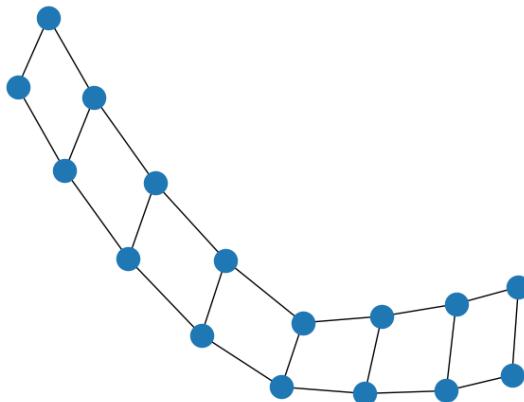
LadderGraph

Ladder graph (NetworkX)

```
LadderGraph[n]
```

Returns the Ladder graph of length n .

```
>> LadderGraph[8]
```



PathGraph

Path graph (WMA)

```
PathGraph[{v_1, v_2, ...}]
```

Returns a Graph with a path with vertices v_i and edges between v_i and v_{i+1} .

```
>> PathGraph[{1, 2, 3}]
```



RandomTree

NetworkX, WMA

```
RandomTree[n]
```

Returns a uniformly random tree on n nodes.

```
>> RandomTree[3]
```



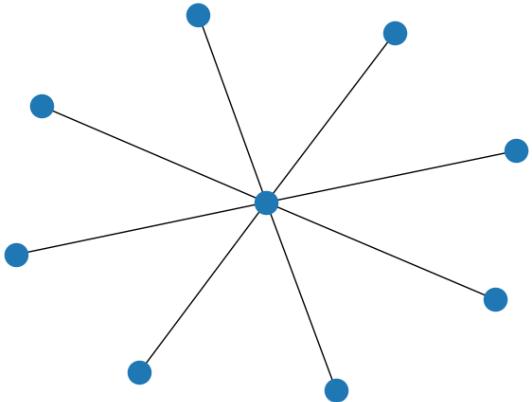
StarGraph

Star graph(NetworkX, WMA)

```
StarGraph[n]
```

Returns a star graph with n vertices.

```
>> StarGraph[8]
```



Random Graphs

Random Graph

RandomGraph

WMA link

```
RandomGraph[{n, m}]
```

Returns a pseudorandom graph with n vertices and m edges.

```
RandomGraph[{n, m}, k]
```

Returns list of k RandomGraph[{ n, m }].

Trees

Tree

TreeGraph

Tree Graph (WMA)

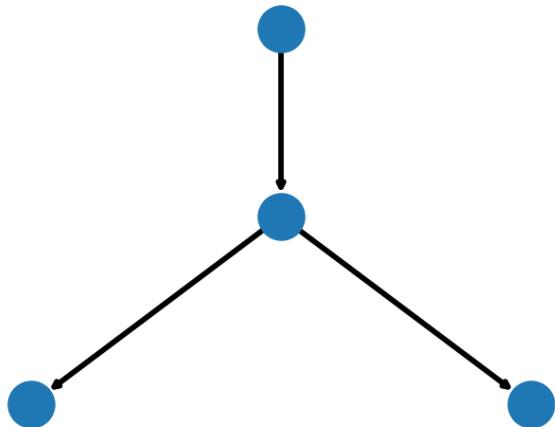
```
TreeGraph[edges]
```

Build a Tree-like graph from the list of edges $edges$.

```
TreeGraph[vert, edges]
```

build a Tree-like graph from the list of vertices $vert$ and edges $edges$.

```
>> TreeGraph[{1->2, 2->3, 2->4}]
```



If the *edges* does not match with a tree-like pattern, the evaluation fails:

```
>> TreeGraph[{1->2, 2->3, 3->1}]\nGraph is not a tree.\nTreeGraph [{1->2, 2->3, 3->1}]
```

TreeGraphQ

Tree Graph (WMA)

TreeGraphQ[g]
returns *True* if the graph *g* is a tree and *False* otherwise.

```
>> TreeGraphQ[StarGraph[3]]\nTrue\n>> TreeGraphQ[CompleteGraph[2]]\nTrue\n>> TreeGraphQ[CompleteGraph[3]]\nFalse
```

2. Natural Language Processing

Mathics3 Module module provides functions and variables to work with expressions in natural language, using the Python libraries:

- spacy for parsing natural languages</url>
- nltk for functions using WordNet-related builtins
- pyenchant and pycountry for language identification

Examples:

```
>> LoadModule["pymathics.natlang"]
pymathics.natlang

>> Pluralize["try"]
tries

>> LanguageIdentify["eins zwei drei"]
German

>> WordFrequency["Apple Tree and apple", "apple", IgnoreCase -> True]
0.5

>> TextCases["I was in London last year.", "Pronoun"]
{I}

>> DeleteStopwords["There was an Old Man of Apulia, whose conduct was
very peculiar"]
Old Man Apulia, conduct peculiar
```

Contents

Language Translation	637	WordStem	640	Text Normalization	642
LanguageIdentify	638	Text Analysis	640	DeleteStopwords	642
Linguistic Data	638	Containing	640	TextCases	643
DictionaryLookup	638	SpellingCorrec-		TextPosition	643
DictionaryWordQ	638	tionList	641	TextSentences	643
RandomWord	639	WordCount	641	TextStructure	643
WordData	639	WordFrequency	641	TextWords	644
WordDefinition	639	WordSimilarity	642	Word manipulation	644
WordList	640	WordStem	642	Pluralize	644

Language Translation

Language Translation

LanguageIdentify

WMA link

```
LanguageIdentify[text]  
    returns the name of the language used in text.
```

```
>> LanguageIdentify["eins zwei drei"]  
German
```

Linguistic Data

Linguistic Data

See the corresponding WMA guide.

DictionaryLookup

WMA link

```
DictionaryLookup[word]  
    lookup words that match the given word or pattern.  
DictionaryLookup[word, n]  
    lookup first n words that match the given word or pattern.
```

```
>> DictionaryLookup["baker" ~~__]  
{baker, baker's dozen, baker's eczema, baker's yeast, bakersfield, bakery}  
>> DictionaryLookup["baker" ~~__, 3]  
{baker, baker's dozen, baker's eczema}
```

DictionaryWordQ

WMA link

```
DictionaryWordQ[word]  
    returns True if word is a word usually found in dictionaries, and False otherwise.
```

```
>> DictionaryWordQ["couch"]  
True  
>> DictionaryWordQ["meep-meep"]  
False
```

RandomWord

WMA link

```
<dl><dt>RandomWord[] <dd>returns a random word.  
<dt>RandomWord[type] <dd>returns a random word of the given type, e.g. of type "Noun" or "Adverb".  
<dt>RandomWord[type, n] <dd>returns n random words of the given type.  
>> RandomWord["Noun"]  
dacrymycetaceae
```

```
>> RandomWord["Noun", 3]
{winter crookneck squash, benzyl radical, emile durkheim}
</dl>
```

WordData

WMA link

```
WordData[word]
    returns a list of possible senses of a word.
WordData[word, property]
    returns detailed information about a word regarding property, e.g. "Definitions" or "Examples".
```

The following are valid properties:

- Definitions, Examples
- InflectedForms
- Synonyms, Antonyms
- BroaderTerms, NarrowerTerms
- WholeTerms, PartTerms, MaterialTerms
- EntailedTerms, CausesTerms
- UsageField
- WordNetID
- Lookup

```
>> WordData["riverside", "Definitions"]
{{riverside, Noun, Bank}} -> the bank of a river

>> WordData[{"fish", "Verb", "Angle"}, "Examples"]
{{fish, Verb, Angle}} -> {fish for compliments}
```

WordDefinition

WMA link

```
WordDefinition[word]
    returns a definition of word or Missing["Available"] if word is not known.
```

```
>> WordDefinition["gram"]
{a metric unit of weight equal to one thousandth of a kilogram}
```

WordList

WMA link

```
WordList[]
    returns a list of common words.
WordList[type]
    returns a list of common words of type type.
```

Evaluate the average length over all the words in the dictionary:

```
>> N[Mean[StringLength /@ WordList[]], 3]
11.6
```

Now, restricted to adjectives:

```
>> N[Mean[StringLength /@ WordList["Adjective"]], 2]
9.3
```

WordStem

WMA link

`WordStem[word]`

returns a stemmed form of *word*, thereby reducing an inflected form to its root.

`WordStem[{word1, word2, ...}]`

returns a stemmed form for list of *word*, thereby reducing an inflected form to its root.

```
>> WordStem["towers"]
tower
>> WordStem[{ "heroes", "roses", "knights", "queens"}]
{hero,rose,knight,queen}
```

Text Analysis

Text Analysis

See the corresponding WMA guide.

Containing

WMA link

`Containing[outer, inner]`

represents an object of the type *outer* containing objects of type *inner*.

Containing

can be used as the second parameter in `TextCases` and `TextPosition`.

Supported *outer* strings are in {"Word", "Sentence", "Paragraph", "Line", "URL", "EmailAddress"}.

Supported *inner* strings are in {"Person", "Company", "Quantity", "Number", "CurrencyAmount", "Country", "City"}.

The implementation of this symbol is based on 'spacy'.

```
>> TextCases["This is a pencil. This is another pencil from England.", 
    Containing["Sentence", "Country"]]
{This is another pencil from England.}

>> TextPosition["This is a pencil. This is another pencil from England
.", Containing["Sentence", "Country"]]
{{19,54}}
```

SpellingCorrectionList

WMA link

```
SpellingCorrectionList[word]  
returns a list of suggestions for spelling corrected versions of word.
```

Results may differ depending on which dictionaries can be found by `enchant`.

```
>> SpellingCorrectionList["hipopotamus"]  
{hippopotamus, hippopotamus's, hippopotamuses}
```

WordCount

WMA link

```
WordCount[string]  
returns the number of words in string.
```

```
>> WordCount["A long time ago"]  
4
```

WordFrequency

WMA link

```
WordFrequency[text, word]  
returns the relative frequency of word in text.
```

word may also specify multiple words using *a* | *b* | ...

```
>> text = "I have a dairy cow, it's not just any cow. She gives me  
milkshake, oh what a salty cow. She is the best cow in the county.";  
  
>> WordFrequency[text, "a" | "the"]  
0.121212  
  
>> WordFrequency["Apple Tree", "apple", IgnoreCase -> True]  
0.5
```

WordSimilarity

WMA link

```
WordSimilarity[text1, text2]  
returns a real-valued measure of semantic similarity of two texts or words.  
WordSimilarity[{text1, i1}, {text2, j1}]  
returns a measure of similarity of two words within two texts.  
WordSimilarity[{text1, {i1, i2, ...}}, {text2, {j1, j2, ...}}]  
returns a measure of similarity of multiple words within two texts.
```

```
>> NumberForm[WordSimilarity["car", "train"], 3]  
0.439  
  
>> NumberForm[WordSimilarity["car", "hedgehog"], 3]  
0.195
```

```
>> NumberForm[WordSimilarity[{"An ocean full of water.", {2, 2}}, {"A desert full of sand.", {2, 5}}], 3]
{0.505,0.481}
```

WordStem

WMA link

`WordStem[word]`

returns a stemmed form of *word*, thereby reducing an inflected form to its root.

`WordStem[{word1, word2, ...}]`

returns a stemmed form for list of *word*, thereby reducing an inflected form to its root.

```
>> WordStem["towers"]
tower

>> WordStem[{"heroes", "roses", "knights", "queens"}]
{hero,rose,knight,queen}
```

Text Normalization

Text Normalization

See the corresponding WMA guide.

This module uses spacy as a backend.

DeleteStopwords

Delete stop words(WMA)

`DeleteStopwords[list]`

returns the words in *list* without stopwords.

`DeleteStopwords[string]`

returns *string* without stopwords.

```
>> DeleteStopwords[{"Somewhere", "over", "the", "rainbow"}]
{Somewhere,over,the,rainbow}

>> DeleteStopwords["There was an Old Man of Apulia, whose conduct was
very peculiar"]
Old Man Apulia,conduct peculiar
```

TextCases

WMA link

`TextCases[text, form]`

returns all elements of type *form* in *text* in order of their appearance.

```
>> TextCases["I was in London last year.", "Pronoun"]
{I}
```

```

>> TextCases["I was in London last year.", "City"]
{London}

>> TextCases["Anne, Peter and Mr Johnes say hello.", "Person",
3][[2;;3]]
{Peter,Johnes}

```

TextPosition

WMA link

```
TextPosition[text, form]
returns the positions of elements of type form in text in order of their appearance.
```

```

>> TextPosition["Liverpool and London are two English cities.", "City"]
{{1,9},{15,20}}

```

TextSentences

Sentences in a text (WMA)

```
TextSentences[string]
returns the sentences in string.
TextSentences[string, n]
returns the first n sentences in string
```

```

>> TextSentences["Night and day. Day and night."]
{Night and day.,Day and night.}

>> TextSentences["Night and day. Day and night.", 1]
{Night and day.}

>> TextSentences["Mr. Jones met Mrs. Jones."]
{Mr. Jones met Mrs. Jones.}

```

TextStructure

WMA link

```
TextStructure[text, form]
returns the grammatical structure of text as form.
```

```

>> TextStructure["The cat sat on the mat.", "ConstituentString"]
{(Sentence, ((Verb Phrase, (Noun Phrase, (Determiner, The),
(Noun, cat)), (Verb, sat), (Prepositional Phrase, (Preposition, on),
(Noun Phrase, (Determiner, the), (Noun, mat)))), (Punctuation, .))))}

```

TextWords

WMA link

```
TextWords[string]
  returns the words in string.
TextWords[string, n]
  returns the first n words in string
```

```
>> TextWords["Hickory, dickory, dock! The mouse ran up the clock."]
{Hickory, dickory, dock, The, mouse, ran, up, the, clock}

>> TextWords["Bruder Jakob, Schläfst du noch?", 2]
{Bruder, Jakob}
```

Word manipulation

Word manipulation

This module uses pattern.en to change the form of a word.

Pluralize

WMA link

```
Pluralize[word]
  returns the plural form of word.
```

```
>> Pluralize["potato"]
potatoes
```

Part IV.

License

A. GNU General Public License

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see \href{http://www.gnu.org/licenses/}{Licenses}.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see Licences.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read Why you shouldn't use the Lesser GPL for your next library.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works. The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers and authors protection, the GPL clearly explains that there is no warranty for this free software. For both users and authors sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." Licensees and "recipients" may be individuals or organizations.

To “modify a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work means either the unmodified Program or a work based on the Program.

To “propagate a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work means the preferred form of the work for making modifications to it. “Object code means any non-source form of a work.

A “Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those

works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the

recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates

your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by

the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

0. Definitions.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." Licensees and "recipients" may be individuals or organizations.

To "modify a work" means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work" or a work "based on the earlier work."

A "covered work" means either the unmodified Program or a work based on the Program.

To “propagate a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work means the preferred form of the work for making modifications to it. “Object code means any non-source form of a work.

A “Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in

source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
 - b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
 - c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
 - d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
 - e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
 - f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
- All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation

prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this

License, to extend the patent license to downstream recipients. "Knowingly relying means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

1. Source Code.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." "Licensees" and "recipients" may be individuals or organizations.

To "modify a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work or a work "based on the earlier work.

A "covered work means either the unmodified Program or a work based on the Program.

To "propagate a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work means the preferred form of the work for making modifications to it. “Object code means any non-source form of a work.

A “Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product,” which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information for a User Product” means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
 - b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
 - c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
 - d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
 - e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
 - f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
- All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not per-

manently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your

recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

2. Basic Permissions.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." Licensees and "recipients" may be individuals or organizations.

To "modify a work" means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work" or a work "based on the earlier work."

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate a work" means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey a work" means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that

licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code for a work means the preferred form of the work for making modifications to it. "Object code means any non-source form of a work.

A "Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with

respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the works users, your or third parties legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product is either (1) a “consumer product, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered “further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to

receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits

the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EX-

PRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." "Licensees" and "recipients" may be individuals or organizations.

To "modify a work" means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work" or a work "based on the earlier work."

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate a work" means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey a work" means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work means the preferred form of the work for making modifications to it. “Object code means any non-source form of a work.

A “Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at

no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may

(if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent

license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

4. Conveying Verbatim Copies.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work or a work "based on the earlier work."

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major

Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or

- service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered “further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot

convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

5. Conveying Modified Source Versions.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." "Licensees" and "recipients" may be individuals or organizations.

To "modify a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work or a work "based on the earlier work.

A "covered work means either the unmodified Program or a work based on the Program.

To "propagate a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code for a work means the preferred form of the work for making modifications to it. "Object code means any non-source form of a work.

A "Standard Interface means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynami-

cally linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."

- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product,” which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial

commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
 - b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
 - c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
 - d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
 - e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
 - f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
- All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that

term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”. A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant such a patent license to a party” means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

6. Conveying Non-Source Forms.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

"Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as "you." Licensees and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version of the earlier work or a work "based on the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component," in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product,” which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information for a User Product” means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User

Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

7. Additional Terms.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

8. Termination.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

9. Acceptance Not Required for Having Copies.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

10. Automatic Licensing of Downstream Recipients.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

11. Patents.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

12. No Surrender of Others' Freedom.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

13. Use with the GNU Affero General Public License.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

14. Revised Versions of this License.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

15. Disclaimer of Warranty.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

16. Limitation of Liability.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices."
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

17. Interpretation of Sections 15 and 16.

0. Definitions.

"This License refers to version 3 of the GNU General Public License.

“Copyright also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program refers to any copyrightable work licensed under this License. Each licensee is addressed as “you.” Licensees and “recipients” may be individuals or organizations.

To “modify a work” means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version of the earlier work or a work “based on the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate a work” means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey a work” means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code for a work” means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component,” in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges

your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sub-licensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product," which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information for a User Product" means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains

the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant such a patent license to a party" means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

B. Included software and data

Included data

Mathics3 includes data from Wikipedia that is published under the Creative Commons Attribution-Sharealike 3.0 Unported License and the GNU Free Documentation License contributed by the respective authors that are listed on the websites specified in "data/elements.txt".

MathJax

Copyright © 2009-2010 Design Science, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Prototype

Copyright © 2005-2010 Sam Stephenson

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SciPy

Copyright © 2001, 2002 Enthought, Inc. All rights reserved.

Copyright © 2003-2019 SciPy Developers. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Enthought nor the names of the SciPy Developers may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Three.js

Copyright l' 2010-2020 Three.js authors.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

mpmath

Copyright (c) 2005-2018 Fredrik Johansson and mpmath contributors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

pymimemagic

Copyright (c) 2009, Xiaohai Lu All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendices

Index

\$Aborted, 475
\$Assumptions, 460
\$BaseDirectory, 197, 330
\$BoxForms, 458
\$ByteOrdering, 87
\$CharacterEncoding, 73
\$CharacterEncodings, 73
\$CommandLine, 239
\$Context, 516
\$ContextPath, 516
\$ContextPathStack, 516
\$ContextStack, 516
\$DateStringFormat, 114
\$Echo, 318
\$ExportFormats, 311, 343
\$ExtensionMappings, 311, 343
\$Failed, 475
\$FormatMappings, 311, 343
\$HistoryLength, 590
\$HomeDirectory, 197, 338
\$ImportFormats, 312, 345
\$InitialDirectory, 198, 338
\$Input, 186, 322
\$InputFileName, 186, 321
\$InstallationDirectory, 198, 338
\$Line, 591
\$Machine, 239
\$MachineEpsilon, 69
\$MachineName, 239
\$MachinePrecision, 70
\$MaxMachineNumber, 401
\$MaxPrecision, 70
\$MinMachineNumber, 401
\$MinPrecision, 70
\$ModuleNumber, 517
\$OperatingSystem, 198, 339
\$OutputForms, 211
\$Packages, 240
\$ParentProcessID, 240
\$Path, 198, 339
\$PathnameSeparator, 198, 339
\$Post, 591
\$Pre, 591
\$PrePrint, 592
\$PreRead, 592
\$PrintForms, 211
\$ProcessID, 240
\$ProcessorType, 240
\$PythonImplementation, 240
\$RandomState, 412
\$RootDirectory, 199, 340
\$ScriptCommandLine, 241
\$SyntaxHandler, 592
\$SystemCharacterEncoding, 76
\$SystemID, 241
\$SystemMemory, 241
\$SystemTimeZone, 114
\$SystemWordLength, 241
\$TemporaryDirectory, 199, 341
\$TimeZone, 114
\$TraceBuiltins, 594
\$TraceEvaluation, 594
\$UserBaseDirectory, 199, 342
\$UserName, 242
\$Version, 242
\$VersionNumber, 242

Abort, 498
Abs, 460
AbsoluteFileName, 199, 330
AbsoluteThickness, 142
AbsoluteTime, 115
AbsoluteTiming, 115
Accumulate, 50
Accuracy, 66
AcyclicGraphQ, 624
AddTo, 59
AdjacencyList, 610
AiryAi, 522
AiryAiPrime, 523
AiryAiZero, 523
AiryBi, 524
AiryBiPrime, 524
AiryBiZero, 525
Algebraic Transformations, 358
All, 490
AllTrue, 580
Alphabet, 72
Alternatives, 504
And, 580
AngerJ, 525

AnglePath, 413
 AngleVector, 484
 Angular Momentum, 236
 AnyTrue, 581
 Apart, 358
 Append, 435
 AppendTo, 435
 Apply, 40, 225
 Applying Functions to Lists, 225
 ApplyLevel, 176
 ArcCos, 414
 ArcCosh, 384
 ArcCot, 415
 ArcCoth, 384
 ArcCsc, 415
 ArcCsch, 384
 ArcSec, 415
 ArcSech, 385
 ArcSin, 415
 ArcSinh, 385
 ArcTan, 416
 ArcTanh, 385
 Arg, 461
 Array, 430
 ArrayDepth, 563
 ArrayQ, 576
 Arrow, 142
 Arrowheads, 144
 Association, 427
 AssociationQ, 428
 Associations, 427
 Assuming, 461
 Atomic Primitives, 65
 AtomQ, 65
 Attributes, 121
 Automatic, 167
 Axes, 167
 Axis, 168

 B64Decode, 312, 342
 B64Encode, 312, 342
 BalancedTree, 627
 BarbellGraph, 627
 BarChart, 245
 BaseForm, 212
 Basic Arithmetic, 45
 Basic Image Processing, 281
 Begin, 517
 BeginPackage, 517
 BernsteinBasis, 268
 Bessel and Related Functions, 522
 BesselI, 525
 BesselJ, 526
 BesselJZero, 526
 BesselK, 526

 BesselY, 527
 BesselYZero, 527
 Beta, 538
 BetweennessCentrality, 605
 BezierCurve, 268
 BezierFunction, 269
 Binarize, 289
 Binary Reading and Writing, 84
 Binary Types, 86
 BinaryImageQ, 303
 BinaryRead, 84
 BinarySearch, 176
 BinaryWrite, 85
 Binomial, 348
 BinomialTree, 628
 BitLength, 388
 Black, 100
 Blank, 504
 BlankNullSequence, 505
 BlankSequence, 505
 Blend, 95
 Block, 517
 Blue, 100
 Blur, 281
 Boole, 462
 BooleanQ, 568
 Bottom, 168
 BoxMatrix, 472
 BrayCurtisDistance, 137
 Break, 498
 Brown, 101
 Byte, 86
 Byte Arrays, 86
 ByteArray, 86
 ByteCount, 175
 ByteOrdering, 87

 C, 381
 Calculus, 369
 CanberraDistance, 137
 Cancel, 359
 Cases, 435
 Catalan, 398
 CatalanNumber, 349
 Catch, 499
 Catenate, 449
 Ceiling, 388
 Center, 420
 Centralities, 605
 CentralMoment, 131
 Character, 186, 320
 Character Codes, 547
 CharacterRange, 549
 Characters, 549
 Characters in Strings, 549

ChartLabels, 169
 ChartLegends, 169
 ChebyshevT, 542
 ChebyshevU, 543
 Check, 475
 ChessboardDistance, 138
 Chop, 479
 Circle, 146
 Clear, 52
 ClearAll, 53
 ClearAttributes, 122
 Clearing Assignments, 52
 ClearTrace, 595
 ClebschGordan, 236
 Close, 187, 320
 ClosenessCentrality, 606
 Closing, 306
 Cluster Analysis, 135
 ClusteringComponents, 135
 CMYKColor, 90, 145
 Coefficient, 359
 CoefficientArrays, 360
 CoefficientList, 360
 Collect, 361
 Color Directives, 90
 Color Operations, 95
 ColorCombine, 291
 ColorConvert, 95
 ColorData, 247
 ColorDataFunction, 248
 ColorDistance, 90
 Colorize, 292
 ColorNegate, 95
 ColorQuantize, 291
 ColorSeparate, 292
 Combinatorial Functions, 348
 Compile, 88
 CompiledFunction, 88
 Complement, 449
 CompleteGraph, 628
 CompleteKaryTree, 629
 Complex, 462
 Complexes, 369
 ComplexInfinity, 398
 CompositeQ, 352
 Composition, 223, 231
 CompoundExpression, 499
 Condition, 506
 ConditionalExpression, 462
 Cone, 270
 Conjugate, 463
 ConnectedComponents, 618
 ConnectedGraphQ, 624
 Constant, 122
 ConstantArray, 430
 Constructing Lists, 430
 Constructing Matrices, 472
 Constructing Vectors, 484
 Containing, 640
 ContainsOnly, 448
 Context, 78
 Contexts, 518
 Continue, 499
 ContinuedFraction, 403
 CoprimeQ, 584
 CopyDirectory, 199, 330
 CopyFile, 200, 330
 Core routines for working with Graphs., 610
 Correlation, 130
 Cos, 416
 Cosh, 386
 CosineDistance, 138
 Cot, 416
 Coth, 386
 Count, 436
 Covariance, 130
 CreateDirectory, 200, 330
 CreateFile, 200, 331
 CreateTemporary, 200, 331
 Cross, 485
 Csc, 416
 CubeRoot, 45
 Cuboid, 271
 Curated Graphs, 618
 Curl, 485
 Cyan, 102
 CycleGraph, 630
 Cylinder, 272
 D, 369
 DamerauLevenshteinDistance, 139
 Darker, 96
 DataImport, 182
 DateDifference, 115
 DateList, 116
 DateObject, 117
 DatePlus, 117
 DateString, 117
 Decrement, 60
 Default, 491
 DefaultValues, 62
 Definition, 78
 Degree, 399
 DegreeCentrality, 607
 Delete, 436
 DeleteCases, 437
 DeleteDirectory, 200, 331
 DeleteDuplicates, 449
 DeleteFile, 201, 331
 DeleteStopwords, 642

Denominator, 362
 DensityPlot, 248
 Dependency and Dispursion Statistics, 130
 Depth, 177
 Derivative, 371
 DesignMatrix, 392
 Det, 392
 Diagonal, 473
 DiagonalMatrix, 472
 DiamondMatrix, 472
 DiceDissimilarity, 349
 DictionaryLookup, 638
 DictionaryWordQ, 638
 Differential Equations, 381
 DigitCharacter, 557
 DigitCount, 388
 DigitQ, 549
 Dilation, 306
 Dimensions, 563
 DirectedEdge, 610
 DirectedGraphQ, 625
 DirectedInfinity, 464
 Directive, 147
 Directory, 201, 332
 DirectoryName, 201, 332
 DirectoryQ, 202, 332
 DirectoryStack, 202, 332
 DiscreteLimit, 372
 DiscretePlot, 249
 DisjointQ, 577
 Disk, 147
 DiskMatrix, 473
 Dispatch, 506
 Divide, 45
 DivideBy, 60
 Divisible, 352
 Division-Related Functions, 352
 Divisors, 404
 Do, 500
 Dodecahedron, 278
 DominantColors, 97
 Dot, 564
 DownValues, 80
 Drop, 438
 DSolve, 381

 E, 399
 EasterSunday, 118
 EdgeConnectivity, 610
 EdgeCount, 621
 EdgeDelete, 611
 EdgeDetect, 304
 EdgeForm, 149
 EdgeIndex, 611
 EdgeList, 611

 EdgeRules, 611
 EditDistance, 140
 Eigensystem, 392
 Eigenvalues, 393
 EigenvectorCentrality, 608
 Eigenvectors, 393
 ElementData, 496
 Elements of Lists, 434
 Elliptic Integrals, 533
 EllipticE, 533
 EllipticF, 534
 EllipticK, 534
 EllipticPi, 534
 End, 518
 EndOfFile, 187, 320
 EndOfLine, 557
 EndOfString, 558
 EndPackage, 518
 Environment, 242
 Equal, 569
 Equality and Inequality, 568
 Equivalent, 581
 Erf, 535
 Erfc, 535
 Erosion, 307
 Error Function and Related Functions, 535
 EuclideanDistance, 138
 EulerGamma, 399
 EulerPhi, 404
 EvenQ, 584
 ExactNumberQ, 67
 Except, 507
 Exp, 382
 Expand, 362
 ExpandAll, 363
 ExpandDenominator, 364
 ExpandFileName, 202, 332
 ExpIntegralE, 537
 ExpIntegralEi, 537
 Exponent, 364
 Exponential Functions, 381
 Exponential Integral and Special Functions, 537
 Export, 312, 343
 ExportString, 313, 343
 Expression, 187, 320
 Expression Sizes and Signatures, 175
 Expression Tests, 576
 Extract, 438

 FaceForm, 150
 Factor, 364
 Factorial, 539
 Factorial2, 539
 FactorInteger, 404
 FactorTermsList, 365

Failure, 476
 False, 581
 Fibonacci, 355
 File, 202, 333
 File and Stream Operations, 319
 FileNameDepth, 205, 335
 FileNameJoin, 205, 335
 FileNames, 206, 336
 FileNameSplit, 205, 336
 FileNameTake, 206, 336
 FilePrint, 187, 320
 Filesystem Operations, 329
 FileType, 206, 337
 FilledCurve, 150
 Filling, 169
 FilterRules, 491
 Find, 187, 320
 FindClusters, 135
 FindFile, 207, 337
 FindList, 207, 337
 FindMaximum, 372
 FindMinimum, 373
 FindRoot, 373
 FindShortestPath, 611
 FindSpanningTree, 623
 FindVertexCut, 612
 First, 438
 FirstCase, 439
 FirstPosition, 439
 FittedModel, 394
 FixedPoint, 232
 FixedPointList, 232
 Flat, 123
 Flatten, 450
 Floor, 389
 Fold, 233
 FoldList, 233
 FontColor, 151
 For, 500
 Form variables, 211
 Format, 420
 FormBaseClass, 212, 218
 Forms of Assignment, 55
 Forms which appear in \$OutputForms., 212
 Forms which are not in \$OutputForms, 218
 FractionalPart, 405
 FreeQ, 177
 FresnelC, 536
 FresnelS, 536
 FromCharCode, 547
 FromContinuedFraction, 405
 FromDigits, 389
 Full, 170
 FullDataImport, 182
 FullForm, 212
 FullSimplify, 366
 Function, 220, 229
 Function Application, 229
 Functional Composition and Operator Forms, 231
 Gamma, 540
 Gamma and Related Functions, 538
 Gather, 450
 GatherBy, 451
 GaussianFilter, 296
 GCD, 353
 GegenbauerC, 543
 General, 476
 General Statistics, 130
 General Structural Expression Functions, 176
 Geometric Operations, 285
 Get, 188, 321
 GetEnvironment, 242
 Glaisher, 400
 GoldenRatio, 400
 Graph, 612
 Graph Components and Connectivity, 618
 Graph Measures and Metrics, 621
 Graph Operations and Modifications, 623
 Graph Properties and Measurements, 623
 GraphAtlas, 630
 GraphData, 618
 GraphDistance, 621
 Graphics, 151, 250, 273
 Graphics3D, 252, 274
 Gray, 102
 GrayLevel, 91, 153
 Greater, 571
 GreaterEqual, 571
 Green, 103
 Grid, 421
 GridBox, 213
 Gudermannian, 386
 HammingDistance, 140
 HankelH1, 528
 HankelH2, 528
 HarmonicNumber, 355
 Hash, 175, 207, 337
 Haversine, 417
 Head, 66

HermiteH, 543
 HexadecimalCharacter, 73
 HighlightGraph, 613
 Histogram, 253
 HTSCentrality, 608
 HknHararyGraph, 631
 HmnHararyGraph, 631
 HoldAll, 123
 HoldAllComplete, 123
 HoldFirst, 123
 HoldPattern, 507
 HoldRest, 124
 HTML, 182
 HTMLGet, 182
 HTMLGetString, 183
 Hue, 91, 153
 Hyperbolic Functions, 384
 HyperlinksImport, 183

 I, 464
 Icosahedron, 278
 Identity, 223, 232
 IdentityMatrix, 473
 If, 501
 Im, 464
 Image Colors, 289
 Image Compositions, 293
 Image Filters, 296
 Image Properties, 301
 Image testing, 303
 ImageAdd, 293
 ImageAdjust, 282
 ImageAspectRatio, 301
 ImageChannels, 301
 ImageColorSpace, 293
 ImageConvolve, 297
 ImageData, 302
 ImageDimensions, 302
 ImageLinksImport, 183
 ImageMultiply, 294
 ImagePartition, 283
 ImageQ, 303
 ImageReflect, 285
 ImageResize, 287
 ImageRotate, 288
 ImageSize, 170
 ImageSubtract, 295
 ImageTake, 308
 ImageType, 302
 Implies, 581
 Import, 314, 344
 Importing and Exporting, 342
 ImportString, 314, 345
 In, 592
 In-place binary assignment operator, 59

 Increment, 60
 Indeterminate, 400
 Inequality, 571
 InexactNumberQ, 68
 Infinity, 400
 Infix, 421
 Information, 81
 Inner, 564
 InputForm, 213
 InputStream, 188, 322
 Insert, 440
 Inset, 153
 Integer, 465
 Integer Functions, 388
 IntegerDigits, 390
 IntegerExponent, 68
 IntegerLength, 69
 IntegerQ, 585
 IntegerReverse, 391
 Integers, 374
 IntegerString, 391
 Integrate, 375
 InterpretedBox, 73
 Interrupt, 501
 IntersectingQ, 577
 Intersection, 451
 Inverse, 394
 InverseErf, 536
 InverseErfc, 537
 InverseGudermannian, 386
 InverseHaversine, 417
 Iteratively Applying Functions, 232

 JaccardDissimilarity, 349
 JacobiP, 543
 Join, 451
 Joined, 171

 KaryTree, 632
 KatzCentrality, 608
 KelvinBei, 528
 KelvinBer, 528
 KelvinKei, 529
 KelvinKer, 530
 Key, 428
 Keys, 428
 Khinchin, 401
 KnownUnitQ, 599
 KroneckerProduct, 487
 Kurtosis, 134

 LABColor, 92, 154
 LadderGraph, 633
 LaguerreL, 544
 Language Translation, 637

LanguageIdentify, 638
 Large, 154
 Last, 440
 LCHColor, 92, 154
 LCM, 353
 LeafCount, 176
 LeastSquares, 394
 Left, 422
 LegendreP, 544
 LegendreQ, 545
 Length, 440
 LerchPhi, 546
 Less, 571
 LessEqual, 572
 LetterCharacter, 558
 LetterNumber, 74
 LetterQ, 549
 Level, 178
 LevelQ, 577
 LightBlue, 104
 LightBrown, 105
 LightCyan, 105
 Lighter, 98
 LightGray, 106
 LightGreen, 106
 LightMagenta, 107
 LightOrange, 107
 LightPink, 108
 LightPurple, 108
 LightRed, 109
 LightYellow, 109
 Limit, 376
 Line, 154
 Linear algebra, 392
 LinearModelFit, 394
 LinearSolve, 395
 Linguistic Data, 638
 List, 41, 226, 431
 List-Oriented Tests, 576
 Listable, 124
 ListLinePlot, 253
 ListLogPlot, 254
 ListPlot, 255
 ListQ, 576
 LoadModule, 55
 Location Statistics, 131
 Locked, 124
 Log, 382
 Log10, 383
 Log2, 383
 LogGamma, 541
 Logical Combinations, 580
 LogisticSigmoid, 383
 LogPlot, 257
 Longest, 507
 Lookup, 429
 LoopFreeGraphQ, 625
 LowerCaseQ, 550
 LUVColor, 92, 154
 MachineNumberQ, 585
 MachinePrecision, 69
 Magenta, 110
 MakeBoxes, 213, 218, 422, 458
 ManhattanDistance, 139
 MantissaExponent, 405
 Map, 41, 226
 MapAt, 42, 227
 MapIndexed, 42, 227
 MapThread, 43, 228
 MatchingDissimilarity, 349
 MatchQ, 576
 Math & Counting Operations on Lists, 447
 Mathematical Constants, 398
 Mathematical Operations, 485
 MathicsVersion, 243
 MathMLForm, 214
 MatrixExp, 396
 MatrixForm, 214
 MatrixPower, 396
 MatrixQ, 578
 MatrixRank, 396
 Max, 572
 MaxFilter, 299
 Maximize, 471
 MaxRecursion, 171
 Mean, 131
 MedianFilter, 300
 Medium, 155
 MemberQ, 578
 MemoryAvailable, 243
 MemoryInUse, 243
 Mesh, 171
 Message, 476
 MessageName, 476
 Messages, 62
 Min, 573
 MinFilter, 300
 MinimalPolynomial, 366
 Minimize, 471
 Minus, 46
 Miscellaneous image-related functions, 303
 Missing, 429
 MixedGraphQ, 625
 Mod, 353
 ModularInverse, 354
 Module, 519
 Morphological Image Processing, 306
 MorphologicalComponents, 307
 Most, 441

MultigraphQ, 625
 Multinomial, 350
 N, 479
 Named Colors, 100
 Names, 81
 Nand, 582
 Nearest, 136
 Needs, 208, 339
 Negative, 585
 Nest, 234
 NestList, 234
 NestWhile, 235
 NextPrime, 406
 NHoldAll, 125
 NHoldFirst, 125
 NHoldRest, 125
 NIntegrate, 376
 NonAssociative, 423
 None, 491
 NoneTrue, 582
 NonNegative, 586
 NonPositive, 586
 Nor, 582
 Norm, 486
 Normal, 431
 Normalize, 487
 Not, 583
 NotListQ, 579
 NotOptionQ, 492
 Now, 118
 Null, 179
 NullSpace, 396
 Number, 189, 322
 Number theoretic functions, 403
 NumberForm, 214
 NumberLinePlot, 257
 NumberQ, 586
 NumberString, 74
 Numerator, 366
 Numerical Data, 137
 Numerical Properties, 584
 NumericFunction, 125
 NumericQ, 586
 NValues, 63
 O, 377
 Octahedron, 279
 OddQ, 587
 Off, 477
 Offset, 155
 On, 477
 OneIdentity, 126
 Opacity, 93, 155
 OpenAppend, 189, 322
 Opening, 308
 OpenRead, 189, 322
 OpenWrite, 189, 323
 Operate, 179
 Operations on Image Structure, 308
 Operations on Strings, 551
 Optional, 508
 OptionQ, 492
 Options, 493
 OptionsPattern, 508
 OptionValue, 493
 Or, 583
 Orange, 110
 Order, 180
 Order Statistics, 131
 OrderedQ, 180
 Orderless, 126
 Orthogonal Polynomials, 542
 Out, 593
 Outer, 565
 OutputForm, 215
 OutputStream, 190, 323
 Overflow, 402
 OwnValues, 82
 PadLeft, 452
 PadRight, 453
 PageRankCentrality, 609
 Parametric Graphs, 627
 ParametricPlot, 258
 ParentDirectory, 208, 339
 Part, 441
 Partition, 453
 PartitionsP, 406
 Parts of Matrices, 473
 PathGraph, 633
 PathGraphQ, 626
 Pattern, 509
 PatternsOrderedQ, 180
 PatternTest, 510
 PauliMatrix, 236
 Pause, 119
 Permutations, 431
 Pi, 402
 Pick, 443
 Piecewise, 465
 PieChart, 259
 Pink, 111
 Pixel Operations, 309
 PixelValue, 309
 PixelValuePositions, 309
 PlaintextImport, 183, 184
 PlanarGraphQ, 626
 Plot, 261
 Plot3D, 264

PlotPoints, 172
PlotRange, 173
Plotting Data, 245
Pluralize, 644
Plus, 46
Pochhammer, 541
Point, 156
PointSize, 157
PolarPlot, 266
PolyGamma, 542
Polygon, 159
PolynomialQ, 366
Position, 443
Positive, 587
PossibleZeroQ, 588
Postfix, 423
Power, 47
PowerExpand, 367
PowerMod, 354
Precedence, 423
Precision, 71
PreDecrement, 60
Predicates on Lists, 448
Prefix, 423
PreIncrement, 61
Prepend, 444
PrependTo, 444
Prime, 406
PrimePi, 407
PrimePowerQ, 407
PrimeQ, 588
Print, 318
PrintTrace, 596
Product, 466
ProductLog, 538
Projection, 487
Property, 614
PropertyValue, 614
Protect, 127
Protected, 127
PseudoInverse, 397
Purple, 111
Put, 190, 323
PutAppend, 191, 324
PythonForm, 215

QRDecomposition, 397
Quantile, 131
Quantity, 599
QuantityMagnitude, 599
QuantityQ, 600
QuantityUnit, 600
Quartiles, 132
Quiet, 477
Quotient, 354

QuotientRemainder, 355
Random, 408
Random Graphs, 635
Random number generation, 408
RandomChoice, 409
RandomComplex, 409
RandomGraph, 635
RandomImage, 305
RandomInteger, 410
RandomPrime, 408
RandomReal, 411
RandomSample, 411
RandomTree, 634
RandomWord, 638
Range, 432
RankedMax, 132
RankedMin, 132
Rational, 467
Rationalize, 481
Re, 467
Read, 191, 325
ReadList, 193, 326
ReadProtected, 128
Real, 467
RealDigits, 71
RealNumberQ, 468
Reals, 377
Reap, 432
Rearranging and Restructuring Lists, 449
Record, 193, 326
Rectangle, 161
Recurrence and Sum Functions, 355
Red, 112
RegisterExport, 315, 345
RegisterImport, 315, 346
Regular Expressions, 557
RegularExpression, 557
RegularPolygon, 162
Remove, 54
RemoveDiacritics, 74
RenameDirectory, 208, 340
RenameFile, 209, 340
Repeated, 510
RepeatedNull, 511
Replace, 511
ReplaceAll, 512
ReplaceList, 513
ReplacePart, 445
ReplaceRepeated, 513
Representation of Numbers, 66
ResetDirectory, 209, 340
Rest, 445
Return, 501
Reverse, 454

RGBColor, 94, 99, 160, 275
 Riffle, 454
 Right, 424
 RogersTanimotoDissimilarity, 350
 Root, 377
 RotateLeft, 454
 RotateRight, 455
 RotationTransform, 566
 Round, 482
 Row, 424
 RowBox, 215, 219, 429, 433, 446
 RowReduce, 397
 RSolve, 520
 Rule, 514
 RuleDelayed, 514
 Run, 243
 RussellRaoDissimilarity, 350

 SameQ, 573
 ScalingTransform, 566
 Scan, 43, 228
 Sec, 417
 Sech, 387
 SeedRandom, 412
 Select, 446
 SequenceHold, 128
 Series, 378
 SeriesData, 378
 SessionTime, 119
 Set, 55
 SetAttributes, 129
 SetDelayed, 56
 SetDirectory, 209, 340
 SetFileDate, 209, 341
 SetOptions, 494
 SetStreamPosition, 193, 326
 Shape Statistics, 134
 Share, 244
 Sharpen, 284
 ShearingTransform, 566
 Shortest, 514
 Show, 163
 Sign, 468
 SimpleGraphQ, 626
 Simplify, 367
 Sin, 418
 SingularValueDecomposition, 398
 Sinh, 387
 SixJSymbol, 237
 Skewness, 134
 Skip, 194, 327
 Slot, 221, 230
 SlotSequence, 221, 231
 Small, 164
 SokalSneathDissimilarity, 350

 Solve, 379
 Sort, 133
 SortBy, 181
 SourceImport, 183
 Sow, 433
 Span, 446
 SparseArray, 521
 SpellingCorrectionList, 640
 Sphere, 276
 SphericalBesselJ, 530
 SphericalBesselY, 531
 SphericalHankelH1, 531
 SphericalHankelH2, 531
 SphericalHarmonicY, 545
 Splines, 268
 Split, 455
 SplitBy, 456
 Sqrt, 48
 SquaredEuclideanDistance, 139
 StandardForm, 216
 StarGraph, 634
 StartOfLine, 558
 StartOfString, 559
 StieltjesGamma, 542
 StirlingS1, 356
 StirlingS2, 356
 StreamPosition, 194, 327
 Streams, 194, 327
 String, 76
 String Distances and Similarity Measures, 139
 String Manipulation, 72
 String Patterns, 557
 StringCases, 559
 StringContainsQ, 75
 StringDrop, 551
 StringExpression, 560
 StringForm, 219
 StringFreeQ, 560
 StringInsert, 551
 StringJoin, 552
 StringLength, 552
 StringMatchQ, 561
 StringPosition, 553
 StringQ, 75
 StringRepeat, 76
 StringReplace, 553
 StringReverse, 554
 StringRiffle, 554
 StringSplit, 555
 StringTake, 555
 StringToStream, 195, 328
 StringTrim, 556
 StruveH, 531
 StruveL, 532
 Style, 424

Subscript, 425
 SubsetQ, 579
 Subsets, 351
 Subsuperscript, 425
 Subtract, 49
 SubtractFrom, 61
 SubValues, 63
 Sum, 469
 Sums, Simple Statistics, 50
 Superscript, 425
 Switch, 502
 Symbol, 83
 Symbol Handling, 78
 SymbolName, 82
 SymbolQ, 82
 SympyForm, 216
 Syntax, 478
 System-related binary handling, 87
 Table, 433
 TableForm, 216
 TagSet, 57
 TagSetDelayed, 58
 TagsImport, 184
 Take, 447
 TakeLargest, 133
 TakeLargestBy, 447
 TakeSmallest, 134
 TakeSmallestBy, 448
 Tally, 456
 Tan, 418
 Tanh, 387
 Tetrahedron, 279
 TeXForm, 218
 Text, 165
 Text Analysis, 640
 Text Normalization, 642
 TextCases, 642
 TextPosition, 643
 TextRecognize, 305
 TextSentences, 643
 TextStructure, 643
 TextWords, 644
 Thick, 165
 Thickness, 165
 Thin, 166
 Thread, 44, 229
 Three-Dimensional Graphics, 270
 ThreeJSymbol, 238
 Threshold, 285
 Through, 181
 Throw, 502
 TicksStyle, 173
 TimeConstrained, 119
 TimeRemaining, 119
 Times, 49
 TimesBy, 61
 TimeUsed, 120
 Timing, 120
 Tiny, 166
 TitleImport, 183
 ToBoxes, 459
 ToCharacterCode, 548
 ToExpression, 76
 ToFileName, 210, 341
 Together, 368
 ToLowerCase, 550
 Top, 174
 ToString, 77
 Total, 50
 ToUpperCase, 550
 Tr, 398
 TraceBuiltins, 596
 TraceEvaluation, 597
 TraditionalForm, 218
 TransformationFunction, 566
 TranslationTransform, 566
 Transliterate, 77
 Transpose, 567
 TreeGraph, 635
 TreeGraphQ, 636
 Trees, 635
 Trigonometric Functions, 413
 True, 583
 TrueQ, 574
 Tube, 277
 Tuples, 434
 Types of Values, 62
 Undefined, 402
 Underflow, 403
 UndirectedEdge, 614
 Unequal, 574
 Uniform Polyhedra, 278
 UniformPolyhedron, 280
 Union, 456
 Unique, 519
 UnitConvert, 600
 UnitVector, 488
 Unprotect, 129
 UnsameQ, 575
 Unset, 54
 UpperCaseQ, 550
 UpSet, 58
 UpSetDelayed, 59
 UpTo, 447
 UpValue-related assignments, 64
 UpValues, 64
 URLFetch, 316, 347
 URLSave, 210, 341

ValueQ, 83
Values, 429
Variables, 369
Vector Space Operations, 487
VectorAngle, 488
VectorQ, 580
Verbatim, 515
VertexAdd, 614
VertexConnectivity, 615
VertexCount, 622
VertexDegree, 623
VertexDelete, 616
VertexIndex, 617
VertexList, 617

WeaklyConnectedComponents, 620
WeberE, 532
Which, 502
While, 503
White, 112
Whitespace, 78
WhitespaceCharacter, 561
With, 519
Word, 195, 328
Word manipulation, 644
WordBoundary, 561
WordCharacter, 562
WordCloud, 295
WordCount, 641
WordData, 639
WordDefinition, 639
WordFrequency, 641
WordList, 639
WordSimilarity, 641
WordStem, 640, 642
Write, 195, 328
WriteString, 196, 329

XML, 184
XMLElement, 184
XMLGet, 185
XMLGetString, 185
XMLObject, 185
XMLObjectImport, 184, 185
Xor, 583
XYZColor, 94, 166

Yellow, 113
YuleDissimilarity, 352

Zeta, 546
Zeta Functions and Polylogarithms, 545

COLOPHON

Mathics3 Core

6.0.0

Python

3.8.16 (default, Dec 13 2022, 19:27:23) [GCC 11.3.0]

mpmath

1.2.1

NumpyPy

1.24.0

SymPy

1.10.1

XeTeX

XeTeX 3.141592653-2.6-0.999993 (TeX Live 2022/dev/Debian)

Asymptote

Asymptote version 2.83 [(C) 2004 Andy Hammerlindl, John C. Bowman, Tom Prince]

Ghostscript

9.56.1