

Run Mathematica

mathematica (evoke Mathematica frontend) Quit[] or Exit[] (exit Mathematica)

Arithmetic

+, -, * (or a space \square), /, ^ (power), e.g. $a*x^2 + 2x - 3/5$ ($ax^2 + 2x - 3/5$) n^{digits} base n number

Mathematical Functions

Sqrt[x]	(\sqrt{x})	Exp[x] or E^x	(e^x)	Log[x]	($\ln x$)	Log[b, x]	($\log_b x$)
Sin[x]	($\sin x$)	Cos[x]	($\cos x$)	Tan[x]	($\tan x$)	ArcSin[x]	($\sin^{-1} x$)
ArcCos[x]	($\cos^{-1} x$)	ArcTan[x]	($\tan^{-1} x$)	Sinh[x]	($\sinh x$)	Cosh[x]	($\cosh x$)
Tanh[x]	($\tanh x$)	ArcSinh[x]	($\sinh^{-1} x$)	n!	(factorial)	Abs[x]	($ x $)
Mod[n, m]	($n \bmod m$)	Max[x, y, ...]	(max)	Min[x, y, ...]	(min)	Binomial[n, m]	(C_m^n)
Re[z]	(real part)	Im[z]	(imaginary part)	Conjugate[z]	(\bar{z})	Arg[z]	(argument)
LegendreP[n, x]	($P_n(x)$)	HermiteH[n, x]	($H_n(x)$)	LaguerreL[n, a, x]	($L_n^a(x)$)	BesselJ[n, z]	($J_n(z)$)
Erf[x]	($\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$)	Gamma[z]	(Euler $\Gamma(z)$)	Zeta[s]	(Riemann $\zeta(s)$)	Random[]	(number)

Mathematical Constants

Pi ($\pi \approx 3.14159$) E ($e \approx 2.71828$) I ($i = \sqrt{-1}$) Infinity (∞)

Data Objects

123 (integer) 3/7 (rational) 1.0 (real) 2 + 8I (complex) "text" (string)
 {a, b, ...} (list with elements a, b, ...) h[a, b, ...] (expression/w head h, elements a, b, ...)
 expr[[i]] (i-th element of the expression expr) a[i] (indexed object)

Arbitrary-Precision Calculation

expr//N or N[expr] (numerical value of expr) N[expr, n] (value of expr with n-digit precision)
 Rationalize[x] (rational number approximation) Precision[x] (significant decimal digits in x)

Defining Variables, Functions, and Rules

x = value (assign a value to symbol x) expr /. x -> a (replace x by a in expr)
 x := expr (delayed assignment) expr //. x -> a (replace repeatedly)
 x = . or Clear[x] (remove value assigned to x) lhs :=> rhs /; test (apply rule if test is True)
 f[x_] := expr (define a function f(x)) lhs := rhs /; test (assign if test is True)

Algebraic Calculations

Expand[expr]	(multiply out products)	Factor[expr]	(reduce to a product of factors)
Together[expr]	(common denominator)	Cancel[expr]	(cancel common factors)
Expand[expr, Trig->True]	($\sin^2 x \rightarrow \sin 2x$, etc.)	Simplify[expr]	(find simplest form)
Factor[expr, Trig->True]	($\sin 2x \rightarrow \sin^2 x$, etc.)	Apart[expr]	(write expr as a sum of terms)
Coefficient[expr, x]	(coefficient of x in expr)	Exponent[expr, x]	(max power of x in expr)
Solve[lhs==rhs, x]	(solve algebraic equation for x)	DSolve[eqn, y[x], x]	(solve differential equation)
Reduce[eqn, x]	(reduce equations)	Eliminate[eqn, e]	(eliminate variable e)

Linear Algebra

{a, b, c}	(vector)	{{a, b}, {c, d}}	(2x2 matrix)	n.m	(matrix multiply)
Inverse[m]	(inverse of matrix)	Transpose[m]	(transpose)	Det[m]	(determinant)
MatrixPower[m, n]	(m^n)	MatrixExp[m]	(e^m)	LinearSolve[m, b]	(solve $mx=b$)
Eigenvalues[m]	(eigenvalues)	Eigenvectors[m]	(eigenvectors)	Eigensystem[m]	(value & vector)

Calculus

D[f, x]	($\partial f / \partial x$)	Integrate[f, x]	($\int f dx$)	Series[f, {x, a, n}]	(expand at a)
D[f, {x, n}]	($\partial^n f / \partial x^n$)	Integrate[f, {x, a, b}]	($\int_a^b f dx$)	Limit[f, x->a]	($\lim_{x \rightarrow a} f$)
Dt[f]	(df)	Sum[f, {i, m, n}]	($\sum_{i=m}^n f$)	Product[f, {i, m, n}]	($\prod_{i=m}^n f$)
NIntegrate, NSum, NProduct, NSolve, NDSolve, FindRoot				(numerical integration, summation, etc.)	

Graphics

Plot[f, {x, a, b}, option->value] (plot f as a function of x from a to b)
 Show[plot₁, plot₂, ...] (redraw plots)
 Plot3D[f, {x, a, b}, {y, c, d}] (three-dimensional plot of f as a function of x and y)
 ListPlot[{x₁, y₁}, {x₂, y₂}, ...] (plot points (x₁, y₁) ...)
 ParametricPlot[{x, y}, {t, a, b}] (plot curve (x(t), y(t)))

Plot options:

AspectRatio (height-to-width ratio) AxesLabel->{"x", "y"} (add labels) Frame -> True (draw frame)

Programming

Module[{ <i>a</i> , <i>b</i> , ... }, <i>expr</i> ₁ ; <i>expr</i> ₂ ; ...]	(a procedure/w local variables <i>a</i> , <i>b</i> , ..., return value of last <i>expr</i>)		
Table[<i>expr</i> , { <i>i</i> , <i>max</i> }]	(make a list of values of <i>expr</i> with <i>i</i> from 1 to <i>max</i>)		
Do[<i>expr</i> , { <i>i</i> , <i>min</i> , <i>max</i> , <i>di</i> }]	(evaluate <i>expr</i> with <i>i</i> run from <i>min</i> to <i>max</i> in steps of <i>di</i>)		
While[<i>test</i> , <i>body</i>]	(evaluate <i>body</i> repetitively, so long as <i>test</i> is True)		
For[<i>start</i> , <i>test</i> , <i>inc</i> , <i>body</i>]	(evaluate <i>start</i> , then repetitively evaluate <i>body</i> and <i>inc</i> , until <i>test</i> fails)		
If[<i>test</i> , <i>then</i> , <i>else</i>]	(evaluate <i>then</i> if <i>test</i> is True, and <i>else</i> if it is False)		
Which[<i>test</i> ₁ , <i>value</i> ₁ , <i>test</i> ₂ , ...]	(give the value associated with the first <i>test</i> that is True)		
Switch[<i>expr</i> , <i>form</i> ₁ , <i>value</i> ₁ , <i>form</i> ₂ , ...]	(give the value associated with first <i>form</i> matching <i>expr</i>)		
Function[<i>x</i> , <i>body</i>] or <i>body</i> &	(specify a pure function)		
Nest[<i>f</i> , <i>x</i> , <i>n</i>]	(apply the function <i>f</i> nested <i>n</i> times to <i>x</i>)		
NestList[<i>f</i> , <i>x</i> , <i>n</i>]	(generate list { <i>x</i> , <i>f</i> (<i>x</i>), <i>f</i> (<i>f</i> (<i>x</i>)), ... } nested up to <i>n</i> deep)		
Fold[<i>f</i> , <i>x</i> , { <i>a</i> , <i>b</i> , <i>c</i> }]	(produce <i>f</i> (<i>f</i> (<i>f</i> (<i>x</i> , <i>a</i>), <i>b</i>), <i>c</i>))		
FixedPoint[<i>f</i> , <i>x</i>]	(apply the function <i>f</i> repeated until the result no longer changes)		
Apply[<i>f</i> , { <i>a</i> , <i>b</i> , <i>c</i> }] or f @@ <i>expr</i>	(produce <i>f</i> (<i>a</i> , <i>b</i> , <i>c</i>))		
Map[<i>f</i> , { <i>a</i> , <i>b</i> , <i>c</i> }] or f /@ <i>expr</i>	(apply <i>f</i> to each elements, { <i>f</i> (<i>a</i>), <i>f</i> (<i>b</i>), <i>f</i> (<i>c</i>))		
<i>i</i> ++ (post-increment)	<i>i</i> -- (post-decrement)	<i>i</i> += <i>di</i> (add <i>di</i> to <i>i</i>)	<i>x</i> *= <i>c</i> (multiply <i>x</i> by <i>c</i>)
++ <i>i</i> (pre-increment)	-- <i>i</i> (pre-decrement)	<i>i</i> -= <i>di</i> (subtract <i>di</i>)	<i>x</i> /= <i>c</i> (divide <i>x</i> by <i>c</i>)
(* <i>text</i> *) (comment)	<i>f</i> ::usage=" <i>text</i> " (info)	Timing[<i>expr</i>] (time)	MemoryInUse[] (space)
Iterators in Do, Table, Sum, etc.			
{ <i>max</i> }	(iterate <i>max</i> times)	{ <i>i</i> , <i>max</i> }	(<i>i</i> from 1 to <i>max</i> in steps of 1)
{ <i>i</i> , <i>min</i> , <i>max</i> }	(<i>i</i> from <i>min</i> to <i>max</i> in steps of 1)	{ <i>i</i> , <i>min</i> , <i>max</i> , <i>di</i> }	(<i>i</i> from <i>min</i> to <i>max</i> in steps of <i>di</i>)

Logical Operators

== (equal)	!= (unequal)	=== (identical)	!= (not identical)
< (less than)	<= (less or equal)	<i>p</i> <i>q</i> (or)	<i>p</i> && <i>q</i> (and)
> (greater than)	>= (greater or equal)	! <i>p</i> (not)	Xor[<i>p</i> , <i>q</i>] (exclusive or)

List Manipulation

Part[<i>t</i> , <i>i</i>] or <i>t</i> [[<i>i</i>]]	(<i>i</i> -th sublist)	Position[<i>t</i> , <i>form</i>]	(the position <i>form</i> occur)
Take[<i>t</i> , <i>n</i>]	(first <i>n</i> elements)	Last[<i>t</i>]	(last element in <i>t</i>)
Drop[<i>t</i> , <i>n</i>]	(drop first <i>n</i> elements)	First[<i>t</i>]	(first element in <i>t</i>)
Count[<i>t</i> , <i>form</i>]	(number of times <i>form</i> occur)	MemberQ[<i>t</i> , <i>form</i>]	(test whether <i>form</i> is in <i>t</i>)
Prepend[<i>t</i> , <i>e</i>]	(add <i>e</i> at the beginning)	Insert[<i>t</i> , <i>e</i> , <i>i</i>]	(insert <i>e</i> at position <i>i</i>)
Append[<i>t</i> , <i>e</i>]	(add <i>e</i> at the end)	Delete[<i>t</i> , <i>i</i>]	(delete element at position <i>i</i>)
ReplacePart[<i>t</i> , <i>e</i> , <i>i</i>]	(replace with <i>e</i> at position <i>i</i>)	Join[<i>t</i> ₁ , <i>t</i> ₂ , ...]	(concatenate lists together)
Union[<i>t</i> ₁ , <i>t</i> ₂ , ...]	(union of lists)	Intersection[<i>t</i> ₁ , <i>t</i> ₂ , ...]	(common to all lists)
Sort[<i>t</i>]	(sort elements in standard order)	Reverse[<i>t</i>]	(reverse the order of elements)
RotateLeft[<i>t</i> , <i>n</i>]	(rotate <i>n</i> places to the left)	RotateRight[<i>t</i> , <i>n</i>]	(rotate <i>n</i> places to the right)

Input/Output

<< <i>file</i> (read expressions from <i>file</i> , return last <i>expr</i>)	Save[" <i>file</i> ", <i>x</i>] (save the definition of <i>x</i> to <i>file</i>)
<i>expr</i> >> <i>file</i> (write <i>expr</i> to a <i>file</i>)	Display["!psfix> <i>file</i> ", <i>graph</i>] (save as PS file)
<i>expr</i> >>> <i>file</i> (append <i>expr</i> to a <i>file</i>)	ReadList[" <i>file</i> ", <i>type</i>] (read objects of a given <i>type</i>)
!! <i>file</i> (display the content of a <i>file</i>)	PSPrint[<i>graph</i>] (print a hardcopy of graphics)
<<Calculus'VectorAnalysis' (load a package)	! <i>command</i> (issue a UNIX command)

Patterns

- (any <i>expr</i>)	<i>x</i> _ (any <i>expr</i> named <i>x</i>)	<i>x</i> : <i>pattern</i> (match pattern)
<i>x</i> _ <i>h</i> (pattern with head <i>h</i>)	<i>pattern</i> /; <i>condition</i> (conditional)	<i>pattern</i> ? <i>test</i> (if <i>test</i> is True)
<i>x</i> __ (sequence of <i>expr</i>)	<i>x</i> ___ (zero or more <i>expr</i>)	<i>x</i> _: <i>v</i> (<i>expr</i> with default)

Expression in Different Formats

FullForm[<i>e</i>] (full form)	CForm[<i>e</i>] (C codes)	FortranForm[<i>e</i>] (fortran)	StandardForm[<i>e</i>] (math)
InputForm[<i>e</i>] (input)	OutputForm[<i>e</i>] (out)	MatrixForm[<i>e</i>] (matrix)	TeXForm[<i>e</i>] (T _E X)

Input Editing and Help

% (last result)	? <i>Name</i> (help on <i>Name</i>)	* (represent 0 or more characters)
% <i>n</i> (result on Out[<i>n</i>])	?? <i>Name</i> (more help on <i>Name</i>)	@ (1 or more lower-case letters)
Shift-Enter (evaluate <i>expr</i>)	Ctrl-C (interrupt execution)	! (shell escape)