Docs : Commands

Search:

!

#link

Syntax

```
value!
```

Description

Factorial. Returns the factorial of a positive integer. For non-integers, ! = Γ(x + 1). This calculates the Gamma function.

Example

```
6! returns 720
```

discussion

%

#link

Syntax

```
%(x, y)
```

Description

x percent of y. Returns (x/100)*y.

Example

```
%(20,50) returns 10
```

discussion

%CHANGE

#link

Syntax

```
%CHANGE(x, y)
```

Description

Percent change from x to y. Returns 100*(y–x)/x.
%CHANGE(20,50) returns 150

discussion

%TOTAL

#link

Syntax

```
%TOTAL(x, y)
```

Description

Percent total; the percentage of x that is y. Returns 100*y/x.
%TOTAL(20,50) returns 250.

discussion

(

#link

Syntax

discussion

*

#link

Syntax

```
Object1×Object2
```

Description

Multiplication.
Returns the result of multiplying Object1 and Object2. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions.

Example

```
3*2 returns 6
```

discussion

+

#link

Syntax

```
Object1 + Object2
```

Description

Addition.
Returns the result of adding Object2 to Object 1. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions.

Example

```
3+2 returns 5
```

discussion

−

#link

Syntax

`Object1 - Object2`

Description

Subtraction.
Returns the result of subtracting Object2 from Object 1. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions.

Example

`3-2 returns 1`

discussion

#link

.*

Syntax

`.*(Lst||Mtrx,Lst||Mtrx)`

Description

Performs an element-by-element multiplication of 2 lists or 2 matrices.

Example

`[[1,2],[3,4]] .* [[3,4],[5,6]] returns [[3,8],[15,24]]`

discussion

#link

.+

Syntax

`matrix .+ real/complex or real/complex .+ matrix`

Description

Adds the real/complex to each element of the matrix

Example

`[1,2].+3 returns [4,5]`

discussion

#link

.-

Syntax

`matrix .- real/complex or real/complex .- matrix`

Description

Substract the real/complex to each element of the matrix (or the reverce as appropriate)

Example

`[3,4].-2 returns [1,2]`

discussion

#link

./

Syntax

`./(Lst||Mtrx,Lst||Mtrx)`

Description

Performs an element-by-element division of 2 lists or 2 matrices.

Example

`[[1,2],[3,4]] ./ [[3,4],[5,6]] returns [[1/3,1/2],[3/5,2/3]]`

discussion

#link

.^

Syntax

`.^(Mtrx,Intg(n))`

Description

Calculates the power of each element of the matrix.

Example

`[[1,2],[3,4]] .^ 3 returns [[1,8],[27,64]]`

discussion

#link

/

Syntax

`Object1/Object2`

Description

Division.
Returns the result of dividing Object1 by Object2. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions.

Example

`3÷2 returns 1.5`

discussion

#link

:=

Syntax

variable := object

Description

Assigns object to variable.

Example

A := 3 stores the value 3 in the variable A

F1 := 3-X makes F1(X)=3-X

M5 := [1, 2] stores a vector in M5

discussion

<

#link

Syntax

Value1 < Value2

Description

<: Less than or equal to.
Tests whether or not Value1 is less than Value2. Returns 1 if true, 0 if false.

Example

2 ≤ 1 returns 0

discussion

<=

#link

Syntax

Value1 ≤ Value2

Description

≤: Less than or equal to.
Tests whether or not Value1 is less than Value 2. Returns 1 if true, 0 if false.

Example

2 ≤ 1 returns 0

Alternative: <=

discussion

<>

#link

Syntax

Value1 ≠ Value2

Description

≠: Not equal to.
Tests if Value1 is not equal to Value 2. Returns 1 if true, 0 if false.

Example

```
3 ≠ 5 returns 1
```

```
Alternatives: <>
```

discussion

=

#link

Syntax

```
Value1 == Value2
```

Description

==: equal to.
Tests is Value1=Value2. Returns 1 if true, 0 if false.

Example

```
3==2 returns 0
```

discussion

==

#link

Syntax

```
Value1 == Value2
```

Description

==: equal to.
Tests is Value1=Value2. Returns 1 if true, 0 if false.

Example

```
3==2 returns 0
```

discussion

>

#link

Syntax

```
Value1 > Value2
```

Description

>: Greater than.
Tests whether or not Value1 is greater than Value 2. Returns 1 if true, 0 if false.

Example

`2 > 1 returns 1`

discussion

#link

>=

Syntax

`Value1 ≥ Value2`

Description

≥: Greater than or equal to.
Tests whether or not Value 1 is either greater than or equal to Value2. Returns 1 if true, 0 if false.

Example

`3 ≥ 4 returns 0`

`Alternative: >=`

discussion

#link

^

Syntax

`Value1^Value2`

Description

Exponentiation.
Returns the result of raising Value1 to the power of Value2.

Example

`2^3 returns 8`

discussion

#link

`

Syntax

`QUOTE(expression)`

Description

Returns the expression unchanged and un-evaluated.
This function is mostly used with the STO► command in order to store a function in a function variable. For example if you want to store SIN(X) in F1.you cannot do SIN(X)►F1 as SIN(X) would be evaluated and a numerical result would be stored into F1.
QUOTE(SIN(X))►F1 will store SIN(X) in F1.

discussion

a2q

#link

Syntax

`a2q(Mtrx,VectVar)`

Description

a2q(A,X)=the quadratic form q associated to A, X=vector of variables.

Example

`a2q([[1,2],[4,4]],[x,y]) returns x^2+6*x*y+4*y^2`

discussion

abcuv

#link

Syntax

`abcuv(Polya,Polyb,Polyc,[Var])`

Description

Returns [u,v] suchthat au+bv=c for 3 polynomials a, b, and c.

Example

`abcuv(x^2+2*x+1,x^2-1,x+1) returns [1/2,-1/2]`

discussion

about

#link

Syntax

`about(Var(a))`

Description

Returns the hypothesis made with assume on the variable a.

Example

`about(n) returns n`

discussion

ABS

#link

Syntax

`ABS(expr) or ABS(matrix)`

Description

For numerical arguments, returns the absolute value of the expression. For matrix arguments, returns the returns the Frobenius (Euclidean) norm of the array.

Example

`ABS(-3.14) returns 3.14 and ABS([[1,2],[3,4]]) returns 5.47722557505`

discussion

#link

abscissa

Syntax

`abscissa(Pnt or Vect)`

Description

Returns the abscissa of a point or a vector.

Example

`abscissa(point(1+2*i)) returns 1`

discussion

#link

ACOS

Syntax

`ACOS(Value)`

Description

ACOS: the inverse cosine function.
This Shift-key combination returns the inverse cosine of Value. The output depends on the Angle Measure setting.

Example

`ACOS(-1) returns 3.14159265359`

discussion

#link

acos2asin

Syntax

`acos2asin(Expr)`

Description

Replaces arccos(x) by π/2−arcsin(x) in the argument Expr.

Example

`acos2asin(acos(x)+asin(x)) returns π/2-asin(x)+asin(x)`

discussion

#link

acos2atan

Syntax

```
acos2atan(Expr)
```

Description

Replaces arccos(x) by π/2−arctan(x/√(1−x^2)) in the argument.

Example

```
acos2atan(2*acos(x)) returns 2*(π/2-atan(x/(√(1-x^2))))
```

discussion

ACOSH

#link

Syntax

```
ACOSH(value)
```

Description

Inverse hyperbolic cosine.

Example

```
ACOSH(1.54308063482) returns 1
```

discussion

ACOT

#link

Syntax

```
ACOT(value)
```

Description

Arc cotangent. The function derived from the inverse of the Cotangent function.

Example

```
ACOT(1) returns 45 in degree mode
```

discussion

ACSC

#link

Syntax

```
ACSC(value)
```

Description

Arc cosecant. The function derived from the inverse of the Cosecant function.

Example

```
ACSC(1) returns 90 in degree mode
```

discussion

#link

ADDCOL

Syntax

```
ADDCOL(matrixname, vector, column_number)
```

Description

Add Column. Inserts values from vector into a column before column_number in the specified matrix. The size of vector must be the same as the number of rows in the matrix matrixname.

discussion

#link

additionally

Syntax

```
additionally(Expr)
```

Description

Make an additional assumption about a variable.

Example

```
assume(n,integer);additionally(n>5) returns DOM_INT,n
```

discussion

#link

ADDROW

Syntax

```
ADDROW(matrixname, vector, row_number)
```

Description

Add Row. Inserts values from vector into a row before row_number in the specified matrix. The size of vector must be the same as the number of columns in the matrix matrixname.

discussion

#link

affix

Syntax

```
affix(Point) or affix(Vector)
```

Description

Returns the coordinates of a point or both the x- and y-lengths of a vector as a complex number.

Example

```
affix(point(3,2)) returns 3+2*i
 if GA is a point at (1, -2), then affix(GA) returns 1-2*i.
```

discussion

#link

algvar

Syntax

```
algvar(Expr)
```

Description

List of the variables by ascending algebraic extension order.

Example

```
algvar(√x+y) returns [[y],[x]]
```

discussion

#link

ALOG

Syntax

```
ALOG(value)
```

Description

The common antilogarithm. This is more accurate than 10^x due to limitations of the power function.

Example

```
ALOG(2) returns 100
```

discussion

#link

alog10

Syntax

```
alog10(Expr)
```

Description

Function x->10^x.

Example

```
alog10(3) returns 1000
```

discussion

#link

altitude

Syntax

```
altitude(point1, point2, point3)
```

Description

Given three non-collinear points, draws the altitude of the triangle defined by the three points that passes through the first point. The triangle does not have to be drawn.

Example

`altitude(A, B, C) draws a line passing through point A that is perpendicular to BC.`

discussion

#link

AND

Syntax

```
Value1 AND Value2
```

Description

Logical AND.
Returns 1 if both value1 and value2 are non-zero; otherwise returns 0.

Example

`3 AND 2 returns 1`

discussion

#link

angle

Syntax

```
angle(Vertex, Point2, Point3)
```

Description

Returns the measure of a directed angle. The first point is taken as the vertex of the angle as the next two points in order give the measure and orientation.

Example

`angle(GA, GB, GC) returns the measure of ∡BAC`

discussion

#link

angleat

Syntax

```
angleat(Vertex, Point2, Point3, Point4)
```

Description

Used in Symbolic view. Given the three points of an angle and a fourth point as a location, displays the measure of the angle defined by the first three points, with a label, at the location in the Plot view given by the fourth point. The first point is the vertex of the angle.

Example

`angleat(GA, GB, GC, point(0,0)) displays “aA=” at the origin, followed by the measure of ∡BAC.`

discussion

#link

angleatraw

Syntax

```
angleatraw(Pnt(A)),Pnt(B),Pnt(C),(Pnt or Cplx(z0)))
```

Description

angleatraw(A,B,C,z0) displays at point(z0), the value of the measure of the angle (AB,AC).

discussion

#link

Ans

Syntax

```
ANS
```

Description

ANS: Last answer.
Returns the result of the last calculation made in Home view to its full precision. The variable ANS is different from the numbers in Home's history. A value in ANS is stored internally with the full precision of the calculated result, whereas the displayed numbers match the display mode.

discussion

#link

append

Syntax

```
append((Lst||Seq|| Set,Elem)
```

Description

Append an element to a list.

Example

`append([1,2,3],4) returns [1,2,3,4]`

discussion

#link

apply

Syntax

```
apply(Fnc(f),Lst(l))
```

Description

Apply the function f at the elements of the list l (option matrix for a matrix).

Example

```
apply(x->x^3,[1,2,3]) returns [1,8,27]
```

discussion

approx

#link

Syntax

```
approx(Expr,[Int])
```

Description

Numerical evaluation of the first argument (we can give the number of digits as second argument).
approx(expression) works also and does the same thing.

Example

```
approx(2/3) returns 0.666666666667
```

discussion

ARC

#link

Syntax

```
ARC(G, x, y, r, [[∡1, ∡2],[color]])
```

Description

Draws a circle on GROB G, centered at (x, y), with radius r. If ∠1 and ∠2 are specified, draws an arc from ∠1 to ∠2 using the current angle mode.

discussion

ARC_P

#link

Syntax

```
ARC_P(G, x, y, r, [[∡1, ∡2],[color]])
```

Description

Draws a circle on GROB G, centered at (x, y), with radius r. If ∠1 and ∠2 are specified, draws an arc from ∠1 to ∠2 using the current angle mode.

discussion

arcLen

#link

Syntax

```
arcLen(Expr, Real1, Real2)
```

Description

Returns the length of the arc of a curve between two points on the curve. The curve is an expression, the independent variable is declared, and the two points are defined by values of the independent variable.

This command can also accept a parametric definition of a curve. In this case, the expression is a list of 2 expressions (the first for x and the second for y) in terms of a third independent variable.

Example

```
arcLen(x^2, x, -2, 2) returns 9.29….
arcLen({sin(t), cos(t)}, t, 0, π/2) returns 1.57…
```

discussion

#link

area

Syntax

```
area(Circle) or area(Polygon) or area(Expr, x=value1..value2)
```

Description

Returns the area of a circle or polygon. Can also return the area under a curve between two points.

Example

```
If GA is defined to be the unit circle, then area(GA) returns π.
area(4-x^2/4, x=-4..4) returns 64/3 or 21.333…
```

discussion

#link

areaat

Syntax

```
areaat(Polygon, Pnt||Cplx(z0))
```

Description

Displays at point(z0), with a legend, algebraic area of a circle or of a (star) polygon (e.g. triangle, square, ...).

discussion

#link

areaatraw

Syntax

```
areaatraw(Polygon, Pnt||Cplx(z0))
```

Description
Displays at point(z0), algebraic area of a circle or of a (star-)polygon (e.g. triangle, square, ...).

discussion

ARG

#link

Syntax

```
ARG(x+yi)
```

Description
The ARG function finds the angle determined by a complex number.

Example

```
ARG(3+3i) returns 45 in degree mode.
```

discussion

ASC

#link

Syntax

```
ASC("string")
```

Description
Returns a vector containing the ASCII codes of string.

Example

```
ASC("AB") returns [65, 66]
```

discussion

ASEC

#link

Syntax

```
ASEC(value)
```

Description
Arc secant. The function derived from the inverse of the Secant function.

Example

```
ASEC(1) returns 0 in degree mode
```

discussion

ASIN

#link

Syntax

```
ASIN(Value)
```

Description

ASIN: the inverse sine function.
This Shift-key combination returns the inverse sine of Value. The output depends on the Angle Measure setting.

Example

```
ASIN(1) returns 1.57079632679
```

discussion

asin2acos

#link

Syntax

```
asin2acos(Expr)
```

Description

Replaces arcsin(x) by π/2−arccos(x) in the argument.

Example

```
asin2acos(acos(x)+asin(x)) returns π/2-acos(x)+acos(x)
```

discussion

asin2atan

#link

Syntax

```
asin2atan(Expr)
```

Description

Replaces arcsin(x) by arctan(x/√(1−x^2)) in the argument.

Example

```
asin2atan(2*asin(x)) returns 2*atan(x/(√(1-x^2)))
```

discussion

ASINH

#link

Syntax

```
ASINH(value)
```

Description

Inverse hyperbolic sine.

Example

```
ASINH(1.17520119365) returns 1
```

discussion

assume

#link

Syntax

```
assume(Expr)
```

Description

Make an assumption on a variable.

Example

```
assume(a>0) returns a
```

discussion

ATAN

#link

Syntax

```
ATAN(Value)
```

Description

ATAN: the inverse tangent function.
This Shift-key combination returns the inverse tangent of Value. The output depends on the Angle Measure setting.

Example

```
ATAN(0) returns 0
```

discussion

atan2acos

#link

Syntax

```
atan2acos(Expr)
```

Description

Replaces arctan(x) by π/2−arccos(x/√(1+x^2)) in the argument.

Example

```
atan2acos(atan(2*x) returns π/2-acos((2*x)/√(1+(2*x)^2))
```

discussion

atan2asin

#link

Syntax

```
atan2asin(Expr)
```

Description

Replaces arctan(x) by arcsin(x/√(1+x^2)) in the argument Expr.

Example

```
atan2asin(atan(y/x) returns asin((y/x)/√(1+(y/x)^2))
```

discussion

ATANH

#link

Syntax

```
ATANH(value)
```

Description

Inverse hyperbolic tangent.

Example

```
ATANH(.761594155956) returns 1
```

discussion

atrig2ln

#link

Syntax

```
atrig2ln(Expr)
```

Description

Rewrites the expression Expr containing inverse trigonometric functions with equivalent logarithmic functions.

Example

```
atrig2ln(atan(x)) returns (i*ln((i+x)/(i-x)))/2
```

discussion

barycenter

#link

Syntax

```
barycenter([Point1, Weight1], [Point2, Weight2],…,[Pointn, Weightn])
```

Description

Calculates the hypothetical center of mass of a set of points, each with a given weight (a real number). Each point, weight pair is enclosed in square brackets as a vector.

Example

```
barycenter([-3,1],[3,1],[4,2]) returns point(2,0)
```

discussion

basis

#link

Syntax

```
basis(Lst(vector1,..,vectorn))
```

Description

Extract a basis from a spanning set of vectors.

Example

```
basis([[1,2,3],[4,5,6],[7,8,9],[10,11,12]]) returns [[-3,0,3],[0,-3,-6]]
```

discussion

BEGIN

#link

Syntax

```
BEGIN commands; END;
```

Description

Defines a set of commands to be executed in a block.

Example

```
SQM1

EXPORT  SQM1(X)
BEGIN
RETURN X^2-1;
END;

This program defines a user function named SQM1(X). From the Home view,
entering SQM1(8) returns 63.
```

discussion

Beta

#link

Syntax

```
Beta(Expr,Expr)
```

Description

Returns Gamma(x)*Gamma(y)/Gamma(x+y).

Example

```
Beta(3,2) returns 1/12
```

discussion

BINOMIAL

#link

Syntax

```
BINOMIAL(n, k, p)
```

Description

Binomial probability density function. Computes the probability of k successes out of n trials, each with a probability of success, p.
Returns Comb(n,k) if there is no third argument. Note that n and k are integers with k=n.

Example

```
BINOMIAL(4, 2, 0.5) returns 0.375
```

discussion

BINOMIAL_CDF

#link

Syntax

```
BINOMIAL_CDF(n, p, k)
```

Description

Cumulative binomial distribution function. Returns the probability of k or fewer successes out of n trials, with a probability of success, p for each trial. Note that n and k are integers with k=n.

Example

```
BINOMIAL_CDF(4, 0.5, 2) returns 0.6875
```

discussion

BINOMIAL_ICDF

#link

Syntax

```
BINOMIAL_ICDF(n, p, q)
```

Description

Inverse cumulative binomial distribution function. Returns the number of successes, k, out of n trials, each with a probability of p, such that the probability of k or fewer successes is q.

Example

```
BINOMIAL_ICDF(4, 0.5, 0.6875) returns 2
```

discussion

bisector

#link

Syntax

```
bisector(Point1, Point2, Point3)
```

Description

Given three points, creates the bisector of the angle defined by the three points whose vertex is at the first point. The angle does not have to be drawn in the Plot view.

Example

```
bisector(GA, GB, GC) draws the bisector of ∡BAC.
bisector(0,-4i,4) draws the line given by y=-x
```

discussion

#link

BITAND

Syntax

```
BITAND(int1[, int2..,intn])
```

Description

Bitwise logical AND. Takes n integers as input and returns their bitwise logical AND.

Example

```
BITAND(20, 13) returns 4
```

discussion

#link

BITNOT

Syntax

```
BITNOT(int)
```

Description

Bitwise logical NOT. Takes one integer as input and returns its bitwise not.

discussion

#link

BITOR

Syntax

```
BITOR(int1[, int2..,intn])
```

Description

Bitwise logical OR. Takes n integers as input and returns their bitwise logical OR.

Example

```
BITOR(9, 26) returns 27
```

discussion

BITSL

#link

Syntax

```
BITSL(int1[, int2])
```

Description

Bitwise shift left. Takes one or two integers as input and returns the result of shifting the bits in the first integer to the left by the number of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the left one place.

Example

```
BITSL(28, 2) returns 112
BITSL(5) returns 10
```

discussion

BITSR

#link

Syntax

```
BITSR(int1[, int2])
```

Description

Bitwise shift right. Takes one or two integers as input and returns the result of shifting the bits in the first integer to the right by the number of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the right one place.

Example

```
BITSR(112, 2) returns 28
BITSR(10) returns 5
```

discussion

BITXOR

#link

Syntax

```
BITXOR(int1[, int2..,intn])
```

Description

Bitwise logical exclusive OR (XOR). Takes n integers as input and returns their bitwise XOR.

Example

```
BITXOR(9, 26) returns 19
```

discussion

#link

black

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB-GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

#link

BLIT

Syntax

```
BLIT([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2], [c])
```

Description

Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Does not copy pixels from srcG that are color c.

The defaults for the optional arguments are:
trgtG=G0
srcG=G0
sx1, sy1=srcGRB top left corner
sx2, sy2=srcGRB bottom right corner
dx1, dx2=trgtGRB top left corner
dx2, dy2=calculated so destination area is the same as source area
c=all pixel colors

discussion

#link

BLIT_P

Syntax

```
BLIT_P([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2],
[c])
```

Description

Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Does not copy pixels from srcG that are color c.

The defaults for the optional arguments are:
trgtG=G0

srcG=G0
sx1, sy1=srcGRB top left corner
sx2, sy2=srcGRB bottom right corner
dx1, dx2=trgtGRB top left corner
dx2, dy2=calculated so destination area is the same as source area
c=all pixel colors

discussion

blue

#link

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB-GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

bounded_function

#link

Syntax

discussion

BREAK

#link

Syntax

```
BREAK [n];
```

Description

Exits from expression local loop structure.

Example

```
FOR A FROM 1 TO 10 DO
  B:= (A+3) MOD 5
  IF B==1 THEN BREAK;
  END;
END;

If n is specified, allow to exit n loop structures.
```

discussion

#link

breakpoint

Syntax

```
breakpoint(Intg)
```

Description

Adds a breakpoint.

discussion

#link

B→R

Syntax

```
B→R(#integer)
```

Description

Transform an integer into a real number.

discussion

#link

canonical_form

Syntax

```
canonical_form(Trinom(a*x^2+b*x+c),[Var])
```

Description

Canonical form of a second degree polynomial.

Example

```
canonical_form(2*x^2-12*x+1) returns 2*(x-3)^2-17
```

discussion

#link

CAS

Syntax

```
CAS(expression) or CAS.function(...) or CAS.variable[(...)]
```

Description

Evalute an expression or variable using the CAS.
Note that outputs in numerical mode are transformed into strings or lists of expressions for symbolic matrices.

discussion

#link

CASE

Syntax

```
CASE IF test1 THEN commands1 END IF test2 THEN commands2 END ... IF
testN THEN commandsN END [DEFAULT] [commandsD] END;
```

Description

Starts a "CASE...END" branch structure.

Example

Evaluates test1. If true, executes commands1 and ends the CASE. Otherwise, evaluates test2. If true, executes commands2. Continues evaluating tests until a true is found. If no true test is found, executes commandsD, if provided.

discussion

cat

#link

Syntax

```
cat(SeqObj)
```

Description

Evaluates the arguments, then concatenates them into a string.

Example

cat("aaa","c",12*3) returns "aaac36"

discussion

CEILING

#link

Syntax

```
CEILING(value)
```

Description

Least integer greater than or equal to value.

Example

CEILING(3.2) returns 4 and CEILING(-3.2) returns -3

discussion

center

#link

Syntax

```
center(Circle)
```

Description

Returns the center of a circle

Example

```
center(circle(x^2+y2-x-y)) returns point(1/2,1/2)
```

discussion

#link

cFactor

Syntax

```
cFactor(Expr)
```

Description

Factorisation of the expression in C (on the Gauss integers if there are more than 2 variables).

Example

```
cFactor(x^2*y+y) returns (x+i)*(x-i)*y
```

discussion

#link

CHAR

Syntax

```
CHAR(list or vector) or CHAR(integer)
```

Description

Returns the string corresponding to the ASCII character codes in vector, or the single character associated with integer.

Example

```
CHAR(65) returns "A" and CHAR({82, 77, 72}) returns "RMH"
```

discussion

#link

charpoly

Syntax

```
charpoly(Mtrx,[Var])
```

Description

List of the coefficients of the characteristic polynomial of a matrix or characteristic polynomial of a matrix with the second argument as variable.

Example

```
charpoly([[1,2],[3,4]]) returns poly1[1,-5,-2]
```

discussion

#link

CHECK

Syntax

```
CHECK(n)
```

Description

Checks (selects) the corresponding symbolic definition field in the current app. The integer n must be between 0 and 9 for most apps. For Statistics 1-Var and Statistics 2-Var apps, n must be between 1 and 5.

Example

```
CHECK(3) would check F3 if the current app is Function. Then a checkmark
would appear next to F3 in Symbolic view,  F3 would be plotted in Plot
view, and evaluated in Numeric view
```

discussion

#link

chinrem

Syntax

```
chinrem([Lst||Expr,Lst||Expr],[Lst||Expr,Lst||Expr])
```

Description

Chinese remainder for polynomials written as matrices.

Example

```
chinrem([[1,2,0],[1,0,1]],[[1,1,0],[1,1,1]]) returns [[2,2,1]
[1,1,2,1,1]].
```

discussion

#link

CHISQUARE

Syntax

```
CHISQUARE(n, x)
```

Description

Chi-square probability density function. Computes the probability density of the Chi-squared distribution at x, given n degrees of freedom.

Example

```
CHISQUARE(2, 3.2) returns 0.100948258997
```

discussion

#link

CHISQUARE_CDF

Syntax

```
CHISQUARE_CDF(n, k)
```

Description

Cumulative χ² (Chi-squared) distribution function. Returns the lower-tail probability of the χ² probability density function for the value x, given n degrees of freedom.

Example

```
CHISQUARE_CDF(2, 6.1) returns 0.952641075609
```

discussion

CHISQUARE_ICDF

#link

Syntax

```
CHISQUARE_ICDF(n, p)
```

Description

Inverse cumulative χ² (Chi-squared) distribution function. Returns the value x such that the χ² lower-tail probability of x, with n degrees of freedom, is p.

Example

```
CHISQUARE_ICDF(2, 0.952641075609) returns 6.1
```

discussion

cholesky

#link

Syntax

```
cholesky(Mtrx)
```

Description

For a numerical symmetric matrix A, returns L matrix such that A=L*tran(L).

Example

```
cholesky([[3,1],[1,4]]) returns [[3*√3/3,0],[√3/3,11/3*√33/11]]
```

discussion

CHOOSE

#link

Syntax

```
CHOOSE(var, “title”, “item1”, “item2”,[…"item14"]) or
CHOOSE(var,"title",{"item1"..."itemN")
```

Description

Displays a choose box with the given title and containing items with the strings "item1", etc. If the user choose an object, var will be updated to contain the number of the selected object (an integer, 1, 2, 3, …); otherwise, stores zero in var if the user exits without choosing.

Returns true (non zero) if the user selects an object, otherwise return false (0).

discussion

chrem

#link

Syntax

```
chrem(LstIntg(a,b,c....),LstIntg(p,q,r,....))
```

Description

Chinese remainders for integers.

Example

```
chrem([2,3],[7,5]) returns [-12,35]
```

discussion

Ci

#link

Syntax

```
Ci(Expr)
```

Description

Cosine integral int(cos(t)/t,t=−∞..x).

Example

```
Ci(1.0) returns 0.337403922901
```

discussion

circle

#link

Syntax

discussion

circumcircle

#link

Syntax

```
circumcircle(Point1, Point2, Point3)
```

Description

Draws the circumcircle of a triangle; that is, the circle circumscribed about a triangle.

Example

```
circumcircle(GA, GB, GC) draws the circle circumscribed about ΔABC
```

discussion

coeff

#link

Syntax

```
coeff(Expr,[Var], [Term])
```

Description

Returns the list of coefficients of a polynomial with respect to the second argument or the coefficient of the term whose degree is Term.

Example

```
coeff(x^3+2) returns [1,0,0,2]
coeff(2*y^2-3,y,0) returns -3
```

discussion

col

#link

Syntax

```
col(Mtrx(A),Intg(n)||Interval(n1..n2))
```

Description

Returns the column n or the sequence of the columns n1...n2 of the matrix A, or optional argument of count,count_eq,count_inf,count_sup.

Example

```
col([[1,2,3],[4,5,6],[7,8,9]],1) returns [2,5,8]
```

discussion

colDim

#link

Syntax

```
coldim(Mtrx)
```

Description

Number of columns of a matrix.

Example

```
coldim([[1,2,3],[4,5,6]]) returns 3
```

discussion

collect

#link

Syntax

```
collect(Expr or {Expr1, Expr2,...,Exprn})
```

Description

Collects likes terms in a polynomial expression (or of a list of polynomial expressions).

Example

`collect(x+2*x+1-4) returns 3*x-3`

discussion

#link

COLNORM

Syntax

`COLNORM(matrix)`

Description

Column Norm. Finds the maximum value (over all columns) of the sums of the absolute values of all elements

Example

`COLNORM([[1,2],[3,4]]) returns 6`

discussion

#link

COMB

Syntax

`COMB(n, r)`

Description

Combinations. Returns the number of combinations (without regard to order) of n things taken r at a time: n!/(r!(n−r))

Example

`COMB(5,2) returns 10`

discussion

#link

comDenom

Syntax

`comDenom(Expr,[Var(var)])`

Description

Returns the expression after reduction at the same denominator: the numerator and the denominator are developed [according to the powers of the variable var].

Example

`comDenom(1/x+1/y^2+1) returns (x*y^2+x+y^2)/(x*y^2)`

discussion

#link

common_perpendicular

Syntax

```
common_perpendicular(Line(D1),Line(D2))
```

Description

Draws the common perpendicular of the lines D1 and D2.

discussion

#link

companion

Syntax

```
companion(Poly,Var)
```

Description

Companion matrix of a polynomial (an=1).

Example

```
companion(x^2+5x-7,x) returns [[0,7],[1,-5]]
```

discussion

#link

compare

Syntax

```
compare(Obj(arg1),Obj(arg2))
```

Description

Returns 1 if type(arg1)<type(arg2) or if type(arg1)=type(arg2) and arg1<arg2, else returns 0.

Example

```
compare(1,2) returns 1
```

discussion

#link

complexroot

Syntax

```
complexroot(Poly(P),Real(l),[Cplx(a)],[Cplx(b)])
```

Description

Returns the list of the vertices of the squares (side<=l) containing roots of P [inside the rectangle with opposed vertices a and b] with their mulitiplicity.

Example

```
complexroot(x^5-2*x^4+x^3+i,0.1) returns [[[(-21-12*i)/32,(-18-9*i)/32],1],[[(6-15*i)/16,(-6-21*i)/(16-16*i)],1],[[(27+18*i)/(16+16*i),(24-3*i)/16],1],[[(6+27*i)/(16+16*i),(9+6*i)/8],1],
```

`[[(-15+6*i)/(16+16*i),(-3+12*i)/16],1]`

discussion

CONCAT

#link

Syntax

`CONCAT(value1, value2, [..value16])`

Description

Concatenation. Concatenates (joins) items into a list.

Example

`CONCAT({1,2,3}, 4) returns {1,2,3,4} and CONCAT(1,2,3,4) returns {1,2,3,4}`

discussion

COND

#link

Syntax

`COND(matrix)`

Description

Condition Number. Finds the 1-norm (column norm) of a square matrix.

Example

`COND([[1,2],[3,4]]) returns 21`

discussion

conic

#link

Syntax

`conic(Expr)`

Description

Plots the graph of a conic section defined by an expression in x and y.

Example

`conic(x^2+y^2-81) draws a circle with center at (0,0) and radius of 9`

discussion

CONJ

#link

Syntax

`CONJ(x+yi)`

Description

Complex Conjugate. Reverses the sign of the imaginary part of a complex number.

Example

```
CONJ(3+4i) returns 3-4i
```

discussion

contains

#link

Syntax

```
contains((Lst(l) or Set(l)),Elem(e))
```

Description

Tests if a set contains an expression (returns the index+1 or 0).

Example

```
contains(%{0,1,2,3%},2) returns 3
```

discussion

content

#link

Syntax

```
content(Poly,[Var])
```

Description

Returns the gcd of the coefficients of the polynomial Poly.

Example

```
content(2*x^2+10*x+6) returns 2
```

discussion

CONTINUE

#link

Syntax

discussion

CONVERT

#link

Syntax

```
CONVERT(Value_Unit1, 1_Unit2)
```

Description

Converts Value in Unit1 to the corresponding value in compatible Unit2.

Example

CONVERT(20_m, 1_ft) returns 65.6167979003_ft

Alternative: 20_m ▶ _ft

discussion

convexhull

#link

Syntax

convexhull(Lst)

Description

Convex hull of a list of 2D points.

Example

convexhull(0,1,1+i,1+2i,-1-i,1-3i,-2+i) returns 1-3*i,1+2*i,-2+i,-1-i

discussion

coordinates

#link

Syntax

coordinates(Pnt or Cplx or Vect)

Description

Returns the list (resp matrix) of the abscissa and of the ordinate of a point or a vector (resp of points or vectors).

Example

coordinates(point(1+2*i)) returns [1,2]

discussion

CopyVar

#link

Syntax

CopyVar(Var(var1),Var(var2))

Description

Copy the storage without evaluation of var1 into var2.

discussion

correlation

#link

Syntax

correlation(Lst||Mtrx,[Lst])

Description

Returns the correlation of the elements of its argument.

Example

`correlation([[1,2],[1,1],[4,7]]) returns 33/(6*√31)`

discussion

COS

#link

Syntax

`COS(Value)`

Description

Returns the cosine of Value. Value is interpreted as either degrees or radians, depending on the setting of Angle Measure in Home Modes or Symbolic Setup.

Example

`in radian mode, COS(π) returns -1.`

discussion

cos2sintan

#link

Syntax

`cos2sintan(Expr)`

Description

Replaces cos(x) by sin(x)/tan(x) in the argument.

Example

`cos2sintan(cos(x)) returns sin(x)/tan(x)`

discussion

COSH

#link

Syntax

`COSH(value)`

Description

Hyperbolic cosine.

Example

`ASINH(1.17520119365) returns 1`

discussion

#link

COT

Syntax

`COT(value)`

Description

Cotangent. The Cotangent function; that is, cos(x)/sin(x).

Example

`COT(45) returns 1 in degree mode`

discussion

#link

count

Syntax

`count(Fnc(f),(Lst||Mtrx)(l),[Opt(row||col)])`

Description

Returns f(l[0])+f(l[1])+...+f(l[size(l)−1]).

Example

`count((x)->x,[2,12,45,3,7,78]) returns 147`

discussion

#link

covariance

Syntax

`covariance(Lst||Mtrx,[Lst])`

Description

Returns the covariance of the elements of its argument.

Example

`covariance([[1,2],[1,1],[4,7]]) returns 11/3`

discussion

#link

covariance_correlation

Syntax

`covariance_correlation(Lst||Mtrx,[Lst])`

Description

Returns the list of the covariance and the correlation of the elements of its argument.

Example

covariance_correlation([[1,2],[1,1],[4,7]]) returns [11/3,33/(6*√31)] discussion

#link

cpartfrac

Syntax

cpartfrac(RatFrac)

Description

Performs partial fraction decomposition in C of a fraction.

Example

cpartfrac((x)/(4-x^2)) returns 1/((x-2)*-2)+1/((x+2)*-2) discussion

#link

crationalroot

Syntax

crationalroot(Poly(P))

Description

Returns the list of complex rational roots of P without indicating the multiplicity.

Example

crationalroot(2*x^3+(-5-7*i)*x^2+(-4+14*i)*x+8-4*i) returns [(3+i)/2,2*i,1+i] discussion

#link

CROSS

Syntax

CROSS(vector1, vector2)

Description

Cross Product. Finds the cross product of vector1 with vector2.

Example

CROSS([1,2],[3,4]) returns [0, 0, -2] discussion

#link

CSC

Syntax

```
CSC(value)
```

Description

Cosecant. The Cosecant function; that is, 1/sin(x)

Example

```
CSC(90) returns 0 in degree mode
```

discussion

#link

cSolve

Syntax

```
csolve(Eq,Var)
```

Description

Returns the solutions, including comlex solutions, of Eq, for Var. If Eq is an expression, solves Eq=0.

Example

```
csolve(x^4=1,x) returns {1,-1,-i,i}
```

discussion

#link

cumSum

Syntax

```
cumSum(Lst(l)||Seq||Str)
```

Description

Returns the list (or the sequence or the string) lr where the elements are the cumulative sum of the list l:lr[k]=sum(l[j],j=0..k) (or lr=sum(l[j],j=0..k)$(k=0..size(l)-1)).

Example

```
cumSum([0,1,2,3,4]) returns [0,1,3,6,10]
```

discussion

#link

curl

Syntax

```
curl(Lst(A,B,C),Lst(x,y,z))
```

Description

Returns the curl of a vector. curl([A,B,C],[x,y,z])=[dC/dy-dB/dz,dA/dz-dC/dx,dB/dx-dA/dy].

Example

```
curl([2*x*y,x*z,y*z],[x,y,z]) returns [z-x,0,z-2*x]
```

discussion

#link

curve

Syntax

```
curve(Expr)
```

Description

Reserved word.

discussion

#link

cyan

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB−GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

#link

cyclotomic

Syntax

```
cyclotomic(Expr)
```

Description

Generates a vector representing the nth cyclotomic polynomial.

Example

```
cyclotomic(20) returns [1,0,-1,0,1,0,-1,0,1]
```

discussion

#link

cZeros

Syntax

```
cZeros(Expr,[Var]) or cZeros(ListExpr, ListVar)
```

Description

Returns the roots, including complex roots, of Expr (that is, the solution of Xpr=0) or the matrix where the lines are the solutions of the system : Expr1=0,Expr2=0....

Example

```
cZeros(x^4-1) returns [1,-1, i, -i]
```

discussion

#link

C→PX

Syntax

```
C→PX(x, y) or C→PX({x, y})
```

Description

Transform cartesian coordinates into pixel coordinates. Returns a list.

discussion

#link

DEBUG

Syntax

```
DEBUG(ProgramName(arguments))
```

discussion

#link

degree

Syntax

```
degree(Poly)
```

Description

Returns the degree of the polynomial Poly.

Example

```
degree(x^4+x) returns 3
```

discussion

#link

DELCOL

Syntax

```
DELCOL(matrixname ,column_number)
```

Description

Delete Column. Deletes the column column_number from the matrix matrixname.

discussion

#link

delcols

Syntax

```
delcols(Mtrx(A),Interval(n1..n2)||n1)
```

Description

Returns the matrix where the columns n1..n2 (or n1) of the matrix A are deleted.

Example

```
delcols([[1,2,3],[4,5,6],[7,8,9]],1..1) returns [[1,3],[4,6],[7,9]]
```

discussion

#link

DELROW

Syntax

```
DELROW(matrixname, row_number)
```

Description

Delete Row. Deletes the row row_number from the matrix matrixname.

discussion

#link

delrows

Syntax

```
delrows(Mtrx(A),Interval(n1..n2)||n1)
```

Description

Returns the matrix where the rows n1..n2 (or n1) of the matrix A are deleted.

Example

```
delrows([[1,2,3],[4,5,6],[7,8,9]],1..1) returns [[1,2,3],[7,8,9]]
```

discussion

#link

deltalist

Syntax

```
deltalist(Lst)
```

Description

Returns the list of the difference of two terms in succession.

Example

```
deltalist([1,4,8,9]) returns [3,4,1]
```

discussion

#link

denom

Syntax

`denom(a/b)`

Description

Simplified Denominator. For the integers a and b, returns the denominator of the fraction a/b after simplification.

Example

`denom(10/12) returns 6`

discussion

#link

desolve

Syntax

`desolve(Eq,[TimeVar],Var)`

Description

Solves a differential equation.

Example

`desolve(y''+y=0,y) returns G_0*cos(x)+G_1*sin(x)`

discussion

#link

DET

Syntax

`DET(matrix)`

Description

Determinant of a square matrix.

Example

`DET([[1,2],[3,4]]) returns -2`

discussion

#link

diag

Syntax

`diag(Lst(l)||Mtrx(A))`

Description

Returns either the diagonal matrix with diagonal l or the diagonal of A.

Example

`diag([1,2],[3,4]) returns [1,4]`

discussion

diff

#link

Syntax

```
diff(Expr,[Var or ListVar])
```

Description

Returns the derivative of an expression with respect to a given variable. You can use the differentiation template in the Template menu as well.

Example

```
diff(x^3-x) returns 3*x^2-1
diff(sin(x)-cos(y), x) returns cos(x)
diff(sin(x)-cos(y), y) returns sin(y)
```

discussion

DIFFERENCE

#link

Syntax

```
DIFFERENCE({list1}, ...{listN})
```

Description

Returns a list of the elements that are not common between 2 or more of the lists.

Example

```
DIFFERENCE({1,2,3},{2,4,8}) returns {1,3,4,8}
```

discussion

DIM

#link

Syntax

```
DIM(string)
```

Description

Returns the number of characters in string.

Example

```
DIM("12345") returns 5
```

discussion

DIMGROB

#link

Syntax

```
DIMGROB(G, w, h, [color]) or DIMGROB(G, w, h, list)
```

Description

Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. If the graphic is initialized using graphic data, then list is a list of integers. Each integer, as seen in base 16, describes one color every 16 bits.

Colors are in A1R5G5B5 format (ie, 1 bit for alpha channel, and 5 bits for R, G and B).

discussion

#link

DIMGROB_P

Syntax

```
DIMGROB_P(G, w, h, [color]) or DIMGROB(G, list)
```

Description

Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. If the graphic is initialized using graphic data, then list is a list of integers. Each integer, as seen in base 16, describes one color every 16 bits.

Colors are in A1R5G5B5 format (ie, 1 bit for alpha channel, and 5 bits for R, G and B).

discussion

#link

Dirac

Syntax

```
Dirac(Real)
```

Description

Function derivative of Heaviside.

Example

```
Dirac(1) returns 0
```

discussion

#link

distance

Syntax

```
distance((Pnt or Cplx),(Pnt or Cplx or Curve))
```

Description

Calculates the distance between 2 points, or a point and a curve.

Example

```
distance(0,1+i) returns √2
```

discussion

#link

distance2

Syntax

```
distance2(point1, point2) or distance2(point, curve)
```

Description

Returns the square of the distance between two points or between a point and a curve.

Example

```
distance2(1+i, 3+3i) returns 8.
if GA is the point at (0, 0) and GB is defined as plotfunc(4-x^2/4),
then distance (GA, GB) returns 12.
```

discussion

#link

distanceat

Syntax

```
distanceat(GeoObj(A),GeoObj(B),(Pnt or Cplx))
```

Description

distanceat(A,B,z0) displays at point(z0), with a legend, the distance between 2 geometrical objects.

Example

```
A:=point(0);B:=point(1+i);distanceat(A,B,(1+i)/2)) returns √2
```

discussion

#link

distanceatraw

Syntax

```
distanceatraw(Point1, Point2, Point3) or distanceatraw(Point1, Curve,
Point3)
```

Description

This command is used in Symbolic view. Similar to distanceat(), this commmand returns the distance between two points or between a point and a curve and places that measurement at the location of Point3 in the Plot view. The distance is unlabeled.

Example

```
distanceatraw(1+I, 3+3i, point(0,0)) returns 2.828…or 2√2 and places
that measure at the origin in Plot view.

If GA is the point at (0, 0) and GB is defined as plotfunc(4-x^2/4),
then distanceat(GA, GB, GA) returns 3.464… or 2√3 and places this
```

```
measure in Plot view at (0,0).

Define A:=point(0) and B:=point(1+i); then distanceatraw(A,B,(1+i)/2))
returns  √2 and places this measurement at (1/2, 1/2)
```

discussion

#link

divergence

Syntax

```
divergence(Lst(A,B,C),Lst(x,y,z))
```

Description

Returns the divergence of a vector. divergence([A,B,C],[x,y,z])=dA/dx+dB/dy+dC/dz.

Example

```
divergence([x^2+y,x+z+y,z^3+x^2],[x,y,z]) returns 2*x+3*z^2+1
```

discussion

#link

divis

Syntax

```
divis(Poly(P) or LstPoly)
```

Description

Returns the list of divisors of a polynomial.

Example

```
divis(x^2-1) returns [1,x-1,x+1,(x-1)*(x+1)]
```

discussion

#link

division_point

Syntax

```
division_point(Point1, Point2, Realk) or division_point(Cplx1, Cplx2,
Cplxk)
```

Description

For two points A and B, and a numerical factor k, returns a point C such that C−B=k*(C−A). The two points may be referenced by name or represented by a complex number.

Example

```
division_point(0,6+6*i,4) returns point (8,8)
```

discussion

divpc

#link

Syntax

```
divpc(Poly1,Poly2,Integer)
```

Description

Returns the n–degree Taylor polynomial for the quotient of 2 polynomials.

Example

```
divpc(x^4+x+2,x^2+1,5) returns the 5th-degree polynomial x^5+3*x^4-x^3-2*x^2+x+2
```

discussion

DO

#link

Syntax

```
FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO command(s) END;
```

Description

Sets variable var to start; then, for as long as this variable’s value is less than or equal to (or more than for a DOWNTO) finish, executes command(s) and adds (or substract for DOWNTO) 1 (or increment) to var.

Example

```
FOR A FROM 1 TO 10 STEP 2
  DO
    PRINT(A);
END;

will print  1 3 5 7 9
```

discussion

DOT

#link

Syntax

```
DOT(matrix1, matrix2)
```

Description

Dot Product. Finds the dot product of two arrays, matrix1 and matrix2.

Example

```
DOT([1,2],[3,4]) returns 11
```

discussion

DRAWMENU

#link

Syntax

`DRAWMENU({text...}) or DRAWMENU(text..)`

Description

Draw a menu containing the items specified

discussion

DrawSlp

#link

Syntax

`DrawSlp(Reala, Realb, Realm)`

Description

Given three real numbers m, a, b, draws a line with slope m that passes through the point (a, b).

Example

`DrawSlp(2,1,3) draws the line given by y=3x-5`

discussion

e

#link

Syntax

`e`

Description

Natural logarithm base, internally represented as 2.71828182846

discussion

EDITLIST

#link

Syntax

`EDITLIST(listname)`

Description

Starts the List Editor and displays the specified list. If used in programming, returns to the program when user presses OK (menu key).

Example

`EDITLIST(L1) edits list L1.`

discussion

EDITMAT

#link

Syntax

`EDITMAT(matrixname)`

Description

Starts the Matrix Editor and displays the specified matrix. If used in programming, returns to the program when user presses OK (menu key).

Example

`EDITMAT(M1) edits matrix M1.`

discussion

egcd

#link

Syntax

`egcd((Poly or Lst),(Poly or Lst),[Var])`

Description

Returns the extended greatest common divisor of 2 polynomials.

Example

`egcd((x-1)^2,x^3-1) returns [-x-2,1,3*x-3]`

discussion

Ei

#link

Syntax

`Ei(Expr)`

Description

Exponential integral int(exp(t)/t,t=−∞..x)

Example

`Ei(1.0) returns 1.89511781636`

discussion

EIGENVAL

#link

Syntax

`EIGENVAL(matrix)`

Description

Displays the eigenvalues in vector form for matrix.

Example

```
EIGENVAL([[1,2],[3,4]]) returns [5.37228132327 -.372281323269]
```

discussion

#link

eigenvals

Syntax

```
eigenvals(Mtrx)
```

Description

Returns the sequence of the (calculable) eigenvalues of a matrix.

Example

```
eigenvals([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) returns 3,-3,-3
```

discussion

#link

eigenvects

Syntax

```
eigenvects(Mtrx)
```

Description

Computes the eigenvectors of a diagonalizable matrix.

Example

```
eigenvects([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) returns [[1,-3,-3],
[-2,0,-3],[1,3,-3]]
```

discussion

#link

EIGENVV

Syntax

```
EIGENVV(matrix)
```

Description

Eigenvectors and Eigenvalues for a square matrix. Displays a list of two arrays. The first contains the eigenvectors and the second contains the eigenvalues.

Example

```
EIGENVV([[1,2],[3,4]]) returns { [[eigenvectors]],[[eigenvalues]] }
```

discussion

#link

eigVc

Syntax

```
eigVc(Mtrx)
```

Description

Computes the eigenvectors of a diagonalizable matrix.

Example

eigVc([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) returns [[1,-3,-3],[-2,0,-3],[1,3,-3]]

discussion

eigVl

#link

Syntax

```
eigVl(Mtrx(A))
```

Description

Returns the Jordan matrix associated to A when the eigenvalues are calculable.

Example

eigVl([[4,1],[-4,0]]) returns [[2,1],[0,2]]

discussion

element

#link

Syntax

```
element(object, real) or element(real1..real2)
```

Description

Creates a point on a geometric object whose abscissa is a given value or creates a real value on a given interval.

Example

element(plotfunc(x^2),−2) creates a point on the graph of y = x^2. Initially, this point will appear at (−2,4). You can move the point, but it will always remain on the graph of its function.

element(0..5) creates a value of 2.5 initially. Tapping on this value and pressing Enter enables you to press a cursor key to increase or decrease the value in a manner similar to a slider bar. Press Enter again to close the slider bar. The value you set can be used as a coefficient in a function you subsequently plot.

discussion

ellipse

#link

Syntax

```
ellipse(Point1, Point2, Point3) or ellipse(Point1, Point2, Realk)
```

Description

Draws an ellipse, given the foci and either a point on the ellipse or a scalar that is one half the constant sum of the distances from a point on the ellipse to each of the foci.

Example

```
ellipse(GA, GB, GC) draws the ellipse whose foci are points A and B and
which passes through point C.

ellipse(GA, GB, 3) draws an ellipse whose foci are points A and B. For
any point P on the ellipse, AP+BP=6.
```

discussion

ELSE

#link

Syntax

```
IF test THEN command(s) [ELSE commands] END;
```

Description

Evaluates test. If test is true (non 0), executes command(s); otherwise, executes the comands in the ELSE clause nothing happens.

Example

```
IF A<1
  THEN PRINT("A IS SMALLER THAN 1");
  ELSE PRINT("A IS LARGER THAN 1");
END;
```

discussion

END

#link

Syntax

discussion

equation

#link

Syntax

```
equation(curve) or equation(point)
```

Description

Returns the Cartesian equation of a curve in x and y, or the Cartesian coordinates of a point.

Example

equation(line(1-i,i)) returns y=-2*x+1

If GA is the point at (0, 0), GB is the point at (1, 0), and GC is defined as circle(GA, GB-GA), then equation(GC) returns x^2 + y^2 =1.

discussion

equilateral_triangle

#link

Syntax

```
equilateral_triangle(Point1, Point2, [Var])
```

Description

Draws an equilateral triangle defined by one of its sides; that is, by two consecutive vertices. The third point is calculated automatically, but is not defined symbolically. If a lowercase variable is added as a third argument, then the third point is labeled with the variable name and the coordinates of the third point are stored in that variable. The orientation of the triangle is counterclockwise from the first point.

Example

equilateral_triangle(point(0,0), point(1,0)) draws the equilateral trangle through the points at (0,0), (1,0), and (1/2, √3/2).

discussion

erf

#link

Syntax

```
erf(Real(x0))
```

Description

Returns the approximate value of 2/√π*int(exp(−t^2),t,0,x0)

Example

erf(1) returns 0.84270079295

discussion

erfc

#link

Syntax

```
erfc(Real(x0))
```

Description

Returns the approximate value of 2/√π*int(exp(−t^2),t,x0,∞).

Example

```
erfc(1) returns 0.15729920705
```

discussion

#link

euler

Syntax

```
euler(x);
```

Description

Euler’s phi (or totient) function. Takes a positive integer x and returns the number of positive integers less than or equal to x that are coprime to x.

Example

```
euler(6) returns 2
```

discussion

#link

EVAL

Syntax

```
EVAL(expression)
```

Description

Evaluates the expression. Usefull in programs where parameters are passed non evaluated with QUOTE

discussion

#link

evalc

Syntax

```
evalc(Expr)
```

Description

Returns a complex expression simplified with the format real+i*imag

Example

```
evalc(1/(x+y*i)) returns x/(x^2+y^2)+(i)*(-y)/(x^2+y^2)
```

discussion

#link

evalf

Syntax

```
evalf(Expr,[Int])
```

Description

Numerical evaluation of the first argument (we can give the number of digits as second argument).
approx(expression) works also and does the same thing.

Example

```
evalf(2/3) returns 0.666666666667
```

discussion

even

#link

Syntax

```
even(Intg(n))
```

Description

Returns 1 if the integer is even, else returns 0.

Example

```
even(6) returns 1
```

discussion

exact

#link

Syntax

```
exact(Expr)
```

Description

Converts the expression to a rational or real expression.

Example

```
exact(1.4141) returns 14141/10000
```

discussion

exbisector

#link

Syntax

```
exbisector(Point1, Point2, Point3)
```

Description

Given three points that define a triangle, creates the bisector of the exterior angles of the triangle whose common vertex is at the first point. The triangle does not have to be drawn in the Plot view.

Example

```
exbisector(GA, GB, GC) draws the bisector of the exterior angles of ΔABC
whose common vertex is at point A.

exbisector(0,-4i,4) draws the line given by y=x
```

discussion

excircle

#link

Syntax

```
excircle(Point1, Point2, Point3)
```

Description

excircle(A,B,C) draws the A-excircle of the ABC triangle.

Draws one of the excircles of a triangle, a circle tangent to one side of the triangle and also tangent to the extensions of the other two sides.

Example

```
excircle(GA, GB, GC) draws the circle tangent to BC and to the rays AB
and AC.
```

discussion

EXECON

#link

Syntax

```
EXECON("expression with &", lists or matrices)
```

Description

Returns a matrix or list composed of the result of the evaluation of the expression after replacement of & by each item in the input.

Example

```
EXECON("&1+1", {1,2,3}) returns {2,3,4}

If EXECON has only 1 list or matrix input, using & followed by a number
A (between 1 and 9) will replace &A by the element i+A-1 of the input.
Example: EXECON("&2-&1", { 1, 4, 3, 5}") returns {3, -1, 2} - the
difference between 2 successive elements.

If EXECON has 2 or more lists or matrices input, using & followed by a
number A (between 1 and 9) will replace &1 by the element from the Ath
input.
Example: EXECON("&1+&2", {1,2,3},{4,5,6}) returns {5,7,9}
```

If EXECON has 2 or more lists or matrices as input, using & followed by 2 numbers A and B (between 1 and 9) will reaplace &AB by the element i+B-1 of the Ath input.
Example: `EXECON("&22-&1", {1,2,3},{4,5,6,7}) returns {4,4,4}`

Note that for matrix input, the elements are treated as if the matrix was a vector.

discussion

EXP

#link

Syntax

`EXP(value)`

Description

The natural exponential. This is more accurate than e^x due to limitations of the power function.

discussion

exp2pow

#link

Syntax

`exp2pow(Expr)`

Description

Transforms an expression of the form exp(n*ln(x)) to x^n.

Example

```
exp2pow(exp(3*ln(x))) returns x^3
```

discussion

exp2trig

#link

Syntax

`exp2trig(Expr)`

Description

Transforms the complex exponential into sine and cosine.

Example

```
exp2trig(exp-(i*x)) returns cos(x)-i*sin(x)
```

discussion

expand

#link

Syntax

```
expand(Expr )
```

Description

Full distribution of multiplication and division over addition and subtraction.

Example

```
expand((x+y)*(z+1)) returns y*z+x*z+y+x
```

discussion

expexpand

#link

Syntax

```
expexpand(Expr)
```

Description

Expands exponentials usinng the identity exp(a*f(x))=(exp(f(x)))^a.

Example

```
expexpand(exp(3*x)) returns exp(x)^3
```

discussion

EXPM1

#link

Syntax

```
EXPM1(value)
```

Description

Exponent minus 1. This is more accurate than EXP when x is close to zero.

Example

```
EXPM1(.23) returns .258600009929
```

discussion

exponential_regression

#link

Syntax

```
exponential_regression(Lst||Mtrx(A),[Lst])
```

Description

Returns the coefficients (a,b) of y=b*a^x : it is the best exponential that approximates the points where the coordinates are the rows of A (or the 2 lists).

Example

```
exponential_regression([[1.0,2.0],[0.0,1.0],[4.0,7.0]]) returns
```

```
1.60092225473,1.10008339351
```

discussion

EXPORT

#link

Syntax

```
Variable declaration: EXPORT var_1[:=value][, more variables]; forward function declaration: EXPORT function(params); Normal function declaration: or EXPORT function[(params)] BEGIN END;
```

Description

In a program, declares a list of exported variable or an exported function.

discussion

EXPR

#link

Syntax

```
EXPR(string)
```

Description

Parses string into a number or expression.

Example

```
EXPR("2+3") returns 5
```

discussion

extract_measure

#link

Syntax

```
extract_measure(Var)
```

Description

Returns the definition of a geometric object. For a point, that definition consists of the coordinates of the point. For other objects, the definition mirrors their definition in Symbolic view, with the coordinates of their defining points supplied.

Example

```
extract_measure(angleatraw(0,1,1+i,1)
extract_measure(distanceatraw(0,1+i,(1+i)/2)) returns √2
```

discussion

ezgcd

#link

Syntax

```
ezgcd(Poly,Poly)
```

Description

Returns the GCD of 2 polynomials with at least 2 variables, with the ezgcd algorithm.

Example

```
ezgcd(x^2-+3*x-xy-3*y,x^2-y^2) returns x-y
```

discussion

f2nd

#link

Syntax

```
f2nd(Frac or RatFrac)
```

Description

Returns the list built with the numerator and the denominator of the simplified fraction.

Example

```
f2nd(42/12) returns [7,2]
```

discussion

factor

#link

Syntax

```
factor(Expr)
```

Description

Factorizes a polynomial.

Example

```
factor(x^4-1) returns (x-1)*(x+1)*(x^2+1)
```

discussion

factor_xn

#link

Syntax

```
factor_xn(Poly)
```

Description

Factorizes x^n in P\the polynomial Poly (n=degree of polynomial P).

Example

```
factor_xn(x^4-1) returns x^4*(1-x^-4)
```

discussion

factorial

#link

Syntax

```
factorial(Intg(n)|| Real(a) )
```

Description

factorial(n)=n!. For non-integers, factorial(a)=a! = G(a + 1). This calculates the Gamma function.

Example

```
factorial(4) returns 24
```

discussion

factors

#link

Syntax

```
factors(Poly) or factors({Poly1, Poly2, ..., Polyn})
```

Description

Returns the list of prime factors of a polynomial; each factor followed by its multiplicity.

Example

```
factors(x^4-1) returns [x-1,1,x+1,1,x^2+1,1]
```

discussion

fcoeff

#link

Syntax

```
fcoeff(Root1, Oder1, Root2, Order2, ..., Rootn, Ordern)
```

Description

Returns the polynomial described by a list of roots, each followed by its order.

Example

```
fcoeff([1,2,0,1,3,-1]) returns ((x-1)^2)*x*(x-3)^-1
```

discussion

fft

#link

Syntax

```
fft(Vect or (Vect(L),Intg(a),Intg(p))
```

Description

Fast Fourier Transform in R or in the field Z/pZ, with a as primitive n–th root of 1 (n=size(L)).

Example

```
fft([1,2,3,4,0,0,0,0]) returns [10.0,-0.414213562373-7.24264068712*(i),-2.0+2.0*i,2.41421356237-1.24264068712*i,-2.0,2.41421356237+1.24264068712*i,-2.0-2.0*i]
```

discussion

FILLPOLY

#link

Syntax

```
FILLPOLY([G], {coordinates...} or [Coordinates], Color, [Alpha])
```

Description

Fills the polygon specified by the provided Cartésian coordinates using the color provided.
If Alpha (0 to 255) is provided, the polygon is drawn with trensparency.

Example

```
FILLPOLY([(0,0),(1,1),(2,0),(3,-1),(2,-2)], #FF, 128)
```

discussion

FILLPOLY_P

#link

Syntax

```
FILLPOLY_P([G], {coordinates...} or [Coordinates], Color, [Alpha])
```

Description

Fills the polygon specified by the provided pixel coordinates using the color provided.
If Alpha (0 to 255) is provided, the polygon is drawn with trensparency.

Example

```
FILLPOLY_P([(20,20),(120,120),(150,20),(180,150),(50,100)], #FF, 128)
```

discussion

FISHER

#link

Syntax

```
FISHER(n, d, x)
```

Description

F (Fisher or Fisher-Snedecor) probability density function. Computes the probability density at the value x, given numerator n and denominator d degrees of freedom.

Example

```
FISHER(5, 5, 2) returns 0.158080231095
```

discussion

#link

FISHER_CDF

Syntax

```
FISHER_CDF(n, d, x)
```

Description

Cumulative F (Fisher or Fisher-Snedecor) distribution function. Returns the lower-tail probability of the F probability density function for the value x, given numerator n and denominator d degrees of freedom.

Example

```
FISHER_CDF(5, 5, 2) returns 0.76748868087
```

discussion

#link

FISHER_ICDF

Syntax

```
FISHER_ICDF(n, d, p)
```

Description

Inverse cumulative F (Fisher or Fisher-Snedecor) distribution function. Returns the value x such that the F lower-tail probability of x, with numerator, n and denominator, d degrees of freedom, is p.

Example

```
FISHER_ICDF(5, 5, 0.76748868087) returns 2
```

discussion

#link

FLOOR

Syntax

```
FLOOR(value)
```

Description

Greatest integer less than or equal to value.

Example

```
FLOOR(-3.2) returns -4
```

discussion

#link

fMax

Syntax

```
fMax(Expr,[Var])
```

Description

Returns the abscissa of the maximum of the expression.

Example

```
fMax(-x^2+2*x+1,x) returns 1
```

discussion

fMin

#link

Syntax

```
fMin(Expr,[Var])
```

Description

Returns the abscissa of the minimum of the expression.

Example

```
fMin(x^2-2*x+1,x) returns 1
```

discussion

FNROOT

#link

Syntax

```
FNROOT(expression, variable, [guess], [guess2])
```

Description

Function root-finder (like the Solve app). Finds the value for variable at which expression most nearly evaluates to zero. Uses guess as initial estimate.

Example

```
FNROOT(M*9.8/600-1, M, 1) returns 61.2244897959
```

discussion

FOR

#link

Syntax

```
FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO command(s)
END;
```

Description

Sets variable var to start; then, for as long as this variable's value is less than or equal to (or more than for a DOWNTO) finish, executes command(s) and adds (or substract for DOWNTO) 1 (or increment) to var.

Example

```
FOR A FROM 1 TO 10 STEP 2
  DO
    PRINT(A);
END;

will print  1 3 5 7 9
```

discussion

#link

format

Syntax

```
format(Real,Str("f4"||"s5"||"e6"))
```

Description

Transforms the real into a string with the indicated format (f=float,s=scientific,e=engineering).

Example

```
format(9.3456,"s3") returns 9.35
```

discussion

#link

FP

Syntax

```
FP(value)
```

Description

Returns the Fractional part of value.

Example

```
FP (23.2) returns .2
```

discussion

#link

fracmod

Syntax

```
fracmod(Expr(Xpr),Intg(n))
```

Description

Returns the fraction a/b such as a/b=Xpr mod n, $-\sqrt{n}/2<a<=\sqrt{n}/2$ and $0<=b<\sqrt{n}/2$

Example

```
fracmod(41,121) returns 2/3
```

discussion

#link

FREEZE

Syntax

```
FREEZE
```

Description

Prevents the screen from being redrawn after the program ends. Leaves the modified display on the screen for the user to see.

discussion

#link

froot

Syntax

```
froot(RatPoly(F))
```

Description

Returns the list of roots and poles of F with their mulitiplicity.

Example

```
froot((x^5-2*x^4+x^3)/(x-3)) returns [0,3,1,2,3,-1]
```

discussion

#link

fsolve

Syntax

```
fsolve(Expr,Var,[Guess or Interval],[Method])
```

Description

Numerical solution of an equation or a system of equations.

Example

```
fsolve(cos(x)=x,x,-1..1) returns [0.739085133215]
```

discussion

#link

function_diff

Syntax

```
function_diff(Fnc(f))
```

Description

Returns the derivative function of the function f.

Example

```
function_diff(sin) returns  (`x`)->cos(`x`)
```

discussion

Gamma

#link

Syntax

```
Gamma(Real(x0))
```

Description

Calculus of Gamma at a point x0 (Gamma(n+1)=n! for n integer).

Example

```
Gamma(5) returns 24
```

discussion

gauss

#link

Syntax

```
gauss(Expr,VectVar)
```

Description

Splits a quadratic form as a sum/difference of square.

Example

```
gauss(x^2+2*a*x*y,[x,y]) returns (a*y+x)^2+(-y^2)*a^2
```

discussion

gbasis

#link

Syntax

```
gbasis(ListPoly, ListVar)
```

Description

Returns the Groebner basis of the ideal spanned by the list of polynomials.

Example

```
gbasis({x^2-y^3,x+y^2},{x,y}) returns [y^4-y^3,x+y^2]
```

discussion

gcd

#link

Syntax

```
gcd(Poly1, Poly2)
```

Description

Returns the greatest common divisor of 2 polynomials of several variables. Can also be used as integer gcd.

Example

```
gcd(x^2-4,x^2-5*x+6) returns x-2
gcd(45,30) returns 15
```

discussion

GETBASE

#link

Syntax

```
GETBASE(#integer)
```

Description

Returns the base used for display for this integer.

0: system
1: bin
2: oct
3: dec
4: hex

discussion

GETBITS

#link

Syntax

```
GETBITS(#integer)
```

Description

Returns the number of bits used for calculations with this integer.

discussion

GETKEY

#link

Syntax

```
GETKEY
```

Description

Returns the ID of the first key in the keyboard buffer, or -1 if no key was pressed since the last call to GETKEY. Key IDs are integers from 0 to 50, numbered from top left (key 0) to bottom right (key 50).

0= Apps
1= Symb
2= Up
3= Help
4= Esc

5= Home
6= Plot
7= Left
8= Right
9= View
10= Cas
11= Num
12= Down
13= Menu

After that, the keys are number from top left (14= Vars) to bottom right (50= +)discussion

GETPIX

#link

Syntax

```
GETPIX([G], x, y)
```

Description

Returns the color of the pixel of G with coordinates (x,y).

discussion

GETPIX_P

#link

Syntax

```
GETPIX_P([G], x, y)
```

Description

Returns the color of the pixel of G with coordinates (x,y).

discussion

GF

#link

Syntax

```
GF(Intg(p), Intg(n))
```

Description

Creates a Galois Field of characteristic p with p^n elements.

Example

```
GF(5,9) returns GF(5,k^9-k^8+2*k^7+2*k^5-k^2+2*k-2,[k,K,g],undef)
```

discussion

grad

#link

Syntax

```
grad(Expr, ListVars)
```

Description

Returns the gradient of the expression Expr.

Example

```
grad(2*x^2*y-x*z^3,[x,y,z]) returns [2*2*x*y-z^3,2*x^2,-x*3*z^2]
```

discussion

#link

gramschmidt

Syntax

```
gramschmidt(Basis(B),ScalarProd(Sp))
```

Description

Returns an orthonormal base of E of base B for the scalar product Sp.

Example

```
gramschmidt([1,1+x],(p,q)->integrate(p*q,x,-1,1)) returns [1/(√2),(1+x-1)/(√6)/3]
```

discussion

#link

greduce

Syntax

```
greduce(Poly, ListPoly, ListVar)
```

Description

Returns the remainder of the division of a polynomial by a Groebner basis.

Example

```
greduce(x*y-1,{x^2-y^2,2*x*y-y^2,y^3},{x,y}) returns (1/2)*y^2-1
```

discussion

#link

green

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB-GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

#link

GROBH

Syntax

`GROBH(G)`

Description

Returns the height of G.

discussion

#link

GROBH_P

Syntax

`GROBH_P(G)`

Description

Returns the height of G.

discussion

#link

GROBW

Syntax

`GROBW(G)`

Description

Returns the width of G.

discussion

#link

GROBW_P

Syntax

`GROBW_P(G)`

Description

Returns the width of G.

discussion

#link

half_line

Syntax

`half_line(Point1, Point2)`

Description
Given 2 points, draws a ray from the first point through the second point.

Example
`half_line(0, 1+i) draws a ray starting at the origin and passing through the point at (1,1)`

discussion

#link

halftan

Syntax
`halftan(Expr)`

Description
Transforms sin(x), cos(x) and tan(x) as a function of tan(x/2).

Example
`halftan(sin(x)) returns (2*tan(x/2))/((tan(x/2))^2+1)`

discussion

#link

halftan_hyp2exp

Syntax
`halftan_hyp2exp(Expr)`

Description
Transforms the trigonometric functions in tan(x/2) and hyperbolic functions into expontials.

Example
`halftan_hyp2exp(sin(x)+sinh(x)) returns (2*tan(x/2)/((tan(x/2))^2+1)+(exp(x)-1/exp(x))/2`

discussion

#link

halt

Syntax
`halt(NULL)`

Description
Puts a program in step-by-step debug mode.

discussion

#link

hamdist

Syntax

`hamdist(Intg,Intg)`

Description

Bit Hamming distance.

Example

`hamdist(0x12,0x38) returns 3`

discussion

harmonic_conjugate

#link

Syntax

`harmonic_conjugate(Line or Pnt,Line or Pnt,Line or Pnt)`

Description

Returns the harmonic conjugate of 3 points or of 3 parallel or concurrent lines or the line of conjugates of a point in respect to 2 lines.

discussion

harmonic_division

#link

Syntax

`harmonic_division(Pnt or Line,Pnt or Line,Pnt or Line,Var)`

Description

Returns the 4 points (resp lines) and affects the last argument, such as the 4 points (resp lines) are in a harmonic division.

discussion

has

#link

Syntax

`has(Expr,Var)`

Description

Checks if a variable is in an expression.

Example

`has(x+y,x) returns 1`

discussion

head

#link

Syntax

```
head(Vect or Seq or Str)
```

Description

Shows the first element of a vector or a sequence or a string.

Example

```
head(1,2,3) returns 1
```

discussion

#link

Heaviside

Syntax

```
Heaviside(Real)
```

Description

Function equal to 0 if x<0 and 1 if x>=0

Example

```
Heaviside(1) returns 1
```

discussion

#link

hermite

Syntax

```
hermite(Integer)
```

Description

Returns nth Hermite polynomial.

Example

```
hermite(3) returns 8*x^3-12*x
```

discussion

#link

hessenberg

Syntax

```
hessenberg(Mtrx(A))
```

Description

Matrix reduction to Hessenberg form. Returns [P,B] such that B=inv(P)*A*P.

Example

```
hessenberg([[1,2,3],[4,5,6],[7,8,1]]) returns [[[1,0,0],[0,4/7,1],
[0,1,0]],[[1,29/7,2],[7,39/7,8],[0,278/49,3/7]]]
```

discussion

hessian

#link

Syntax

`hessian(Expr,LstVar)`

Description

Returns the hessian matrix of the expression Expr.

Example

`hessian(2*x^2*y-x*z,[x,y,z]) returns [[4*y,4*x,-1],[2*2*x,0,0],[-1,0,0]]`

discussion

hexagon

#link

Syntax

`hexagon(Point1, Point2, [Var1, Var2, Var3, Var4])`

Description

Draws a regular hexagon defined by one of its sides; that is, by two consecutive vertices. The remaining points are calculated automatically, but are not defined symbolically. The orientation of the hexagon is counterclockwise from the first point.

Example

`hexagon(0,6) draws a regular hexagon whose first two vertices are at (0, 0) and (6, 0).`

`hexagon(0,6, a, b, c, d) draws a regular hexagon whose first two vertices are at (0, 0) and (6, 0)] labels the other four vertices a, b, c, and d, and stores the coordinates into the CAS variables a, b, c, and d. You do not have to define variables for all four remaining points, but the coordinates are stored in order. For example, hexagon(0,6, a) stores just the third point into the CAS variable a.`

discussion

hilbert

#link

Syntax

`hilbert(Intg(n))`

Description

Returns the order n Hilbert matrix: Hjk=1/(j+k+1) j,k=1..n

Example

```
hilbert(4) returns [[1,1/2,1/3,1/4],[1/2,1/3,1/4,1/5],[1/3,1/4,1/5,1/6],
[1/4,1/5,1/6,1/7]]
```

discussion

#link

→HMS

Syntax

```
→HMS(value)
```

Description

Decimal to hours-minutes-seconds.
Change the way a number is displayed to HMS format.
→HMS(8.5) returns 8°3

discussion

#link

HMS→

Syntax

```
HMS→(value)
```

Description

Hours-minutes-seconds to decimal.
Forces a number to be displayed in decimal format if it was previously displayed in DMS format
HMS→(8°30) returns 8.5

discussion

#link

homothety

Syntax

```
homothety(Point, Realk, Object)
```

Description

Dilates a geometric object, with respect to a center point, by a scale factor.

Example

```
homothety(GA, 2, GB) creates a dilation centered at point A that has a
scale factor of 2. Each point P on geometric object B has its image P’
on ray AP such that AP’=2AP.

homothety(point(0,0),1/3,point(9,9)) creates an image point at (3,3)
```

discussion

#link

hyp2exp

Syntax

`hyp2exp(ExprHyperb)`

Description

Transforms the hyperbolic functions with the exponential function.

Example

`hyp2exp(cosh(x)) returns (exp(x)+1/exp(x))/2`

discussion

hyperbola

#link

Syntax

`hyperbola(Point1, Point2, Point3) or hyperbola(Point1, Point2, Realk)`

Description

Draws a hyperbola, given the foci and either a point on the hyperbola or a scalar that is one half the constant difference of the distances from a point on the hyperbola to each of the foci.

Example

```
hyperbola(GA, GB, GC) draws the hyperbola whose foci are points A and B
and which passes through point C.

hyperbola(GA, GB, 3) draws a hyperbola whose foci are points A and B.
For any point P on the hyperbola, |AP-BP|=6.
```

discussion

iabcuv

#link

Syntax

`iabcuv(Intg(a),Intg(b),Intg(c))`

Description

Returns [u,v] such as au+bv=c for 3 integers a,b,c

Example

`iabcuv(21,28,7) returns [-1,1]`

discussion

ibasis

#link

Syntax

`ibasis(Lst(Vect,..,Vect),Lst(Vect,..,Vect))`

Description

Basis of the intersection of two vector spaces.

Example

```
ibasis([[1,0,0],[0,1,0]],[[1,1,1],[0,0,1]]) returns [[-1,-1,0]]
```

discussion

#link

ibpdv

Syntax

```
ibpdv(Expr1,Expr2,[Var],[Real1],[Real2])
```

Description

Integration by parts of Expr1=u(Var)*v'(Var) with Expr2= v'(Var) (or 0) as 2nd argument. You can specify a variable of integration and also the bounds of integration (Real1 and Real2).

Example

```
ibpdv(x*ln(x),1) returns (-1/4)*x^2+(1/2)*(x^2)*ln(x)
```

discussion

#link

ibpu

Syntax

```
ibpu(Expr1,Expr2,[Var],[Real1],[Real2])
```

Description

Integration by parts of Expr1=u(Var)*v'(Var) with Expr2= u(Var) (or 0) as 2nd argument. You can specify a variable of integration and also the bounds of integration (Real1 and Real2).

Example

```
ibpu(ln(x),ln(x),x,1,3) returns [3*ln(3),-1]
```

discussion

#link

ichinrem

Syntax

```
ichinrem([a,p],[b,q]))
```

Description

Integer Chinese Remainder Theorem for two equations. Takes two lists [a, p] and [b, q] and returns a list of two integers, [r, n], such that x≡r mod n. In this case, x is such that x≡a mod p and x≡b mod q; also, n=p*q.

Example

```
ichinrem([2, 7], [3, 5]) returns [-12, 35]
```

discussion

icontent

#link

Syntax

```
icontent(Poly,[Var])
```

Description

Returns the GCD of the integer coefficients of a polynomial.

Example

```
icontent(24x^3+6x^2-12x+18) returns 6
```

discussion

id

#link

Syntax

```
id(Seq)
```

Description

The name of the identity function (R^n-> R^n)

Example

```
id(1,2,3) returns 1,2,3
```

discussion

IDENMAT

#link

Syntax

```
IDENMAT(n)
```

Description

Identity matrix. Creates a square matrix of dimension n x n whose diagonal elements are 1 and off-diagonal elements are zero.

Example

```
IDENMAT(2) returns [[1,0],[0,1]]
```

discussion

identity

#link

Syntax

```
identity(Intg(n))
```

Description

Returns the identity matrix of specified dimension n.

Example

```
identity(3) returns [[1,0,0],[0,1,0],[0,0,1]]
```

discussion

#link

idivis

Syntax

```
idivis(a)
```

Description

Integer divisors. Returns a list of all the factors of the integer a.

Example

```
idivis(12) returns [1, 2, 3, 4, 6, 12]
```

discussion

#link

iegcd

Syntax

```
iegcd(a,b)
```

Description

Extended greatest common divisor for two integers. Returns [u,v,igcd(a,b)] such that a*u+b*v=igcd(a,b).

Example

```
iegcd(14, 21) returns [-1, 1, 7]
```

discussion

#link

IF

Syntax

```
IF test THEN command(s) [ELSE commands] END;
```

Description

Evaluates test. If test is true (non 0), executes command(s); otherwise, executes the comands in the ELSE clause nothing happens.

Example

```
IF A<1
  THEN PRINT("A IS SMALLER THAN 1");
  ELSE PRINT("A IS LARGER THAN 1");
```

```
END;
```

discussion

#link

ifactor

Syntax

```
ifactor(a)
```

Description

Prime factorization. Returns the prime factorization of the integer a as a product. Can be used with STO▶.

Example

```
ifactor(150) returns  2*3*5^2
```

discussion

#link

ifactors

Syntax

```
ifactors(a)
```

Description

Prime factors. Similar to ifactor, but returns a list of the factors of the integer a with their multiplicities.

Example

```
ifactors(150) returns [2, 1, 3, 1, 5, 2]
```

discussion

#link

IFERR

Syntax

```
IFERR commands1 THEN commands2 [ELSE commands3] END;
```

Description

Executes sequence of commands1. If an error occurs during execution of commands1, execute sequence of commands2. Otherwise, execute sequence of commands3.

Many conditions are automatically recognized by the HP Prime as error conditions and are automatically treated as errors in programs. This command facilitates error-trapping of such errors.

discussion

#link

ifft

Syntax

```
ifft(Vect)
```

Description

Inverse Fast Fourier Transform.

Example

```
ifft([100.0,-52.2842712475+6*i,-8.0*i,4.28427124746-6*i,4.0,4.28427124746+6*i,8*i,-52.2842712475-6*i]) returns [0.99999999999,3.99999999999,10.0,20.0,25.0,24.0,16.0,-6.39843733552e-12]
```

discussion

IFTE

#link

Syntax

```
IFTE(Expr, Trueclause, Falseclause)
```

Description

If...Then...Else...
If Expr evaluates true (1), evaluates Trueclause; if not, evaluates Falseclause.

Example

```
IFTE(2<3, 5-1, 2+7) returns 4
```

discussion

igcd

#link

Syntax

```
igcd(a, b)
```

Description

Greatest common divisor. Returns the integer that is the greatest common divisor of the integers a and b.

Example

```
igcd(24, 36) returns 12
```

discussion

ihermite

#link

Syntax

```
ihermite(Mtrx(A))
```

Description

Hermite normal form of a matrix with coefficients in Z: returns U,B such that U is invertible in Z, B upper triangular and B=U*A

Example

`ihermite([[1,2,3],[4,5,6],[7,8,9]]) returns [[-3,1,0],[4,-1,0],[-1,2,-1]],[[1,-1,-3],[0,3,6],[0,0,0]]`

discussion

ilaplace

#link

Syntax

`ilaplace(Expr,[Var],[IlapVar])`

Description

Inverse Laplace transform of a rational fraction.

Example

`ilaplace(1/(x^2+1)^2) returns (-x)*cos(x)/2+sin(x)/2`

discussion

IM

#link

Syntax

`IM(x+yi)`

Description

Imaginary Part. Returns the imaginary part of a complex number.

Example

`IM(3+4i)  returns 4`

discussion

incircle

#link

Syntax

`incircle(Point1, Point2, Point3)`

Description

Draws the incircle of a triangle, the circle tangent to all three sides of the triangle.

Example

`incircle(GA, GB, GC) draws the incircle of ΔABC.`

discussion

INPUT

#link

Syntax

```
INPUT(var,["title"], ["label"], ["help"], [reset])
```

Description

or INPUT({vars},["title"], [{"labels"}], [{"help"}], [{reset}])

Starts a dialog box with header title and one field named label (with value default), displaying help at the bottom. The dialog box includes CANCEL and OK menu keys. If the user presses the OK menu key, the variable var is updated and 1 is returned. If the user presses the CANCL menu key, var is not updated and 0 is returned.

discussion

INSTRING

#link

Syntax

```
INSTRING(string1, string2)
```

Description

Returns the index of the first occurrence of string2 in string1. Returns 0 if str2 is not present in str1. Note that the first character in a string is in position 1.

Example

```
INSTRING("vanilla", "van") returns 1
INSTRING("banana","na") returns 3
INSTRING("ab","abc") returns 0
```

discussion

int

#link

Syntax

```
int(Expr,[Var],[Real1,Real2])
```

Description

Integral (definite or indefinite). You can specify a variable of integration as well as the bounds ofr integration. You can use the integration template in the Template menu as well.

Example

```
int(1/x) returns ln(abs(x))
int(sin(x),x,0,π) returns 2
```

discussion

inter

#link

Syntax

```
inter(Curve1, Curve2)
```

Description

Returns the intersections of two curves as a vector.

Example

```
inter(8-x^2/6, x/2-1) returns [[6, 2] [-9, -11/2]], indicating that
there are two intersections-one at (6,2) and the other at (-9,-5.5).
```

discussion

INTERSECT

#link

Syntax

```
INTERSECT({list1}, ...{listN})
```

Description

Returns a list of the elements common to all the lists.

Example

```
INTERSECT({1,2,3},{2,4,8}) returns {2}
```

discussion

interval2center

#link

Syntax

```
interval2center(Interval or Real)
```

Description

Returns the center of the interval or the object.

Example

```
interval2center(2..5) returns 7/2
```

discussion

inv

#link

Syntax

```
inv(Expr||Mtrx)
```

Description

Returns the inverse of an expression or matrix.

Example

```
inv(9/5) returns 5/9
```

discussion

inversion

#link

Syntax

```
inversion(Point1, Realk, Point2)
```

Description

Draws the inversion of a point, with respect to another point, by a scale factor.

Example

inversion(GA, 3, GB) draws point C on line AB such that AB*AC=3. In this case, point A is the center of the inversion and the scale factor is 3. Point B is the point whose inversion is created.

In general, the inversion of point A through center C, with scale factor k, maps A onto A’, such that A’ is on line CA and CA*CA’=k, where CA and CA’ denote the lengths of the corresponding segments. If k=1, then the lengths CA and CA’ are reciprocals.

discussion

INVERT

#link

Syntax

```
INVERT([G], [x1, y1], [x2, y2])
```

Description

Inverts the rectangle on G defined by the diagonal points (x1,y1) and (x2,y2). The effect is reverse video.

The following values are optional and their defaults are listed:
x1, y1=top left corner of G
x2, y2=bottom right corner of G

If only one x,y pair is specified, it refers to the top left corner of G.

discussion

INVERT_P

#link

Syntax

```
INVERT_P([G], [x1, y1], [x2, y2])
```

Description

Inverts the rectangle on G defined by the diagonal points (x1,y1) and (x2,y2). The effect is reverse video.

The following values are optional and their defaults are listed:
x1, y1=top left corner of G
x2, y2=bottom right corner of G

If only one (x,y) pair is specified, it refers to the top left corner of G.

discussion

invlaplace

#link

Syntax

```
ilaplace(Expr,[Var],[IlapVar])
```

Description

Returns the inverse Laplace transform of Expr.

Example

```
ilaplace(1/(x^2+1)^2) returns (-x/2)*cos(x)+(1/2)*sin(x)
```

discussion

invztrans

#link

Syntax

```
invztrans(Expr,[Var],[InvZtransVar])
```

Description

Inverse z transform of a rational fraction.

Example

```
invztrans(1/(x^2+1)^2) returns (x*exp(x*(-i)*π/2)+x*exp(x*
(i)*π/2)+4*Dirac(x)-2*exp(x*(-i)*π/2)-2*exp(x*(i)*π/2))/4
```

discussion

IP

#link

Syntax

```
IP(value)
```

Description

Integer part. Returns the Integer part of value.

Example

```
IP(23.2) returns 23
```

discussion

iPart

#link

Syntax

```
iPart(Real||LstReal)
```

Description

Returns the argument without its fractional part (type=DOM_FLOAT).

Example

```
iPart(4.3) returns 4.0
```

discussion

iquo

#link

Syntax

```
iquo(a, b)
```

Description

Euclidean quotient. Returns the integer quotient when the integer a is divided by the integer b.

Example

```
iquo(63, 23) returns 2
```

discussion

iquorem

#link

Syntax

```
iquorem(a, b)
```

Description

Euclidean quotient and remainder. Returns the integer quotient and remainder when the integer a is divided by the integer b.

Example

```
iquorem(63, 23) returns [2, 17]
```

discussion

irem

#link

Syntax

```
irem(a, b)
```

Description

Euclidean remainder. Returns the integer remainder when the integer a is divided by the integer b.

Example

```
irem(63, 23) returns 17
```

discussion

#link

is_collinear

Syntax

```
is_collinear(Point1, Point2, ..., Pointn)
```

Description

Takes a set of points as argument and tests whether or not they are collinear. Returns 1 if the points are collinear and 0 otherwise.

Example

```
is_collinear(point(0,0), point(5,0), point(6,1)) returns 0.
```

discussion

#link

is_concyclic

Syntax

```
is_concyclic(Point1, Point2, …, Pointn)
```

Description

Takes a set of points as argument and tests if they are all on the same circle. Returns 1 if the points are all on the same circle and 0 otherwise.

Example

```
is_concyclic(point(-4,-2), point(-4,2), point(4,-2), point(4,2)) returns
1
```

discussion

#link

is_conjugate

Syntax

```
is_conjugate(Crcle, Point1, Point2, [Point3]) or is_conjugate(Line1,
Line2, Line3, {Line4])
```

Description

Returns 1 if the 3 (resp 4) arguments are conjugated toward a circle (resp 2 lines) and 0 otherwise.

discussion

#link

is_coplanar

Syntax

```
is_coplanar(Point1, Point2, Point3, Point4)
```

Description

Tests if 4 points are in the same plane. Returns 1 if true or 0 if false.

discussion

is_element

#link

Syntax

```
is_element(Point, Object)
```

Description

Tests if a point is on a geometric object. Returns 1 if it is and 0 otherwise

Example

```
is_element(point((√(2)/2),(√(2)/2)),circle(0,1)) returns 1
```

discussion

is_equilateral

#link

Syntax

```
is_equilateral(Point1, Point2, Point3)
```

Description

Takes three points and tests whether or not they are vertices of a single equilateral triangle. Returns 1 if they are and 0 otherwise..

Example

```
is_equilateral(triangle(0,2,1+i*√3)) returns 1.
```

discussion

is_harmonic

#link

Syntax

```
is_harmonic(Pnt or Cplx,Pnt or Cplx,Pnt or Cplx,Pnt or Cplx)
```

Description

Returns 1 if the 4 points are in a harmonic division and 0 otherwise.

discussion

is_harmonic_circle_bundle

#link

Syntax

```
is_harmonic_circle_bundle(Lst(Crcle))
```

Description

Returns 1 if the circles build a bundle, 2 if they have the same center, 3 if they are the same and 0 otherwise.

discussion

is_harmonic_line_bundle

#link

Syntax

```
is_harmonic_line_bundle(Lst(Line))
```

Description

Returns 1 if the lines have a common point, 2 if they are parallels, 3 if they are the same and 0 otherwise.

discussion

is_isosceles

#link

Syntax

```
is_isosceles(Point1, Point2, Point3)
```

Description

Takes three points and tests whether or not they are vertices of a single isosceles triangle. Returns 0 if they are not. If they are, returns the number order of the common point of the two sides of equal length (1, 2, or 3). Returns 4 if the three points form an equilateral triangle.

Example

```
is_isosceles(point(0,0), point(4,0), point(2,4)) returns 3
is_isosceles(triangle(0,i,1+i)) returns 2
```

discussion

is_orthogonal

#link

Syntax

```
is_orthogonal(Line1, Line2) or is_orthogonal(Circle1, Circle2
```

Description

Tests whether or not two lines or two circles are orthogonal (perpendicular). In the case of two circles, tests whether or not the tangent lines at a point of intersection are orthogonal. Returns 1 if they are and 0 otherwise.

Example

```
is_orthogonal(line(y=x),line(y=-x)) returns 1.
```

discussion

is_parallel

#link

Syntax

```
is_parallel(Line1, Line2)
```

Description

Tests whether or not two lines are parallel. Returns 1 if they are and 0 otherwise.

Example

```
is_parallel(line(2x+3y=7),line(2x+3y=9) returns 1.
```

discussion

is_parallelogram

#link

Syntax

```
is_parallelogram(Point1, Point2, Point3, Point4)
```

Description

Tests whether or not a set of four points are vertices of a parallelogram. Returns 0 if they are not. If they are, then returns 1 if they form only a parallelogram, 2 if they form a rhombus, 3 if they form a rectangle, and 4 if they form a square.

Example

```
is_parallelogram(point(0,0), point(2,4), point(0,8), point(-2,4))
returns 2.
```

discussion

is_perpendicular

#link

Syntax

```
is_perpendicular(line1, Line2)
```

Description

Similar to is_orthogonal. Tests whether or not two lines are perpendicular. Returns 1 if they are or 0 if they are not.

Example

```
is_perpendicular(line(y=x),line(y=-x)) returns 1
```

discussion

is_rectangle

#link

Syntax

```
is_rectangle(Point1, Point2, Point3, Point4)
```

Description

Tests whether or not a set of four points are vertices of a rectangle. Returns 0 if they are not, 1 if they are, and 2 if they are vertices of a square.

Example

```
is_rectangle(point(0,0), point(4,2), point(2,6), point(-2,4)) returns 2.

With a set of only three points as argument, tests whether or not they
are vertices of a right triangle. Returns 0 if they are not. If they
are, returns the number order of the common point of the two
perpendicular sides (1, 2, or 3).

is_rectangle(point(0,0), point(4,2), point(2,6)) returns 2.
```

discussion

is_rhombus

#link

Syntax

```
is_rhombus(Pnt or Cplx,Pnt or Cplx,Pnt or Cplx,Pnt or Cplx)
```

Description

Returns 1 or 2 if the 4 points (or the object) build a rhombus (2 for a square) and 0 otherwise.

discussion

is_square

#link

Syntax

```
is_square(Point1, Point2, Point3, Point4)
```

Description

Tests whether or not a set of four points are vertices of a square. Returns 1 if they are and 0 otherwise.

Example

```
is_square(point(0,0), point(4,2), point(2,6), point(-2,4)) returns 1
```

discussion

ISKEYDOWN

#link

Syntax

```
ISKEYDOWN(Key_ID)
```

Description

Returns true (non-zero) if the key whose Key_ID is provided is currently pressed, and false (0) if it is not.

discussion

#link

ismith

Syntax

```
ismith(Mtrx(A))
```

Description

Smith normal form of a matrix with coefficients in Z : returns U,B,V such that U and V are invertible in Z, B is the diagonal, B[i,i] divide B[i+1,i+1] and B=U*A*V.

Example

```
ismith([[1,2,3],[4,5,6],[7,8,9]]) returns [[1,0,0],[4,-1,0],[-1,2,-1]],
[[1,0,0],[0,3,0],[0,0,0]],[[1,-2,1],[0,1,-2],[0,0,1]]
```

discussion

#link

isobarycenter

Syntax

```
isobarycenter(Point1, Point2, …, Pointn)
```

Description

Returns the hypothetical center of mass of a set of points. Works like barycenter but assumes all points have equal weight.

Example

```
isobarycenter(-3,3,3*√3*i) returns
point(3*√3*i/3), which is equivalent to (0,√3).
```

discussion

#link

isopolygon

Syntax

```
isopolygon(Point1, Point2, Realn), where realn is an integer greater
than 1.
```

Description

Draws a regular polygon given the first two vertices and the number of sides, where the number of sides is greater than 1. If the number of sides is 2, then the segment is drawn. You can provide CAS variable names for storing the coordinates of the calculated points in the order they were created. The orientation of the polygon is counterclockwise.

Example

```
isopolygon(GA, GB, 6) draws a regular hexagon whose first two vertices
```

are the points A and B.

discussion

isosceles_triangle

#link

Syntax

`isosceles_triangle(Point1, Point2, Angle)`

Description

Draws an isosceles triangle defined by two of its vertices and an angle. The vertices define one of the two sides equal in length and the angle defines the angle between the two sides of equal length. Like equilateral_triangle, you have the option of storing the coordinates of the third point into a CAS variable.

Example

isosceles_triangle(GA, GB, angle(GC, GA, GB) defines an isosceles triangle such that one of the two sides of equal length is AB, and the angle between the two sides of equal length has a measure equal to that of ∡ACB.

discussion

isPrime

#link

Syntax

`isprime(a)`

Description

Prime integer test. Returns true if the integer a is prime; otherwise, returns false.

Example

isprime(1999) returns true

discussion

ITERATE

#link

Syntax

`ITERATE(expr, var, ivalue, #times)`

Description

Repeatedly for #times evaluates expr in terms of var. The value for var is updated each time, starting with ivalue.

ITERATE(X^2, X, 2, 3) returns 256.

discussion

ithprime

#link

Syntax

`ithprime(n)`

Description

Nth prime. For the integer n, returns the nth prime number less than 100,000-200,000.

Example

`ithprime(5) returns 11`

discussion

jacobi_symbol

#link

Syntax

`jacobi_symbol(Intg,Intg)`

Description

Jacobi symbol.

Example

`jacobi_symbol(132,5) returns -1`

discussion

jordan

#link

Syntax

`jordan(Mtrx)`

Description

Returns the list made by the matrix of passage and the Jordan form of a matrix.

Example

`jordan([[0,2],[1,0]]) returns [[√2,-√2],[1,1]],[[√2,0],[0,-√2]]`

discussion

JordanBlock

#link

Syntax

`JordanBlock(Expr(a),Intg(n))`

Description

Returns a matrix n*n with a on the diagonal, 1 above, and 0 everywhere else.

Example

`JordanBlock(7,3) returns [[7,1,0],[0,7,1],[0,0,7]]`

discussion

ker

#link

Syntax

```
ker(Mtrx(M))
```

Description

Kernel of a linear application of matrix M.

Example

```
ker([[1,2],[3,6]]) returns [ 2, -1 ]
```

discussion

KILL

#link

Syntax

```
KILL;
```

Description

Stops the execution of the program.

discussion

l1norm

#link

Syntax

```
l1norm(Vect)
```

Description

Returns the l1 norm of the vector=sum of the absolute value of its coordinates.

Example

```
l1norm([3,-4,2]) returns 9
```

discussion

l2norm

#link

Syntax

```
l1norm(Vect)
```

Description

Returns the l1 norm of the vector=sum of the absolute value of its coordinates.

Example

```
l1norm([3,-4,2]) returns 9
```

discussion

lagrange

#link

Syntax

```
lagrange((Listxk, Listyk) or lagrange(Matrix)
```

Description

Returns the polynomial of degree n−1 such that P(xk)=yk, for k=0, 1, ..., n−1.

Example

```
lagrange([1,3],[0,1]) returns (1/2)* (x-1)
```

discussion

laguerre

#link

Syntax

```
laguerre(Integer)
```

Description

Returns the nth Laguerre polynomial.

Example

```
laguerre(4) returns (1/24)*a^4+(-1/6)*a^3*x+5/12*a^3+1/4*a^2*x^2+
(-3/2)*a^2*x+35/24*a^2+(-1/6)*a*x^3+7/4*a*x^2+
(-13/3)*a*x+25/12*a+1/24*x^4+(-2/3)*x^3+3*x^2-4*x+1
```

discussion

laplace

#link

Syntax

```
laplace(Expr,[Var],[LapVar])
```

Description

Returns the Laplace transform of Expr.

Example

```
laplace(exp(x)*sin(x)) returns 1/(x^2-2*x+2)
```

discussion

laplacian

#link

Syntax

```
laplacian(Expr(Xpr),LstVar)
```

Description

Returns the Laplacian of the expression Xpr with respect to the list of variables.

Example

`laplacian(exp(z)*cos(x*y),[x,y,z]) returns -x^2*cos(x*y)*exp(z)-y^2*cos(x*y)*exp(z)+cos(x*y)*exp(z)`

discussion

lcm

#link

Syntax

`lcm(Intgr1, Intgr2) or lcm(Poly1, Poly2) or lcm(Rational1, Rational2)`

Description

Returns the lowest common multiple of 2 polynomials of several variables or of 2 integers or of 2 rationals.

Example

`lcm(6,4) returns 12`

discussion

lcoeff

#link

Syntax

`lcoeff(Poly||Lst)`

Description

Returns the coefficient of the term of highest degree of a polynomial (l=leading).

Example

`lcoeff(-2*x^3+x^2+7*x) returns -2`

discussion

left

#link

Syntax

discussion

LEFT

#link

Syntax

`LEFT(string, n)`

Description

Returns the first n characters of the string.

Example

```
LEFT("MOMOGUMBO",3) returns "MOM"
```

discussion

legendre

#link

Syntax

```
legendre(Integer)
```

Description

Returns the nth Legendre polynomial.

Example

```
legendre(4) returns (35/8)*x^4+(-15/4)*x^2+3/8
```

discussion

legendre_symbol

#link

Syntax

```
legendre_symbol(Intg,Intg)
```

Description

Legendre symbol.

Example

```
legendre_symbol(132,5) returns -1
```

discussion

length

#link

Syntax

```
size(Lst or Str or Seq)
```

Description

Returns the size of a list, a string or a sequence.

Example

```
size([1,2,3]) returns 3
```

discussion

lgcd

#link

Syntax

```
lgcd(Seq or Lst )
```

Description

Returns the greatest common divisor of a list of polynomials or of integers.

Example

```
lgcd({45,75,20,15}) returns 5
lgcd({x^2-2*x+1,x^3-1,x-1}) returns x-1
```

discussion

limit

#link

Syntax

```
limit(Expr,Var,Val)
```

Description

Limit of an expression as a variable approaches a value. Returns the limit (2 sided or 1-sided) of the given expression as the given variable approaches a value.

Example

```
limit((n*tan(x)-tan(n*x))/(sin(n*x)-n*sin(x)),x,0) returns 2
```

discussion

## lin

#link

Syntax

```
lin(Expr )
```

Description

Linearization of exponentials.

Example

```
lin((exp(x)^3+exp(x))^2) returns exp(6*x)+2*exp(4*x)+exp(2*x)
```

discussion

line

#link

Syntax

```
line(Point1, Point2) or line(a*x+b*y+c) or line(point1, slope=realm)
```

Description

Draws a line. The arguments can be two points, a linear expression of the form a*x+b*y+c, or a point and a slope.

Example

```
line(2+i, 3+2i) draws the line whose equation is y=x-1; that is, the
line through the points (2,1) and (3,2).
line(2x-3y-8) draws the line whose equation is 2x-3y=8
line(3-2i,slope=1/2) draws the line whose equation is x-2y=7; that is,
the line through (3, -2) with slope m=1/2
```

discussion

LINE

#link

Syntax

```
LINE([G], x1, y1, x2, y2, [color])
```

Description

Draws a line on GROB G between points (x1,y1) and (x2,y2).

discussion

LINE_P

#link

Syntax

```
LINE_P([G], x1, y1, x2, y2, [color])
```

Description

Draws a line on GROB G between points (x1,y1) and (x2,y2).

discussion

linear_interpolate

#link

Syntax

```
linear_interpolate(Mtrx,xmin,xmax,xstep)
```

Description

Makes a regular sample from a polygonal line defined by a 2 row matrix.

Example

```
linear_interpolate([[1,2,6,9],[3,4,6,7]],1,9,1) returns
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0],
[3.0,4.0,4.5,5.0,5.5,6.0,6.33333333333,6.66666666667,7.0]]
```

discussion

linear_regression

#link

Syntax

```
linear_regression(Lst||Mtrx(A),[Lst])
```

Description

Returns the coefficients a and b of y=a*x+b
It is the best line approximation to the points where the coordinates are the rows of A (or the 2 lists).

Example

```
linear_regression([[0.0,0.0],[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]])
returns 4.0,-2.0
```

discussion

LineHorz

#link

Syntax

```
LineHorz(Expr(a))
```

Description

Draws the horizontal line y=a

discussion

LineTan

#link

Syntax

```
LineTan(f(x), [Var], Value)
```

Description

Draws the tangent to y=f(x) at x=Value.

Example

```
LineTan(x^2-x, 1) draws the line whose equation is y=x-1, which is
tangent to the graph of y=x^2-x at x=1.
```

discussion

LineVert

#link

Syntax

```
LineVert(Expr(a))
```

Description

Draws the vertical line x=a

discussion

linsolve

#link

Syntax

```
linsolve(ListLinEq,ListVar)
```

Description

Linear equations system solver. Solves a set of linear equations for their common variable set.

Example

```
linsolve([x+y+z=1,x-y=2,2*x-z=3],[x,y,z]) returns [3/2,-1/2,0]
```

discussion

ΠLIST

#link

Syntax

```
ΠLIST(list)
```

Description

List Product. Calculates the product of all elements in list.

Example

```
ΠLIST({2,3,4}) returns 24.
```

discussion

ΔLIST

#link

Syntax

```
ΔLIST(list)
```

Description

List Difference. Creates a new list composed of the first differences of list; that is, the differences between the sequential elements in list. The new list has one fewer elements than list.

Example

```
ΔLIST({1, 2, 3, 5, 8}) returns {1, 1, 2, 3}
```

discussion

ΣLIST

#link

Syntax

```
ΣLIST(list)
```

Description

Sum of a list. Returns the sum of all elements in list.

Example

```
ΣLIST({2,3,4}) returns 9
```

discussion

list2mat

#link

Syntax

```
list2mat(Lst(l),Intg(n))
```

Description

Returns the matrix with n columns and where terms are the list l completed eventually by 0.

Example

```
list2mat([1,8,4,9],1) returns [[1],[8],[4],[9]]
```

discussion

LN

#link

Syntax

```
LN(Value)
```

Description

Returns the natural logarithm of Value. The natural logarithm is the logarithm to the base e, Euler's number.

Example

```
LN(1) returns 0
```

discussion

lname

#link

Syntax

```
lname(Expr )
```

Description

List of variables in the expression.

Example

```
lname(exp(x)*2*sin(y)) returns [x,y]
```

discussion

lncollect

#link

Syntax

```
lncollect(Expr)
```

Description

Collect logarithms. Applies ln(a)+n*ln(b)=ln(a*b^n) where n is an integer.

Example

```
lncollect(ln(x)+2*ln(y)) returns ln(x*y^2)
```

discussion

#link

lnexpand

Syntax

```
lnexpand(Expr)
```

Description

Expands logarithms.

Example

```
lnexpand(ln(3*x)) returns ln(3)+ln(x)
```

discussion

#link

LNP1

Syntax

```
LNP1(value)
```

Description

Natural log plus 1. This is more accurate than the natural logarithm function when x is close to zero.

Example

```
LNP1(.23) returns .207014169384
```

discussion

#link

LOCAL

Syntax

```
LOCAL var_1[:=value][, more variables];
```

Description

Declares a local variable.
If the declaration is in a function block, these variables will be local to the function.
if the declaration is in the main program body, the variables are local to the program.

discussion

locus

#link

Syntax

```
locus(Point,Element)
```

Description

Given a first point and a second point that is an element of (a point on) a geometric object, draws the locus of the first point as the second point traverses its object.

discussion

LOG

#link

Syntax

```
LOG(Value, [Base])
```

Description

Returns the logarithm of Value in Base. By default, Base=10.

Example

```
LOG(8,2) returns 3 while LOG(8) returns 0.903089986992
```

discussion

log10

#link

Syntax

```
log10(Expr)
```

Description

Common logarithm (base 10).

Example

```
log10(10) returns 1
```

discussion

logarithmic_regression

#link

Syntax

```
logarithmic_regression(Lst||Mtrx(A),[Lst])
```

Description

Returns the coefficients a and b of y=a*ln(x)+b : it is the best logarithm that approximates the points where the coordinates are the rows of A (or the 2 lists).

Example

```
logarithmic_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]])
returns 10.1506450002,-0.564824055818
```

discussion

logb

#link

Syntax

```
logb(Real)
```

Description

Logarithm of base b.

Example

```
logb(5,2) returns ln(5)/ln(2) which is approximately 2.32192809489
```

discussion

logistic_regression

#link

Syntax

```
logistic_regression(Lst(L),Real(x0),Real(y0) )
```

Description

Returns y,y',C,y'max,xmax,R : y is a logistic function (sol of y'/y=a*y+b), such that y(x0)=y0 and where [y'(x0),y'(x0+1)...] is the best approximation of L.

Example

```
logistic_regression([0.0,1.0,2.0,3.0,4.0],0.0,1.0) returns
[-17.77/(1+exp(-0.496893925384*x+2.82232341488+3.14159265359*i)),-2.4854222
```

discussion

LQ

#link

Syntax

```
LQ(matrix)
```

Description

LQ Factorization. Factors an m n matrix into three matrices: {[[m n lowertrapezoidal]],[[n n orthogonal]], [[m m permutation]]}.

Example

```
LQ([[1,2],[3,4]])
```

discussion

LSQ

#link

Syntax

```
LSQ(matrix1, matrix2)
```

Description

Least Squares. Displays the minimum norm least squares matrix (or vector).

Example

```
LSQ([[1,2],[3,4]],[[5],[11]]) returns [[1],[2]]
```

discussion

LU

#link

Syntax

```
LU(matrix)
```

Description

LU Decomposition. Factors a square matrix into three matrices:
{[[lowertriangular]],[[uppertriangular]],[[permutation]]}
The uppertriangular has ones on its diagonal.

Example

```
LU([[1,2],[3,4]])
```

discussion

lvar

#link

Syntax

```
lvar(Expr)
```

Description

List of variables of an object (with rational dependence).

Example

```
lvar(exp(x)*2*sin(y)) returns [exp(x),sin(y)]
```

discussion

magenta

#link

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB-GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

MAKELIST

#link

Syntax

```
MAKELIST(expression, variable, begin, end, [increment])
```

Description

Make List. Calculates a sequence of elements for a new list. Evaluates expression, incrementing variable from begin to end values, using increment steps (default 1). The MAKELIST function generates a series by automatically producing a list from the repeated evaluation of an expression.

Example

```
MAKELIST(2*X-1, X, 1, 5, 1) returns {1, 3, 5, 7, 9}
```

discussion

MAKEMAT

#link

Syntax

```
MAKEMAT(expression, n, [m])
```

Description

Make Matrix. Creates a matrix of dimension n × m, using expression to calculate each element. If expression contains the variables I and J, then the calculation for each element substitutes the current row number for I and the current column number for J. With two arguments, this creates a vector of size n.

Example

```
MAKEMAT(0,3,3) returns [[0,0,0],[0,0,0],[0,0,0]]
MAKEMAT(√2,2,3) returns [[√2,√2,√2],[√2,√2,√2]]
MAKEMAT(I+J–1,2,3) returns [[1,2,3],[2,3,4]]
MAKEMAT(√2,2) returns [√2,√2]
```

discussion

MANT

#link

Syntax

```
MANT(value)
```

Description

Mantissa. Returns the significant digits of value.

Example

```
MANT(21.2E34) returns 2.12
```

discussion

map

#link

Syntax

```
map(Lst(l),Fnc(f))
```

Description

Apply the function f at the elements of the list l or at a polynomial of internal format.

Example

```
map([1,2,3],x->x^3) returns [1,8,27]
```

discussion

mat2list

#link

Syntax

```
mat2list(Mtrx)
```

Description

Returns the list of the terms of the matrix.

Example

```
mat2list([[1,8],[4,9]]) returns [1,8,4,9]
```

discussion

matpow

#link

Syntax

```
matpow(Mtrx,Intg(n))
```

Description

Calculates the n power of a matrix by jordanization.

Example

```
matpow([[1,2],[3,4]],n) returns [[(√33-3)*((√33+5)/2)^n*-6/(-12*√33)+(-(√33)-3)*((-(√33)+5)/2)^n*6/(-12*√33),(√33-3)*((√33+5)/2)^n*(-(√33)-3)/(-12*√33)+(-(√33)-3)*((-(√33)+5)/2)^n*(-(√33)+3)/(-12*√33)],[6*((√33+5)/2)^n*-6/(-12*√33)+6*((-(√33)+5)/2)^n*6/(-12*√33),6*((√33+5)/2)^n*(-(√33)-3)/(-12*√33)+6*((-(√33)+5)/2)^n*(-(√33)+3)/(-12*√33)]]
```

discussion

MAX

#link

Syntax

```
MAX(value1,[value2],[..value16])
```

Description

Maximum. Returns the greatest of the values given, or the greatest value of a list.

Example

```
MAX(210,25) returns 210 and MAX({1, 8, 2}) returns 8
```

discussion

maxnorm

#link

Syntax

```
maxnorm(Vect or Mtrx)
```

Description

Norm with the maximum of a vector (or of a matrix):
maxnorm([x1,x2,..,xn])=max(|x1|,..,|xn|)

Example

```
maxnorm([1,2]) returns 2
```

discussion

MAXREAL

#link

Syntax

```
MAXREAL
```

Description

Maximum real number. The largest real number the HP Prime is capable of representing. The value of MAXREAL is 9.99999999999E499. Any number larger than this is represented as this number.

discussion

mean

#link

Syntax

```
mean(Lst||Mtrx,[Lst])
```

Description

Mean of a list with the second argument as weight, or of the columns of a matrix.

Example

```
mean([1,2,3],[1,2,3]) returns 7/3
```

discussion

#link

median

Syntax

```
median(Lst||Mtrx,[Lst])
```

Description

Returns the median of a list with the second argument as the weight, or of the columns of a matrix.

Example

```
median([1,2,3,5,10,4]) returns 3.0
```

discussion

#link

median_line

Syntax

```
median_line(Point1, Point2, Point3)
```

Description

Given three points that define a triangle, creates the median of the triangle that passes through the first point and contains the midpoint of the segment defined by the other two points.

Example

```
median_line(0, 8i, 4) draws the line whose equation is y=2x; that is,
the line through (0,0) and (2,4), the midpoint of the segment whose
endpoints are (0, 8) and (4, 0).
```

discussion

#link

member

Syntax

```
member(Elem(e),(Lst(l) or Set(l)))
```

Description

Tests if e is in the list or the set l (=0 or k+1 with l[k]=e)

Example

```
member(1,[4,3,1,2]) returns 3
```

discussion

MID

#link

Syntax

```
MID(string, position, [n])
```

Description

Extracts n characters from string starting at position. If n is not specified, then MID extracts the remainder of the string from position.

Example

```
MID("MOMOGUMBO",3,5) returns "MOGUM"
MID("PUDGE",4) returns "GE"
```

discussion

midpoint

#link

Syntax

```
midpoint(Segment) or midpoint(Point1, Point2)
```

Description

Returns the midpoint of a segment. The argument can be either the name of a segment or two points that define a segment. In the latter case, the segment need not actually be drawn.

Example

```
midpoint(0,6+6i) returns point(3,3)
```

discussion

MIN

#link

Syntax

```
MIN(value1,[value2],[..value16])
```

Description

Minimum. Returns the lesser of the values given, or the lesser value of a list.

Example

```
MIN(210,25) returns 25 and MIN({1, 8, 2}) returns 1
```

discussion

MINREAL

#link

Syntax

```
MINREAL
```

Description

Minimum real number. The smallest real number that the HP Prime can represent. Its value is 1E-499. Any number smaller than this is represented as zero. discussion

#link

mkisom

Syntax

```
mkisom(Vect,(Sign(1) or -1))
```

Description

Matrix of an isometry given by its proper elements.

Example

`mkisom(π,1) returns [[-1,0],[0,-1]] in radian mode`

discussion

#link

MKSA

Syntax

```
MKSA(Value_Unit)
```

Description

Converts the measurement Value_Unit to its corresponding value and unit in Unit's MKSA equivalent. MKSA stands for the Meter-Kilogram-Second-Ampere system.

Example

`MKSA(32_yd) returns 29.2608_m.`

discussion

#link

MOD

Syntax

```
value1 MOD value2
```

Description

Modulo. Returns the remainder of value1/value2.

Example

`9 MOD 4 returns 1`

discussion

#link

modgcd

Syntax

```
modgcd(Poly,Poly)
```

Description

Returns the GCD of 2 polynomials, with the modular algorithm.

Example

```
modgcd(x^4-1,(x-1)^2) returns x-1
```

discussion

#link

MOUSE

Syntax

```
MOUSE[(index)]
```

Description

Returns the current pointer's location.
returns: two lists of the form {#x, #y, #originalx, #originaly, #type}, one for each potential pointer.
Note, if a pointer is unused, returns an empty list
#type is: #0: New, #1: Completed, #2: Drag, #3: Stretch, #4: Rotate, #5: LongClick

MOUSE(x) returns the nth element that would be returned if MOUSE was called with no arguements or -1 if the associated pointer is not down.

discussion

#link

mRow

Syntax

```
mRow(Expr(Xpr),Mtrx(A),Intg(n1))
```

Description

Multiplies the row n1 of the matrix A by Xpr.

Example

```
mRow(12,[[1,2],[3,4],[5,6]],1) returns [[12,24],[3,4],[5,6]]
```

discussion

#link

MSGBOX

Syntax

```
MSGBOX(expr,[OK_Cancel]) or MSGBOX(string,[OK_Cancel])
```

Description

Displays a message box with either the value of expression or string. If OK_Cancel? is true, displays OK and CANCEL menu keys, otherwise only displays the OK menu key.

Default value for OK_Cancel is false.

Returns true (non-zero) if the user presses OK, false (0) if the user presses CANCEL. discussion

#link

mult_c_conjugate

Syntax

`mult_c_conjugate(Expr)`

Description

Returns the expression after multiplication by the complex conjugated quantity of the denominator (or of the numerator if no denominator).

Example

`mult_c_conjugate(1/(3+i*2)) returns 1*(3+(-i)*2)/((3+(i)*2)*(3+(-i)*2))` discussion

#link

mult_conjugate

Syntax

`mult_conjugate(Expr)`

Description

Returns the expression after multiplication by the conjugated quantity of the denominator (or of the numerator if no denominator).

Example

`mult_conjugate(√3-√2) returns (√3-(√2))*(√3+√2)/(√3+√2)`

discussion

#link

nDeriv

Syntax

`nDeriv(Expr(Xpr),Var(var),[Real(h)])`

Description

Returns an approximation of the derivative number at a point:(Xpr(var+h)-Xpr(var-h))/(2*h) (by default h=0.001).

Example

`nDeriv(f(x),x,h) returns (f(x+h)-(f(x-h)))*0.5/h`

discussion

#link

NEG

Syntax

```
-Value or -Expression
```

Description

Unary minus.
Changes the sign of Value or Expression. Used to enter negative numbers.

discussion

#link

nextprime

Syntax

```
nextprime(a)
```

Description

Next prime. Returns the next prime number greater than the integer a.

Example

```
nextprime(12) returns 13
```

discussion

#link

normal

Syntax

```
normal(Expr)
```

Description

Simplify the expression.

Example

```
normal(2*x*2) returns 4*x
```

discussion

#link

NORMALD

Syntax

```
NORMALD([μ, σ,] x)
```

Description

Normal probability density function. Computes the probability density at the value x, given the mean, μ, and standard deviation, σ, of a normal distribution. With one argument, x, returns the probability density at x, assuming a mean of zero and standard deviation of 1.

Example

```
NORMALD(0.5) returns 0.352065326765 and NORMALD(0, 2, 0.5) returns
```

0.193334058402

discussion

NORMALD_CDF

#link

Syntax

NORMAL_CDF(μ, σ, x)

Description

Cumulative normal distribution function. Returns the lower-tail probability of the normal probability density function for the value x, given the mean, μ, and standard deviation, σ, of a normal distribution. With one argument, x, returns the lower-tail probability of the normal probability density function for the value x, assuming a mean of zero and standard deviation of 1.

Example

NORMAL_CDF(0, 1, 2) returns 0.97724986805

discussion

NORMALD_ICDF

#link

Syntax

NORMALD_ICDF(μ, ϭ, p)

Description

Inverse cumulative normal distribution function. Returns the cumulative normal distribution value associated with the lower-tail probability, p, given the mean, μ, and standard deviation, ϭ, of a normal distribution. With one argument, p, assumes a mean of zero and a standard deviation of one.

Example

NORMALD_ICDF(0, 1, 0.841344746069) returns 1

discussion

normalize

#link

Syntax

normalize(Lst||Cplx)

Description

Returns the vector divided by its l2norm. It is also an option for plotfield.

Example

normalize(3+4*i) returns (3+4*i)/5

discussion

NOT

#link

Syntax

`NOT Value`

Description

Logical NOT.

Returns 1 if Value is zero; otherwise returns 0.

Example

`NOT 3 returns 0`

discussion

nSolve

#link

Syntax

`nSolve(Expr,Var||orVar=Guess)`

Description

Returns a numerical solution of an equation or a system of equations.

Example

```
nSolve(cos(x)=x,x) returns 0.739085133215
nSolve(cos(x)=x,x=1.3) returns 0.739085133215
```

discussion

NTHROOT

#link

Syntax

`Value1 √ Value2`

Description

NTHROOT: the nth root function.
This Shift-key combination is the NTHROOT function. It returns the primary Value1 root of Value2. On the keyboard, the NTHROOT function is represented by n√ .

Example

`3√ 8 returns 2`

discussion

numer

#link

Syntax

```
numer(a,b)
```

Description

Simplified Numerator. For the integers a and b, returns the numerator of the fraction a/b after simplification.

Example

```
numer(10/12) returns 5
```

discussion

odd

#link

Syntax

```
odd(Intg(n))
```

Description

Returns 1 if the integer is odd, otherwise returns 0.

Example

```
odd(6) returns 0
```

discussion

odesolve

#link

Syntax

```
odesolve(Expr,VectVar,VectInitCond,FinalVal,[tstep=Val,curve])
```

Description

Ordinary Differential Equation solver. Solves an ordinary differential equation given by Expr, with variables declared in VectrVar and initial conditions for those variables declared in VectrInit. For example, odesolve(f(t,y),[t,y],[t0,y0],t1) returns the approximate solution of y'=f(t,y) for the variables t and y with initial conditions t=t0 and y=y0.

Example

```
odesolve(sin(t*y),[t,y],[0,1],2) returns [1.82241255674]
```

discussion

open_polygon

#link

Syntax

```
open_polygon(LstPnt||LstCplx)
```

Description

Returns and draws the polygonal line where its vertices are the element of l.

discussion

OR

#link

Syntax

```
Value1 OR Value2
```

Description

Logical OR.
Returns 1 if either Value1 or Value2 is non-zero, otherwise returns 0.

Example

```
3 OR 2 returns 1
```

discussion

order_size

#link

Syntax

```
order_size(Expr)
```

Description

Remainder (O term) of a series expansion: limit(x^a*order_size(x),x=0)=0 if a>0

discussion

ordinate

#link

Syntax

```
ordinate(Poinnt) or ordinate(Vecctor)
```

Description

Returns the ordinate of a point or a vector.

Example

```
ordinate(point(1+2*i)) returns 2
```

discussion

orthocenter

#link

Syntax

```
orthocenter(Triangle) or orthocenter(Point1, Point2, Point3)
```

Description

Returns the orthocenter of a triangle; that is, the intersection of the three altitudes of a triangle. The argument can be either the name of a triangle or three non-collinear points that define a triangle. In the latter case, the triangle does not need to be drawn.

Example

```
orthocenter(0,4i,4) returns (0,0)
```

discussion

pa2b2

#link

Syntax

```
pa2b2(Intg(n))
```

Description

Returns [a,b] such as a^2+b^2=n (for n prime and n=1 (mod 4))

Example

```
pa2b2(17) returns [4,1]
```

discussion

pade

#link

Syntax

```
pade(Expr(Xpr), Var(x), (Intg(n) || Poly(N)), Intg(p))
```

Description

Pade approximation P/Q=Xpr mod x^(n+1) or mod N with degree(P)<p

Example

```
pade(exp(x),x,10,6) returns (-x^5-30*x^4-420*x^3-3360*x^2-15120*x-30240)/(x^5-30*x^4+420*x^3-3360*x^2+15120*x-30240)
```

discussion

parabola

#link

Syntax

```
parabola(Point, Line) or parabola(Point, Realk) or parabola(Expr)
```

Description

Draws a parabola, given a focus point and a directrix line, or the vertex of the parabola and a real number that represents the focal length

Example

```
parabola(GA, GB) draws a parabola whose focus is point A and whose directrix is line B.
parabola(GA, 1) draws a parabola whose vertex is point A and whose focal length is 1.
parabola(x-y^2+y-2) draws the graph of the parabolic equation x=y^2-y+2
```

discussion

parallel

#link

Syntax

```
parallel(Point, Line)
```

Description

Draws a line through a given point that is parallel to a given line.

Example

`parallel(A, B)` draws the line through point A that is parallel to line B.

`parallel(point(3-2*i), line(x+y-5))` draws the line through the point (3, -2) that is parallel to the line whose equation is x+y=5; that is, the line whose equation is y=-x+1.

discussion

parallelogram

#link

Syntax

```
parallelogram(Point1, Point2, Point3)
```

Description

Draws a parallelogram given three of its vertices. The fourth point is calculated automatically but is not defined symbolically. As with most of the other polygon commands, you can store the fourth point’s coordinates into a CAS variable. The orientation of the parallelogram is counterclockwise from the first point.

Example

`parallelogram(0,6,9+5i)` draws a parallelogram whose vertices are at (0, 0), (6, 0), (9, 5), and (3,5). The coordinates of the last point are calculated automatically.

discussion

parameq

#link

Syntax

```
parameq(Obj)
```

Description

Returns a parametric equation for the geometric object Obj. The parametric equation is true for all complex numbers that represent points on Obj.

Example

`parameq(circle(0,1))` returns -exp(i*t)

discussion

partfrac

#link

Syntax

`partfrac(RatFrac or Opt)`

Description

Performs partial fraction decomposition on a fraction.

Example

`partfrac(x/(4-x^2)) returns (-1/2)/(x-2)-(1/2)/((x+2)`

discussion

pcoeff

#link

Syntax

`pcoeff(Vect)`

Description

Returns the polynomial coefficients having the roots specified in the vector Vect.

Example

`pcoeff([1,0,0,0,1]) returns poly1[1,-2,1,0,0,0]`

discussion

perimeter

#link

Syntax

`perimeter(Polygon) or perimeter(Circle)`

Description

Returns the perimeter of a polygon or the circumference of a circle.

Example

```
perimeter(0,1,i) returns √2+2
If GA is the point at (0, 0), GB is the point at (1, 0), and GC is
defined as circle(GA, GB-GA), then perimeter(GC) returns 2π.
If GA is the point at (0, 0), GB is the point at (1, 0), and GC is
defined as square(GA, GB-GA), then perimeter(GC) returns 4.
```

discussion

perimeterat

#link

Syntax

```
perimeterat(Polygon, Pnt||Cplx(z0))
```

Description

Displays at point(z0), with a legend, the perimeter of a circle or of a polygon (e.g. triangle, square, ...).

discussion

perimeteratraw

#link

Syntax

```
perimeteratraw(Polygone, Pnt||Cplx(z0))
```

Description

Displays at point(z0), the perimeter of a circle or of a polygon (e.g. triangle, square, ...).

discussion

PERM

#link

Syntax

```
PERM(n, r)
```

Description

Permutations. Returns the number of permutations (with regard to order) of n things taken r at a time: n!/(n−r)!

Example

```
PERM(5,2) returns 20
```

discussion

perpen_bisector

#link

Syntax

discussion

perpendicular

#link

Syntax

```
perpendicular(Point, Line) or perpendicular(Point1, Point2, Point3)
```

Description

Draws a line through a given point that is perpendicular to a given line. The line may be defined by its name, two points, or an expression in x and y.

Example

```
perpendicular(GA, GD) draws a line perpendicular to line D through point
```

```
A.
perpendicular(3+2i, GB, GC) draws a line through the point whose
coordinates are (3, 2) that is perpendicular to line BC.
perpendicular(3+2i,line(x-y=1)) draws a line through the point whose
coordinates are (3, 2) that is perpendicular to the line whose equation
is x – y = 1; that is, the line whose equation is y=-x+5.
```

discussion

#link

PI

Syntax

π

Description

The ratio of the circumference to the diameter of any circle. Internally represented as 3.14159265359.

discussion

#link

PIECEWISE

Syntax

discussion

#link

pivot

Syntax

`pivot(Mtrx(A),Intg(nl),Intg(nc))`

Description

Returns the matrix from A creating zeros in the column nc, by the method of Gauss-Jordan with the element A[nl,nc] as pivot.

Example

```
pivot([[1,2],[3,4],[5,6]],0,1) returns [[1,2],[0,-2],[0,-4]]
```

discussion

#link

PIXOFF

Syntax

`PIXOFF([G], x, y)`

Description

Sets the color of the pixel of G with coordinates (x,y) to white.

discussion

PIXOFF_P

#link

Syntax

```
PIXOFF_P([G], x, y)
```

Description

Sets the color of the pixel of G with coordinates (x,y) to white.

discussion

PIXON

#link

Syntax

```
PIXON([G], x, y, [color])
```

Description

Sets the color of the pixel of GROB G with coordinates (x,y).

discussion

PIXON_P

#link

Syntax

```
PIXON_P([G], x, y, [color])
```

Description

Sets the color of the pixel of GROB G with coordinates (x,y).

discussion

plotcontour

#link

Syntax

```
plotcontour(Expr(Xpr),[LstVar],[LstVal])
```

Description

Draws 11 contour-lines z=z_min,,...z=z_max of the surface z=Xpr, where the contour-lines are defined by the 3rd argument.

discussion

plotfield

#link

Syntax

```
plotfield(Expr,VectVar,[Opt])
```

Description

plotfield(f(t,y),[t,y]) draws the slope field of the differential equation y'=f(t,y)

discussion

plotfunc

#link

Syntax

```
plotfunc(Expr)
```

Description

Draws the plot of a function, given an expression in the independent variable x. Note the use of lowercase x.

Example

```
plotfunc(3*sin(x)) draws the graph of y=3*sin(x).
```

discussion

plotimplicit

#link

Syntax

```
plotimplicit(Expr,Var1,Var2)
```

Description

plotimplicit(f(x,y),x,y) or plotimplicit(f(x,y),[x,y]) graph of f(x,y)=0

discussion

plotinequation

#link

Syntax

```
plotinequation(Expr,[x=xrange,y=yrange],[xstep],[ystep])
```

Description

Shows the graph of the solution of inequations with 2 variables.

discussion

plotlist

#link

Syntax

```
plotlist(Lst(l)||Mtrx(M))
```

Description

Draws a polygonal line through the points of abscissa 0,...,n and ordinate l=[y0,...,yn] or the line through the points of abscissa in the first M column and the ordinates in the second column.

discussion

plotode

#link

Syntax

```
plotode(Expr,VectVar,VectInitCond)
```

Description

plotode(f(t,y),[t,y],[t0,y0]) draws the solution of y'=f(t,y) and y(t0)=y0 or of the system [x'=g(t,x,y),y'=h(t,x,y)] with x(t0)=x0 and y(t0)=y0.

discussion

plotparam

#link

Syntax

```
plotparam(Cplx||Lst,Var||Lst(Var))
```

Description

plotparam(a(x)+i*b(x),x=x0..x1) draws the curve X=a(x),Y=b(x) x=x0..x1 or plotparam([a(u,v),b(u,v),c(u,v)],[u=u0..u1,v=v0..v1]) draws the surface X=a(u,v),Y=b(u,v),Z=c(u,v) u=u0..u1 and v=v0..v1.

discussion

plotpolar

#link

Syntax

```
plotpolar(Expr,Var,VarMin,VarMax)
```

Description

plotpolar(f(x),x,a,b) draws the polar curve r=fx) for x in [a,b]

discussion

plotseq

#link

Syntax

```
plotseq(Expr(f(Var)),Var=[a,xm,xM],Intg(p))
```

Description

For seeing the pth terms of the sequence u(0)=a,u(n)=f(u(n-1))

discussion

pmin

#link

Syntax

```
pmin(Mtrx,[Var])
```

Description

Returns the minimal polynomial of a square matrix.

Example

`pmin([[1,0],[0,1]],x) returns x-1`

discussion

#link

point

Syntax

`point(Real1, Real2) or point(Expr1, Expr2) or point(Complex)`

Description

Creates a point, given the coordinates of the point. Each coordinate may be a value or an expression involving variables or measurements on other objects in the geometric construction.

Example

`point(3,4) creates a point whose coordinates are (3,4). This point may be selected and moved later.`

`point(abscissa(GA), ordinate(GB)) creates a point whose x-coordinate is the same as that of a point A and whose y-coordinate is the same as that of a point B. This point will change to reflect the movements of point A or point B.`

discussion

#link

point2d

Syntax

`point2d(Var1, Var2, ..., Varn)`

Description

Randomly re-distributes a set of points such that, for each point, x is in the interval [-5, 5] and y is in the interval [-5, 5]. Any further movement of one of the points will randomly re-distribute all of the points.

discussion

#link

POISSON

Syntax

`POISSON(μ, k)`

Description

Poisson probability mass function. Computes the probability of k occurrences of an event in a time interval, given μ expected (or mean) occurrences of the event in that interval.

For this function, k is a non-negative integer and μ is a real number.

Example

`POISSON(4, 2) returns 0.14652511111`

discussion

POISSON_CDF

#link

Syntax

`POISSON_CDF(μ, x)`

Description

Cumulative poisson distribution function. Returns the probability of x or fewer occurrences of an event in a given time interval, given μ expected (or mean) occurrences.

POISSON_CDF(4, 2) returns 0.238103305554

discussion

POISSON_ICDF

#link

Syntax

`POISSON_ICDF(μ, p)`

Description

Inverse cumulative poisson distribution function. Returns the value x such that the probability of x or fewer occurrences of an event in a time interval, with μ expected (or mean) occurrences of the event in the interval, is p.

Example

`POISSON_ICDF(4, 0.238103305554) returns 2`

discussion

polar

#link

Syntax

`polar(Crcle,Pnt or Cplxe(A))`

Description

Returns the line of the conjugated points of A with respect to the circle.

discussion

polar_coordinates

#link

Syntax

`polar_coordinates(Pnt or Cplx or LstRectCoord)`

Description

Returns the list of the norm and of the argument of the affix of a point (for 2D) or of a complex number or of the the list of rectangular coordinates.

Example

```
polar_coordinates(point(1+2*i)) returns [√5,atan(2)]
```

discussion

polar_point

#link

Syntax

```
polar_point(Real(r),Real(t))
```

Description

Returns the point (for 2D) with the arguments r and t as polar coordinates.

discussion

pole

#link

Syntax

```
pole(Crcle,Line)
```

Description

Returns the point having the line as polar with respect to the circle.

discussion

poly2symb

#link

Syntax

```
poly2symb(Lst,Var)
```

Description

Returns a polynomial (orthe polynomial and its value) in Var (by default x), the polynomial being defined by the vector of coefficents in Vect .

Example

```
poly2symb([1,2,3],x) returns (x+2)*x+3
poly2symb([1,2,3],x=2) returns (x+2)*x+3=11
```

discussion

POLYCOEF

#link

Syntax

```
POLYCOEF(vector or list)
```

Description

Polynomial coefficients. Returns the coefficients of the polynomial with the roots specified in vector.
POLYCOEF({−1, 1}) returns {1, 0, −1}

discussion

#link

POLYEVAL

Syntax

```
POLYEVAL(vector or list , value)
```

Description

Polynomial evaluation. Evaluates a polynomial with the coefficients specified in vector, at value.
POLYEVAL({1, 0, −1}, 3) returns 8

discussion

#link

polygon

Syntax

```
polygon(Point1, Point2, …, Pointn)
```

Description

Draws a polygon from a set of vertices.

Example

```
polygon(GA, GB, GD) draws ΔABD
```

discussion

#link

polygonplot

Syntax

```
polygonplot(Mtrx)
```

Description

Draws the polygons joining for j fixed and for k=0..nrows, the points (xk,yk) where xk=element row k column 0 and yk=element row k column j, when the xk are sorted (we obtain ncols−1 polygons).

discussion

#link

polygonscatterplot

Syntax

```
polygonscatterplot(Mtrx)
```

Description

Draws the points (xk,yk) and the polygons joining for j fixed and for k=0..nrows, the points (xk,yk) where xk=element row k column 0 and yk=element row k column j ,when the xk are sorted (we obtain ncols-1 polygons).

discussion

polynomial_regression

#link

Syntax

```
polynomial_regression(Lst||Mtrx(A),[Lst],Intg(n))
```

Description

Returns the coefficients (an,...a1,a0) of y=an*x^n+..a1x+a0): it is the best polynomial that approximates the points where the coordinates are the rows of A (or the 2 lists) (n is the 2nd argument).

Example

```
polynomial_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]],3)
returns [-0.0,1.0,-0.0,0.0]
```

discussion

POLYROOT

#link

Syntax

```
POLYROOT(vector)
```

Description

Polynomial roots. Returns the roots for the polynomial whose coefficients are specified in vector.

Example

```
POLYROOT([1, 0, -1]) returns {-1, 1}
```

discussion

POS

#link

Syntax

```
POS(list, element)
```

Description

List Position. Returns the position of element within list. If there is more than one instance of the element, the position of the first occurrence is returned. A value of 0 is returned if there is no occurrence of the specified element.

Example

```
POS({0, 1, 3, 5}, 1) returns 2
```

discussion

potential

#link

Syntax

```
potential(Vect(V),VectVar)
```

Description

Returns U such as derive(U,Vector_of_variable)=V

Example

```
potential([2*x*y+3,x^2-4*z,-4*y],[x,y,z]) returns 2*x^2*y/2+3*x-4*y*z
```

discussion

pow2exp

#link

Syntax

```
pow2exp(Expr)
```

Description

Converts powers to exponentials. Essentially the inverse of exp2pow.

Example

```
pow2exp(a^b) returns exp(b*ln(a))
```

discussion

power_regression

#link

Syntax

```
power_regression(Lst|Mtrx(A),[Lst])
```

Description

Returns the coefficients (m,b) of y=b*x^m: it is the best monomial that approximates the points where the coordinates are the rows of A (or the 2 lists).

Example

```
power_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]]) returns
2.0,1.0
```

discussion

powerpc

#link

Syntax

```
powerpc(Cercle,Pnt or Cplx)
```

Description

Returns the real number d^2-R^2 (d=distance between point and center, R=radius).

Example

```
powerpc(circle(0,1+i),3+i) returns 8
```

discussion

#link

powexpand

Syntax

```
powexpand(Expr)
```

Description

Expresses a power in the form of a product.

Example

```
powexpand(2^(x+y)) yields (2^x)*(2^y)
```

discussion

#link

powmod

Syntax

```
powmod(a, n, p)
```

Description

Power and modulo. For the integers a, n, and p, returns a^n mod p.

Example

```
powmod(5,2,13) returns 12
```

discussion

#link

prepend

Syntax

```
prepend(Lst,Elem )
```

Description

Puts the element at the beginning of the list.

Example

```
prepend([1,2],3) returns [3,1,2]
```

discussion

#link

preval

Syntax

```
preval(f(Var), Real1, Real2, [Var])
```

Description

Returns f(Real2)−f(Real1).

Example

```
preval(x^2-2,2,3) returns 5
```

discussion

#link

prevprime

Syntax

```
prevprime(a)
```

Description

Previous prime. Returns the previous prime number before the integer a.

Example

```
prevprime(11) returns 7
```

discussion

#link

primpart

Syntax

```
primpart(Poly,[Var])
```

Description

Returns the polynomial P divided by the gcd of its coefficients.

Example

```
primpart(2x^2+10x+6) returns x^2+5*x+3
```

discussion

#link

PRINT

Syntax

```
PRINT(expr) or PRINT(string) or PRINT( )
```

Description

Prints either the result of expr or string to the terminal.

The terminal is a program text output viewing mechanism which is displayed only when PRINT commands are executed. When visible, you can use the up/down keys to view the text, BKSP to erase the text and any other key to hide the terminal. You can show the terminal at anytime using the ON+T combination (press and HOLD the ON key, then press the T key, then release both keys). Pressing ON stops the interaction with the terminal.

PRINT with no argument clears the terminal.

discussion

product

#link

Syntax

`product(Expr||Lst,[Var||Lst],[Intg(a)],[Intg(b)],[Intg(p)])`

Description

Multiplicates the values of the expression when the variable goes from a to b with a step p (product expression,var,begin,end,step) by default p=1) or product of the elements of a list or product element by element of 2 lists or matrix.

Example

`product(n,n,1,10,2) returns 945`

discussion

projection

#link

Syntax

`projection(Curve, Point)`

Description

Draws the orthogonal projection of a point onto a curve.

Example

`projection(circle(x^2+y^2=4),point(6,6)) creates a point on the circle at (√2, √2)`

discussion

proot

#link

Syntax

`proot(Vect||Poly)`

Description

Returns all computed roots of a polynomial given by its coefficients (may not work if roots are not simple).

Example

`proot([1,0,-2]) returns [-1.41421356237,1.41421356237]`

discussion

propfrac

#link

Syntax

```
propfrac(Frac or RatFrac)
```

Description

Simplifies and writes the fraction (or rationnal fraction) A/B as Q+R/B with R<B (or deg(R)<deg(B))

Example

```
propfrac(28/12) returns 2+1/3
```

discussion

Psi

#link

Syntax

```
Psi(Real(a),Intg(n))
```

Description

Psi(a,n) returns the nth derivative of the digamma function at x=a (Psi(a,0)=Psi(a))

Example

```
Psi(3,1) returns π^2/6-5/4
```

discussion

ptayl

#link

Syntax

```
ptayl(Poly(P(var)),Real(a),[Var])
```

Description

Returns the Taylor polynomial Q such as P(x)=Q(x-a)

Example

```
ptayl(x^2+2*x+1,1) returns x^2+4*x+4
```

discussion

purge

#link

Syntax

```
purge(Var)
```

Description

purge(varname) unassigns the variable varname

discussion

PX→C

#link

Syntax

```
PX→C(x, y) or PX→C({x, y})
```

Description

Transform pixel coordinates into cartesian coordinates. Returns a list.

discussion

q2a

#link

Syntax

```
q2a(QuadraForm,VectVar)
```

Description

q2a(q(x,y),[x,y]) returns the symmetric matrix associated with the quadratic form q

Example

```
q2a(x^2+2*x*y+2*y^2,[x,y]) returns [[1,1],[1,2]]
```

discussion

QR

#link

Syntax

```
QR(matrix)
```

Description

QR Factorization. Factors an mn matrix into three matrices:
{[[mm orthogonal]],[[mn uppertrapezoidal]],[[nn permutation]]}.

Example

```
QR([[1,2],[3,4]])
```

discussion

quadrilateral

#link

Syntax

```
quadrilateral(Point1, Point2, Point3, Point4)
```

Description

Draws a quadrilateral from a set of four points.

Example

`quadrilateral(GA, GB, GC, GD) draws quadrilateral ABCD.`

discussion

quantile

#link

Syntax

`quantile(Lst(l),Real(p))`

Description

Returns the quantile of the elements of l corresponding to p (0<p<1)

Example

`quantile([0,1,3,4,2,5,6],0.25) returns [1.0]`

discussion

quartile1

#link

Syntax

`quartile1(Lst||Mtrx,[Lst])`

Description

Returns the 1st quartile of the elements (or of the columns) of the argument.

Example

`quartile1([1,2,3,5,10,4]) returns 2.0`

discussion

quartile3

#link

Syntax

`quartile3(Lst||Mtrx,[Lst])`

Description

Returns the 3rd quartile of the elements (or of the columns) of the argument

Example

`quartile3([1,2,3,5,10,4]) returns 5.0`

discussion

quartiles

#link

Syntax

`quartiles(Lst||Mtrx,[Lst])`

Description

Returns the min, 1st quartile, median, 3rd quartile, and max of the elements (or of the columns) of the argument.

Example

```
quartiles([1,2,3,5,10,4]) returns [[1.0],[2.0],[3.0],[5.0],[10.0]]
```

discussion

#link

quo

Syntax

```
quo((Vect or Poly),(Vect or Poly),[Var])
```

Description

Returns the Euclidean quotient of 2 polynomials

Example

```
quo([1,2,3,4],[-1,2]) returns poly1[-1,-4,-11]
```

discussion

#link

quorem

Syntax

```
quorem(Poly1, Poly2) or quorem(Vector1, Vector2)
```

Description

Returns the Euclidean quotient and remainder of the quotient of 2 polynomials in a vector. If the polynomials are expressed as vectors of their coefficients, then this command returns a similar vector of the quotient and a vector of the remainder.

Example

```
quorem(x^3+2*x^2+3*x+4,-x+2) returns [-x^2-4*x-11, 26]
quorem([1,2,3,4],[-1,2]) returns [[-1, -4, -11] [26]]
```

discussion

#link

QUOTE

Syntax

```
QUOTE(expression)
```

Description

Returns the expression unchanged and un-evaluated.
This function is mostly used with the STO► command in order to store a function in a function variable. For example if you want to store SIN(X) in F1.you cannot do SIN(X)►F1 as SIN(X) would be evaluated and a numerical result would be stored into F1. QUOTE(SIN(X))►F1 will store SIN(X) in F1.

discussion

radical_axis

#link

Syntax

```
radical_axis(Crcle,Crcle)
```

Description

Returns the line of points with same powerpc with respect to the 2 circles.

discussion

radius

#link

Syntax

```
radius(Circle)
```

Description

Returns the radius of a circle.

Example

```
If GA is the point at (0, 0), GB is the point at (1, 0), and GC is
defined as circle(GA, GB-GA), then radius(GC) returns 1.
```

discussion

randexp

#link

Syntax

```
randexp(Real(a))
```

Description

Returns a random real according to the exponential distribution of parameter a>0

Example

```
randexp(1) returns 1.17118631006
```

discussion

RANDINT

#link

Syntax

```
RANDINT([a],[b],[c])
```

Description

Random number. Returns a pseudo-random integer generated using a seed value, and updates the seed value.

With no argument, this function returns a random integer x from 0 to 1. With one argument, this returns a random integer x from 0 to a. With two arguments, this returns a random integer x from a to b. With three arguments, this returns a list of size a with each element being a random integer x from b to c.

Example

```
RANDINT(3,1,6) returns { random1, random2, random3 }
```

discussion

#link

RANDMAT

Syntax

```
RANDMAT (matrixname, rows, columns)
```

Description

Creates a random matrix with the specified number of rows and columns, and stores the result in matrixname. The entries will be integers ranging from -99 to 99.

Example

```
RANDMAT(M1,2,2) returns [[n1,n2],[n3,n4]]
```

discussion

#link

randMat

Syntax

```
ranm(Intg(n),[Intg(m)],[Interval or quote(DistribLaw)])
```

Description

Returns a list of size n or a n*m matrix that contains random integers in the range -99 through 99 with uniform distribution or contains random numbers according to the law in quote.

Example

```
ranm(3) returns [-20,72,-86]
```

discussion

#link

RANDNORM

Syntax

```
RANDNORM([μ],[σ]) or RANDNORM(n,μ,σ)
```

Description

Return a random number from the normal distribution with the specified mean μ and standard deviation σ. Default values are 0 and 1.

With three arguments, returns a list of size n with each element being a random number fron the normal distribution with the specified mean μ and standard deviation σ.

Example

```
RANDNORM(3,0,1) returns { random1, random2, random3 }
```

discussion

#link

randNorm

Syntax

```
randnorm(Real(mu),Real(sigma))
```

Description

Returns a random real with normal distribution N(mu,sigma)

Example

```
randnorm(0,1) returns -0.860967215689
```

discussion

#link

RANDOM

Syntax

```
RANDOM([a],[b],[c])
```

Description

Random number. Returns a pseudo-random number generated using a seed value, and updates the seed value.

With no argument, this function returns a random number x with $0 \leq x < 1$. With one argument, this returns a random number x with $0 \leq x < a$. With two arguments, this returns a random number x with $a \leq x < b$. With three arguments, this returns a list of size a with each element being a random number x with $b \leq x < c$.

Example

```
RANDOM(3,0,10) returns { random1, random2, random3 }
```

discussion

#link

randperm

Syntax

```
randperm(Intg(n))
```

Description

Returns a random permutation of [0,1,2,..,n−1]

Example

```
randperm(4) returns [2,1,3,0]
```

discussion

#link

randPoly

Syntax

```
randpoly([Var],Intgr,[Dist])
```

Description

Returns a vector of coefficients of a polynomial of variable Var (or x), of degree Intgr and where the coefficients are random integers in the range –99 through 99 with uniform distribution or in an interval specified by Intrvl.

Example

```
randpoly(t, 8, -1..1) returns a vector of 9 random integers, all of them
between -1 and 1.
```

discussion

#link

RANDSEED

Syntax

```
RANDSEED([value])
```

Description

Sets the random number generator seed. With no input, uses current time value as seed.

Example

```
RANDSEED(3.14)
```

discussion

#link

RANK

Syntax

```
RANK(matrix)
```

Description

Rank of a rectangular matrix.

Example

```
RANK([[1,2],[3,4]]) returns 2
```

discussion

#link

ratnormal

Syntax

```
ratnormal(Expr)
```

Description

Rewrites Expr as an irreducible rational fraction

Example

```
ratnormal((x^2-1)/(x^3-1)) returns (x+1)/(x^2+x+1)
```

discussion

#link

RE

Syntax

```
RE(x+yi)
```

Description

Real Part. Returns the real part of a complex number.

Example

```
RE(3+4i) returns 3
```

discussion

#link

reciprocation

Syntax

```
reciprocation(Crcle,Lst(Pnt,Line))
```

Description

Returns the list where a point is replaced with its polar or a line is replaced with its pole, with respect to the circle C

discussion

#link

RECT

Syntax

```
RECT([G], [x1, y1], [x2, y2], [edgecolor],[fillcolor])
```

Description

Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgecolor for the perimeter and fillcolor for the inside.

The following values are optional and their defaults are listed:
x1, y1=top left corner of G
x2, y2=bottom right corner of G
edgecolor=white
fillcolor=edgecolor

Note: To erase a GROB, execute RECT(G). To clear the screen execute RECT(). discussion

#link

RECT_P

Syntax

```
RECT_P([G], [x1, y1], [x2, y2], [edgeColor],[fillColor])
```

Description

Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgeColor for the perimeter and fillColor for the inside.

The following values are optional and their defaults are listed:
x1, y1=top left corner of G
x2, y2=bottom right corner of G
edgeColor=white
fillColor=edgeColor

Note: To erase a GROB, execute RECT(G). To clear the screen, execute RECT(). discussion

#link

rectangle

Syntax

```
rectangle(Point1, Point2, Point3) or rectangle(Point1, Point2, Realk)
```

Description

Draws a rectangle given two consecutive vertices and a point on the side opposite the side defined by the first two vertices or a scale factor for the sides perpendicular to the first side. As with many of the other polygon commands, you can specify optional CAS variable names for storing the coordinates of the other two vertices as points.

Example

```
rectangle(GA, GB, GE) draws a rectangle whose first two vertices are
points A and B (one side is segment AB). Point E is on the line that
contains the side of the rectangle opposite segment AB.
rectangle(GA, GB, 3, p, q) draws a rectangle whose first two vertices
are points A and B (one side is segment AB). The sides perpendicular to
segment AB have length 3*AB. The third and fourth points are stored into
the CAS variables p and q, respectively.
```

discussion

#link

rectangular_coordinates

Syntax

```
rectangular_coordinates(LstPolCoord)
```

Description

Returns the list of the abscissa and of the ordinate of a point given by the list of its polar coordinates.

Example

```
rectangular_coordinates([1,-1]) returns [cos(1),-sin(1)]
```

discussion

#link

red

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB–GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

#link

REDIM

Syntax

```
REDIM(matrixname, size)
```

Description

Redimensions the specified matrix or vector to size. For a matrix, size is a list of two integers {n1, n2}. For a vector, size is a list containing one integer {n}. Existing values in the matrix are preserved. Fill values will be zeros.

discussion

#link

reduced_conic

Syntax

```
reduced_conic(Expr,[LstVar])
```

Description

Returns the origin and the matrix of a base in which the conic (given by its equation) is reduced, 0 or 1 (0 if the conic is degenerate), and the equation of the conic in this base and also its parametric equation

Example

```
reduced_conic(x^2+2*x-2*y+1) returns [[-1,0],[[0,1],[-1,0]],1,y^2+2*x,
[[-1+(-i)*(t*t/-2+(i)*t),t,-4,4,0.1]]]
```

discussion

ref

#link

Syntax

```
ref(Mtrx(M))
```

Description

Performs Gauss reduction of a matrix AX=b (M=A|(-b))

Example

```
ref([[3,1,-2],[3,2,2]]) returns [[1,1/3,-2/3],[0,1,4]]
```

discussion

reflection

#link

Syntax

```
reflection(line, Object) or reflection(Point, Object)
```

Description

Reflects a geometric object over a line or through a point. The latter is sometimes referred to as a half-turn.

Example

```
eflection(line(x=3),point(1,1)) reflects the point at (1, 1) over the
vertical line x=3 to create a point at (5,1).
reflection(1+I, 3-2i) reflects the point at (3, -2) through the point at
(1, 1) to create a point at (-1, 4).
```

discussion

rem

#link

Syntax

```
rem(Poly1, Poly2) or rem(Vector1, Vector2)
```

Description

Returns the Euclidean remainder of the quotient of 2 polynomials. If the polynomials are expressed as vectors of their coefficients, then this command returns a similar vector of the remainder.

Example

```
rem(x^3+2*x^2+3*x+4,-x+2) returns 26
rem([1,2,3,4],[-1,2]) returns [26]
```

discussion

remove

#link

Syntax

```
remove(FncBool(f)||e,Lst(l))
```

Description

Removes the occurences e of l or the elements e such that f(e)=true

Example

```
remove(x->x>=5,[1,2,6,7]) returns [1,2]
```

discussion

reorder

#link

Syntax

```
reorder(Expr,LstVar)
```

Description

Reorders the variables in E according to the order of the 2nd argument

Example

```
reorder(x^2+2*x+y^2,[y,x]) returns y^2+x^2+2*x
```

discussion

REPEAT

#link

Syntax

```
REPEAT command(s) UNTIL test;
```

Description

executes command(s) UNTIL the test is true.

```
A:=5;
REPEAT
    PRINT(A);
   A:= A-1;
UNTIL A<1;
```

will print 5 4 3 2 1

discussion

REPLACE

#link

Syntax

```
REPLACE(object,start,object)
```

Description

Replaces portion of a matrix, vector or string starting at start by object.
For a matrix, start is a list containing two numbers; for a vector or string it is a single number.
Note: for strings, you can do: REPLACE("string", "sub_string", "replace_string") discussion

#link

residue

Syntax

```
residue(Expr,Var(v),Cplx(a))
```

Description

Returns the residue in a of the expression Expr with v as variable

Example

```
residue(1/z,z,0) returns 1
```

discussion

#link

restart

Syntax

```
restart(NULL)
```

Description

Purges all the variables

discussion

#link

resultant

Syntax

```
resultant(Poly,Poly,Var)
```

Description

Returns the inert form of the resultant for modular computation (irem/mod)

discussion

#link

RETURN

Syntax

```
RETURN expression;
```

Description

Exits from a function and returns the value of expression (optional).

Example

```
EXPORT FACTORIAL(N)
BEGIN
IF N==1 THEN RETURN 1; ELSE RETURN N*FACTORIAL(N-1); END;
END;
```

discussion

#link

REVERSE

Syntax

```
REVERSE(list)
```

Description

Reverse list. Reverses the order of the elements in list and returns them in a new list.

Example

```
REVERSE({2, 3, 4, 5}) returns {5, 4, 3, 2}.
```

discussion

#link

revlist

Syntax

```
revlist(Lst(l))
```

Description

Returns the list l in reverse order

Example

```
revlist([1,2,3]) returns [3,2,1]
```

discussion

#link

RGB

Syntax

```
RGB(R, G, B, [A])
```

Description

Returns an integer number that can be used as the color parameter for a drawing function. Based on Red, Green and Blue components values (0 to 255).

If Alpha is greater than 128, returns the color flagged as transparent. There is no alpha channel blending on Prime.

discussion

rhombus

[#link]

Syntax

```
rhombus(Pnt(A)||Cplx,Pnt(B)||Cplx,Angle(a)||Pnt(P)||Lst(P,a)),[Var(C)],
[Var(D)])
```

Description

Returns and draws the rhombus ABCD such that the angle (AB,AD)=a or such that in the plane ABP the angle(AB,AD)=angle(AB,AP)

discussion

RIGHT

#link

Syntax

```
RIGHT(string, n)
```

Description

Returns the last n characters of the string.

Example

```
RIGHT("MOMOGUMBO",5) returns "GUMBO"
```

discussion

right

#link

Syntax

discussion

right_triangle

#link

Syntax

```
right_triangle(Point1, Point2, Realk)
```

Description

Draws a right triangle given two points and a scale factor. One leg of the right triangle is defined by the two points, the vertex of the right angle is at the first point, and the scale factor multiplies the length of the first leg to determine the length of the second leg.

Example

```
right_triangle(GA, GB, 1) draws an isosceles right triangles with its
right angle at point A, and with both legs equal in length to segment
AB.
```

discussion

romberg

#link

Syntax

```
romberg(Expr(f(x)),Var(x),Real(a),Real(b))
```

Description

Uses Romberg's method to return the approximate value of the integral of the expression over the interval a to b

Example

```
romberg(exp(x^2),x,0,1) returns 1.46265174591
```

discussion

ROTATE

#link

Syntax

```
ROTATE(string, n)
```

Description

If n is not negative, takes the first n characters of string and put them on the right of string. If n is negative, takes the last n characters and put them on the left of string. If ABS(n)>dim(string), returns string.

Example

```
ROTATE("12345",2) returns "34512"
ROTATE("12345",-1) returns "51234"
ROTATE("12345",6) returns "12345"
```

discussion

rotation

#link

Syntax

```
rotate(Point, Angle, Object)
```

Description

Rotates a geometric object, about a given center point, through a given angle.

Example

```
rotate(GA, angle(GB, GC, GD),GK) rotates the geometric object labeled K,
about point A, through an angle equal to ∡CBD.
```

discussion

ROUND

#link

Syntax

`ROUND(value, [places])`

Description

Rounds value to system display settings. If optional places is given, rounds value to places decimal places. If places is negative, rounds to significant digits instead.

Example

`ROUND(7.8676,2) returns 7.87`

discussion

#link

row

Syntax

`row(Mtrx(A),Intg(n)||Interval(n1..n2))`

Description

Returns the row n or the sequence of the rows n1..n2 of the matrix A

Example

`row([[1,2,3],[4,5,6],[7,8,9]],1) returns [4,5,6]`

discussion

#link

rowAdd

Syntax

`rowAdd(Mtrx(A),Intg(n1),Intg(n2))`

Description

Returns the matrix obtained from matrix A when the n2th row is replaced by the sum of the n1th and n2th rows

Example

`rowAdd([[1,2],[3,4],[5,6]],1,2) returns [[1,2],[3,4],[8,10]]`

discussion

#link

rowDim

Syntax

`rowDim(Mtrx)`

Description

Returns the number of rows of a matrix

Example

```
rowDim([[1,2,3],[4,5,6]]) returns 2
```

discussion

#link

ROWNORM

Syntax

```
ROWNORM(matrix)
```

Description

Row Norm. Finds the maximum value (over all rows) for the sums of the absolute values of all elements in a row.

Example

```
ROWNORM([[1,2],[3,4]]) returns 7
```

discussion

#link

rowSwap

Syntax

```
rowSwap(Mtrx(A),Intg(n1),Intg(n2))
```

Description

Returns the matrix obtained from A by swapping the n1th row and the n2th row

Example

```
rowSwap([[1,2],[3,4],[5,6]],1,2) returns [[1,2],[5,6],[3,4]]
```

discussion

#link

RREF

Syntax

```
RREF(matrix)
```

Description

Reduced-Row Echelon Form. Changes a rectangular matrix to its reduced row-echelon form.

Example

```
RREF([[1,-2,1],[3,4,-1]]) returns [[1,0,.2],[0,1,-.4]]
```

discussion

#link

rsolve

Syntax

```
rsolve((Expr or LstExpr),(Var or LstVar),(InitVal or LstInitVal))
```

Description

Gives the value of a recurrent sequence or of a system of recurrent sequences

Example

```
rsolve(u(n+1)=2*u(n)+n,u(n),u(0)=1 returns [-n+2*2^n-1]
```

discussion

R→B

#link

Syntax

```
R→B(Real [, bits [,base]])
```

Description

Transform a real number into an integer. Optionally specifies bits and base.
-64<Bits<65
0<=Base<=4
0: system, 1: bin, 2: oct, 3: dec, 4: hex

discussion

SCALE

#link

Syntax

```
SCALE(matrixname, value, rownumber)
```

Description

Multiplies the specified row_number of the specified matrix by value.

discussion

SCALEADD

#link

Syntax

```
SCALEADD(matrixname, value, row1, row2)
```

Description

Multiplies the specified row1 of the matrix name by value, then adds this result to the second specified row2 of the matrix matrixname.

discussion

SCHUR

#link

Syntax

```
SCHUR(matrix)
```

Description

Schur Decomposition. Factors a square matrix into two matrices. If matrix is real, then the result is {[[orthogonal]],[[upper-quasi triangular]]}.
 If Complex mode is on and the matrix is complex, then the result is {[[unitary]],[[upper-triangular]]}.

Example

```
SCHUR([[1,2],[3,4]])
```

discussion

#link

SEC

Syntax

```
SEC(value)
```

Description

Secant. The Secant function; that is, 1/cos(x).

Example

```
SEC(0) returns 1 in degree mode
```

discussion

#link

segment

Syntax

```
segment(Point1, Point2)
```

Description

Draws a segment defined by its endpoints.

Example

```
segment(1+2i, 4) draws the segment defined by the points whose
coordinates are (1, 2) and (4, 0).
segment(GA, GB) draws segment AB.
```

discussion

#link

select

Syntax

```
select(FncBool(f),Lst(l))
```

Description

Selects the elements e of l such that f(e)=true

Example

```
select(x->x>=5,[1,2,6,7]) returns [6,7]
```

discussion

seq

#link

Syntax

```
seq(Expr(Xpr),Intg(n)||Var(var),[Intg(a)],[Intg(b)],[Intg(p)])
```

Description

Returns the sequence (if 2 or 3 arguments) or the list (if 4 or 5 arguments) obtained when var goes from a to b (step p) in Xpr, or when Xpr is repeated n times.

Example

```
seq(2^k,k=0..8) returns 1,2,4,8,16,32,64,128,256
```

discussion

seqsolve

#link

Syntax

```
seqsolve((Expr or LstExpr),(Var or LstVar),(InitVal or LstInitVal))
```

Description

Gives the value of a recurrent sequence (u_{n+1}=f(u_n) or u_{n+2}=f(u_{n+1},u_n)...) or of a system of recurrent sequences

Example

```
seqsolve(2x+n,[x,n],1) returns -n-1+2*2^n
```

discussion

series

#link

Syntax

```
series(Expr,Equal(var=limit_point),[Order],[Dir(1,0,-1)])
```

Description

Returns the series expansion of an expression in the vicinity of a given equality variable. With the optional third and fourth arguments you can specify the order and direction of the series expansion. If no order is specified the series returned is fifth order. If no direction is specified, the series is bidirectional.

Example

```
series((x^4+x+2)/(x^2+1),x=0,5) returns 2+x-2x^2-
x^3+3x^4+x^5+x^6*order_size(x)
```

discussion

SETBASE

#link

Syntax

`SETBASE(#integer[, base])`

Description

Sets the base used for display of this integer.
If base is not specified the calculator default is used.
0<=Base<=4
0: system, 1: bin, 2: oct, 3: dec, 4: hex

discussion

SETBITS

#link

Syntax

`SETBITS(#integer[, bits])`

Description

Sets the number of bits used for calculations with this integer to bits.
If bits is not specified the calculator default is used.
-64<Bits<65

discussion

shift_phase

#link

Syntax

`shift_phase(Expr)`

Description

Returns the expressions where the phase of the evaluated trigonometric expressions is increased by π/2

Example

`shift_phase(sin(x)) returns -cos((π+2*x)/2)`

discussion

Si

#link

Syntax

`Si(Expr)`

Description

Sine integral int(sin(t)/t,t=0..x)

Example

```
Si(1.0) returns 0.946083070367
```

discussion

#link

SIGN

Syntax

```
SIGN(value) or SIGN(x+yi)
```

Description

Sign. Returns the sign of value. If positive, the result is 1; if negative, −1. If zero, the result is zero. For complex inputs returns the unit vector.

Example

```
SIGN (2) returns 1
```

discussion

#link

signature

Syntax

```
signature(Permut)
```

Description

Returns the signature of a permutation

Example

```
signature([2,1,4,5,3]) returns -1
```

discussion

#link

similarity

Syntax

```
similarity(Point, Realk, Angle, Object)
```

Description

Dilates and rotates a geometric object about the same center point.

Example

```
similarity(0, 3, angle(0,1,i),point(2,0)) dilates the point at (2,0) by
a scale factor of 3 (a point at (6,0)), then rotates the result 90°
counterclockwise to create a point at (0, 6)
```

discussion

#link

simplify

Syntax

```
simplify(Expr)
```

Description

Simplifies an expression.

Example

```
simplify(4*atan(1/5)-atan(1/239)) yields (1/4)*π
```

discussion

simult

#link

Syntax

```
simult(Mtrx(A),Mtrx(B))
```

Description

Returns the matrix where the column of index k is solution of A*X=column of index k of B (=B[0..nr−1,k..k] with nr=number of rows of B)

Example

```
simult([[3,1],[3,2]],[[-2],[2]]) returns [[-2],[4]]
```

discussion

SIN

#link

Syntax

```
SIN(Value)
```

Description

Returns the sine of Value. Value is interpreted as either degrees or radians, depending on the setting of Angle Measure in Home Modes or Symbolic Setup.

Example

```
in radians mode, SIN(π/2) returns 1
```

discussion

sin2costan

#link

Syntax

```
sin2costan(Expr)
```

Description

Rewrites Expr so that sin(x) is replaced by cos(x)*tan(x)

Example

```
sin2costan(sin(x)) returns tan(x)*cos(x)
```

discussion

sincos

#link

Syntax

```
sincos(Expr)
```

Description

Returns an expression with the complex exponentials rewritten in terms of sine and cosine.

Example

```
sincos(exp(-i*x)) returns cos(x)-i*sin(x)
```

discussion

single_inter

#link

Syntax

```
single_inter(Curve,Curve,[Pnt(A)||LstPnt(L)])
```

Description

Gives one of the intersections of 2 curves or surfaces (or the intersection near A or not in L)

discussion

SINH

#link

Syntax

```
SINH(value)
```

Description

Hyperbolic sine.

Example

```
SINH(1) returns 1.17520119364
```

discussion

SIZE

#link

Syntax

```
SIZE(list)
```

Description

List Size. Returns the number of elements in list.

Example

`SIZE({0, 1, 2, 3}) returns 4`

discussion

#link

slope

Syntax

`slope(Line||Pnt||Cplx,[Pnt||Cplx])`

Description

Returns the slope of the line defined in the argument

Example

`slope(line(1,2i)) returns -2`

discussion

#link

slopeat

Syntax

`slopeat(Segment, Point) or slopeat(Line, Point) or slopeat(Ray, Point)`

Description

Displays, with a legend, the value of the slope of the segment, ray, or line (Line may be a tangent, bisector, etc.) at the location Point in Plot view.

Example

```
slopeat(line(point(0,1), point(3,2)),point(-10,4)) places
"sline(point(0,1),point(3,2)=1/3" at the point (-10,4) in Plot view
```

discussion

#link

slopeatraw

Syntax

`slopeatraw(Line, Pnt||Cplx(z0))`

Description

slopeatraw(d,z0) displays the value of the slope of the line or segment d at point(z0)

discussion

#link

solve

Syntax

`solve(Expr,[Var] )`

Description

Solves a polynomial equation or a set of polynomial equations.

Example

`solve(x^2-3=1) returns {-2,2}`

discussion

#link

SORT

Syntax

`SORT(list)`

Description

Sort list. Sorts the elements of list in ascending order.

Example

`SORT({2, 9, 5, 3}) returns {2, 3, 5, 9}.`

discussion

#link

SPECNORM

Syntax

`SPECNORM(matrix)`

Description

Spectral Norm of matrix.

Example

`SPECNORM([[1,2],[3,4]]) returns 5.46498570422`

discussion

#link

SPECRAD

Syntax

`SPECRAD(matrix)`

Description

Spectral radius of matrix.

Example

`SPECRAD([[1,2],[3,4]]) returns 5.37228132327`

discussion

spline

#link

Syntax

```
spline(Lst(lx),Lst(ly),Var(x),Intg(d))
```

Description

Returns the natural spline through the points given by lx and ly, variable x, degree d

Example

```
spline([0,1,2],[1,3,0],x,3) returns [-5*x^3/4+13*x/4+1,5*(x-1)^3/4+-15*(x-1)^2/4+(x-1)/-2+3]
```

discussion

sqrfree

#link

Syntax

```
sqrfree(Expr)
```

Description

Returns a polynomial factorized as a product of powers of coprime factors where each factor has roots of multiplicity 1

Example

```
sqrfree(x^4-2*x^2+1) returns (x^2-1)^2
```

discussion

sqrt

#link

Syntax

```
√(Expr)
```

Description

Returns the square root of Expr

Example

```
√50 returns 5*√2
```

discussion

square

#link

Syntax

```
square(Point1, Point2)
```

Description

Draws a square, given two consecutive vertices as points.

Example

`square(0, 3+2i, p, q)` draws a square with vertices at (0, 0), (3, 2), (1, 5), and (-2, 3). The last two vertices are computed automatically and are saved into the CAS variables p and q.

discussion

STARTAPP

#link

Syntax

```
STARTAPP("AppName")
```

Description

Starts the app AppName. The App's START function will run if present. The App’s default view will be started. Note that the START function is always executed when the user presses the START menu key in the App Library. Also works for apps saved in the App Library.

discussion

STARTVIEW

#link

Syntax

```
STARTVIEW(ViewNumber[,Redraw])
```

Description

Starts a view of the current app. Redraw, is optional; if Redraw, is true (non 0), it will force a refresh for the view.

The view numbers are as follows:
0=Symbolic
1=Plot
2=Numeric
3=Symbolic Setup
4=Plot Setup
5=Numeric Setup
6=App Info
7=Views key

If the current app has views defined under the Views menu, then the following view numbers are used:
8=First special view (Split Screen Plot Detail)
9=Second special view (Split Screen Plot Table)
10=Third special view (Autoscale)
11=Fourth special view (Decimal)
12=Fifth special view (Integer)
13=Sixth special view (Trig)

If ViewNumber is negative, the following global views are used:
-1=HomeScreen
-2=Modes
-3=Memory Manager
-4=App Library
-5=Matrix Catalog
-6=List Catalog
-7=Program Catalog
-8=Note Catalog

discussion

stddev

#link

Syntax

```
stddev(Lst||Mtrx,[Lst])
```

Description

Returns the standard deviation of the elements in a list or of the list of standard deviations

Example

```
stddev([1,2,3]) returns (√6)/3
```

discussion

stddevp

#link

Syntax

```
stddevp(Lst||Mtrx,[Lst])
```

Description

Returns the population standard eviation of the elements of a list with the second argument as weight.

Example

```
stddevp([1,2,3]) returns 1
```

discussion

STEP

#link

Syntax

```
FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO command(s)
END;
```

Description

Sets variable var to start; then, for as long as this variable’s value is less than or equal to (or more than for a DOWNTO) finish, executes command(s) and adds (or substract for DOWNTO) 1 (or increment) to var.

```
FOR A FROM 1 TO 10 STEP 2
  DO
    PRINT(A);
END;
```

will print 1 3 5 7 9

discussion

#link

sto

Syntax

`sto(arg1,Var)`

Description

Stores the first argument in the variable given as second argument

Example

`sto("hello",b)`

discussion

#link

STRING

Syntax

`STRING(expression)`

Description

Evaluates expression and returns the result as a string.

discussion

#link

STRINGFROMID

Syntax

`STRINGFROMID(integer)`

Description

Returns the built-in string associated with the ID of the current language.

discussion

#link

STUDENT

Syntax

`STUDENT(n, x)`

Description

Student’s t probability density function. Computes the probability density of the Student’s-t distribution at x, given n degrees of freedom.

Example

```
STUDENT(3, 5.2) returns 0.00366574413491
```

discussion

#link

STUDENT_CDF

Syntax

```
STUDENT_CDF(n, x)
```

Description

Cumulative Student’s t distribution function. Returns the lower-tail probability of the Student’s t probability density function at x, given n degrees of freedom.

Example

```
STUDENT_CDF(3, -3.2) returns 0.0246659214813
```

discussion

#link

STUDENT_ICDF

Syntax

```
STUDENT_ICDF(n, p)
```

Description

Inverse cumulative Student’s t distribution function. Returns the value x such that the Student’s-t lower-tail probability of x, with n degrees of freedom, is p.

Example

```
STUDENT_ICDF(3, 0.0246659214813) returns -3.2
```

discussion

#link

sturmab

Syntax

```
sturmab(Poly,Var,Cplx1, Cplx2)
```

Description

Returns the number of sign changes of a polynomial in the interval (Cplx1, Cplx2] or the number of complex roots in (Cplx1, Cplx2] if Cplx1 or Cplx2 is non-real.

Example

```
sturmab(x^3-1,x,-2,5) returns 1
```

discussion

sturmseq

#link

Syntax

```
sturmseq(Poly,[Var])
```

Description

Returns the Sturm sequence corresponding to a polynomial or to a rational fraction

Example

```
sturmseq(x^3-1,x) returns [1,[[1,0,0,-1],[3,0,0],9],1]
```

discussion

SUB

#link

Syntax

```
SUB(object, start, end)
```

Description

Extracts a portion, of a list or matrix.

For a matrix, start and end are two lists of two numbers ({row, col}) specifying the top left and bottom right of the portion to extract.
For a vector or list, start and end are two numbers specifying the indexes of the first and last objects of the portion to extract.

discussion

SUBGROB

#link

Syntax

```
SUBGROB(srcG, [x1, y1], [x2, y2], trgtG)
```

Description

Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used.

trgtGRB can be any of the graphic variables except G0.

SUBGROB(G1, G4) will copy G1 in G4.

discussion

SUBGROB_P

#link

Syntax

```
SUBGROB_P(srcG, [x1, y1], [x2, y2], trgtG)
```

Description

Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used.

trgtGRB can be any of the graphic variables except G0.

SUBGROB(G1, G4) will copy G1 in G4.

discussion

subMat

#link

Syntax

```
subMat(Mtrx(A),Intg(n1),Intg(n2),Intg(n3),Intg(n4))
```

Description

Extracts a sub matrix with first element=A[n1,n2] and last element=A[n3,n4]

Example

```
subMat([[1,2],[3,4],[5,6]],1,0,2,1) returns [[3,4],[5,6]]
```

discussion

subst

#link

Syntax

```
subst(Expr,Var=value)
```

Description

Substitutes a value for a variable in an expression.

Example

```
subst(x/(4-x^2),x=3) returns -3/5
```

discussion

sum

#link

Syntax

```
sum(Expr,Var,Real1, Real2,[Step])
```

Description

Returns the discrete sum of Expr with respect to the variable Var from Real1 to Real2. You can also use the summation template in the Template menu.

Example

`sum(n^2,n,1,5) returns 55`

discussion

sum_riemann

#link

Syntax

`sum_riemann(Expr,List(Var1,Var2))`

Description

Returns, in the neighbourhood of n=∞, an equivalent of the sum of Expr(Var1,Var2) for Var2 from Var2=1 to Var2=Var1 when the sum is looked at as a Riemann sum associated with a continuous function defined on [0,1]

Example

`sum_riemann(1/(n+k),[n,k]) returns ln(2)`

discussion

suppress

#link

Syntax

`suppress(Vect(l),Intg(n))`

Description

Returns l without the element of index n

Example

`suppress([0,1,2,3],2) returns [0,1,3]`

discussion

surd

#link

Syntax

`surd(Expr,Intg(n))`

Description

Returns Expr to the power of 1/n

Example

`surd(8,3) returns 8^(1/3)`

discussion

SVD

#link

Syntax

`SVD(matrix)`

Description

Singular Value Decomposition. Factors an m n matrix into two matrices and a vector: {[[m m square orthogonal]],[[n n square orthogonal]], [real]}.

Example

`SVD([[1,2],[3,4]])`

discussion

#link

SVL

Syntax

`SVL(matrix)`

Description

Singular Values. Returns a vector containing the singular values of matrix.

Example

`SVL([[1,2],[3,4]])`

discussion

#link

SWAPCOL

Syntax

`SWAPCOL(matrixname, column1, column2)`

Description

Swap Columns. Exchanges column1 and column2 in the specified matrix matrixname.

discussion

#link

SWAPROW

Syntax

`SWAPROW(matrixname, row1, row2)`

Description

Swap Rows. Exchanges row1 and row2 in the specified matrix matrixname.

discussion

#link

sylvester

Syntax

```
sylvester(Poly,Poly,Var)
```

Description

Returns the Sylvester matrix of two polynomials

Example

```
sylvester(x^2-1,x^3-1,x) returns [[1,0,-1,0,0],[0,1,0,-1,0],
[0,0,1,0,-1],[1,0,0,-1,0],[0,1,0,0,-1]]
```

discussion

symb2poly

#link

Syntax

```
symb2poly(Expr,[Var]) or symb2poly(Expr, ListVar)
```

Description

Returns the coefficients of a polynomial Expr with respect tothe variable Var or if the second argument is a list returns the internal format of the polynomial. Essentaiilly the inverse of poly2symb().

Example

```
symb2poly( (x+2)*x+3) returns [1,2,3]
```

discussion

table

#link

Syntax

```
table(SeqEqual(index=value))
```

Description

Defines an array where the index are strings or real numbers

discussion

tail

#link

Syntax

```
tail(Lst or Seq or Str)
```

Description

Returns the list (or sequence or string) without its first element

Example

```
tail([3,2,4,1,0]) returns [2,4,1,0]
```

discussion

TAN

#link

Syntax

`TAN(Value)`

Description

Returns the tangent of Value. Value is interpreted as either degrees or radians, depending on the setting of Angle Measure in Home Modes or Symbolic Setup.

Example

`in radians mode, TAN(0) returns 0`

discussion

tan2cossin2

#link

Syntax

`tan2cossin2(Expr)`

Description

Rewrites Expr with tan(x) replaced by (1−cos(2*x))/sin(2*x)

Example

`tan2cossin2(tan(x)) returns (1-cos(2*x))/sin(2*x)`

discussion

tan2sincos

#link

Syntax

`tan2sincos(Expr)`

Description

Rewrites Expr with tan(x) using sin(x)/cos(x)

Example

`tan2sincos(tan(x)) returns sin(x)/cos(x)`

discussion

tan2sincos2

#link

Syntax

`tan2sincos2(Expr)`

Description

Rewrites Expr with tan(x) replaced by sin(2*x)/(1+cos(2*x))

Example

```
tan2sincos2(tan(x)) returns sin(2*x)/(1+cos(2*x))
```

discussion

#link

tangent

Syntax

```
tangent(Curve, Point)
```

Description

Draws the tangent(s) to a given curve through a given point. The point does not have to be a point on the curve.

Example

```
tangent(plotfunc(x^2), point(1,1)) draws the tangent to the graph y=x^2
through the point (1,1); that is, the line whose equation is y=2*x-1.
tangent(plotfunc(x^2), GA) draws the tangent to the graph of y=x^2
through point A. Point A can then be moved and the tangent will move
with it.
tangent(circle(GB, GC-GB), GA) draws one or more tangent lines through
point A to the circle whose center is at point B and whose radius is
defined by segment BC.
```

discussion

#link

TANH

Syntax

```
TANH(value)
```

Description

Hyperbolic tangent.

Example

```
TANH(1) returns .761594155956
```

discussion

#link

taylor

Syntax

```
taylor(Expr,[Var=Value],[Order])
```

Description

Returns the Taylor series expansion of an expression at a point or at infinity (by default, at x=0 and with relative order=5).

Example

`taylor(sin(x)/x,x=0) returns 1-(1/6)*x^2+(1/120)*x^4+x^6*order_size(x)`

discussion

tchebyshev1

#link

Syntax

`tchebyshev1(Integer))`

Description

Returns the nth Tchebyshev polynomial of the first kind.

Example

`tchebyshev1(3) returns 4*x^3-3*x`

discussion

tchebyshev2

#link

Syntax

`tchebyshev2(Integer)`

Description

Returns the nth Tchebyshev polynomial of the second kind.

Example

`tchebyshev2(3) returns 8*x^3-4*x`

discussion

tcollect

#link

Syntax

`tcollect(Expr)`

Description

Collects trigonometric expressions.

Example

`tcollect(sin(x)+cos(x)) returns √2*cos(x-1/4*π)`

discussion

texpand

#link

Syntax

`texpand(Expr)`

Description

Expands a transcendental expression; that is, an expression containing trigonometric, logarithmic, or exponential functions. texpand develops the expression in terms of sin(), cos(), ln(), and exp().

Example

`texpand(sin(2*x)+exp(x+y)) returns 2*cos(x)*sin(x)+exp(x)*exp(y)` discussion

#link

TEXTOUT

Syntax

```
TEXTOUT(text, [G], x, y, [font], [textColor], [width],
[backgroundColor])
```

Description

Draws text on graphic G at position (x, y) using font. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased.

The sizes for font are:

0=current font (default)
1=font_10
2=font_12 (Small)
3=font_14 (Medium)
4=font_16 (Large)
5=font_18
6=font_20
7=font_22

discussion

#link

TEXTOUT_P

Syntax

```
TEXTOUT_P(text, [G], x, y, [font], [textColor], [width],
[backgroundColor])
```

Description

Draws text on graphic G at position (x, y) using font. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased.

The sizes for font are:

0=current font (default)

1=font_10
2=font_12 (Small)
3=font_14 (Medium)
4=font_16 (Large)
5=font_18
6=font_20
7=font_22

discussion

THEN

#link

Syntax

```
IF test THEN command(s) [ELSE commands] END;
```

Description

Evaluates test. If test is true (non 0), executes command(s); otherwise, executes the comands in the ELSE clause nothing happens.

```
IF A<1
  THEN PRINT("A IS SMALLER THAN 1");
  ELSE PRINT("A IS LARGER THAN 1");
END;
```

discussion

tlin

#link

Syntax

```
tlin(Expr)
```

Description

Returns a trigonometric expression with the products and integer powers linearized

Example

```
tlin(sin(x)^3) returns (3/4)*sin(x)-(1/4)*sin(3*x)
```

discussion

TO

#link

Syntax

```
FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO command(s)
END;
```

Description

Sets variable var to start; then, for as long as this variable's value is less than or equal to (or more than for a DOWNTO) finish, executes command(s) and adds (or substract for DOWNTO) 1 (or increment) to var.

```
FOR A FROM 1 TO 10 STEP 2
  DO
    PRINT(A);
END;
```

will print 1 3 5 7 9

discussion

#link

TRACE

Syntax

`TRACE(matrix)`

Description

Trace of a square matrix. Finds the trace of a square matrix, equal to the sum of the diagonal elements (also equal to the sum of the eigenvalues).

Example

`TRACE([[1,2],[3,4]]) returns 5`

discussion

#link

trace

Syntax

`trace(Point)`

Description

Begins tracing the specified point.

discussion

#link

translation

Syntax

`translation(Vector, Object)`

Description

Translates a geometric object along a given vector. The vector is given as the difference of two points (head-tail).

Example

```
translation(0-i, GA) translates object A down one unit.
translation(GB-GA, GC) translates object C along the vector AB.
```

discussion

#link

transpose

Syntax

```
tran(Mtrx)
```

Description

Transposes a matrix (without conjugation)

Example

```
tran([[1,2,3],[1,3,6],[2,5,7]]) returns [[1,1,2],[2,3,5],[3,6,7]]
```

discussion

triangle

#link

Syntax

```
triangle(Point1, Point2, Point3)
```

Description

Draws a triangle, given its three vertices.

Example

```
triangle(GA, GB, GC) draws ΔABC.
```

discussion

TRIANGLE

#link

Syntax

```
TRIANGLE([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) or TRIANGLE([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) or TRIANGLE([G], [[x/y coordinate matrix]], [[color matrix]], {[[z matrix]], [zcode], [[[projection matrix]]], [zstring]) or TRIANGLE([G])
```

Description

Draws a triangle between specified cartesian coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes.

The next form of TRIANGLE allows display of multiple triangles at a time.
This is mostly used if you have a set of vertices and want to display them all at once.

The first 2 matrices indicate the x/y coordinates and colors of each points. TRIANGLE will draw 1 quadrilateral for each set of 4 adjacent vertices and blends the colors associated with the 4 points.
If a z and projection matrix are provided, for each point, this matrix is multiplied by the [x,y,z,1] vector to create the display x,y coordinates.
If zcode is a list that contains 3 real numbers { ex, ey, ez } then x,y are further modified by doing x=ez/z*x-ex and y=ez/z*y-ey creating a perspective projection.
If zstring is provided, z clipping will happen using the z value (see below).

If zcode="N" or is a list that starts with "N", then each z is normalized to be between 0 and 255.

About ZString
TRIANGLE([G]) returns a string adapted for z clipping.

To use Z clipping, call TRIANGLE to create a Z clipping string (initialized at 255 for each pixels). You can then call TRIANGLE with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate.

Example

```
TRIANGLE(0,0,5,5,5,-5,#FFh,#FF00h,#FF0000h,128)
```

discussion

TRIANGLE_P

#link

Syntax

TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) or TRIANGLE_P([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) or TRIANGLE_P([G], [[x/y coordinate matrix]], [[color matrix]], {[[z matrix]], [zcode], [[[projection matrix]]], [zstring]) or TRIANGLE_P([G])

Description

Draws a triangle between specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes.

The next form of TRIANGLE allows display of multiple triangles at a time.
This is mostly used if you have a set of vertices and want to display them all at once.

The first 2 matrices indicate the x/y coordinates and colors of each points. TRIANGLE_P will draw 1 quadrilateral for each set of 4 adjacent vertices and blends the colors associated with the 4 points.
If a z and projection matrix are provided, for each point, this matrix is multiplied by the [x,y,z,1] vector to create the display x,y coordinates.
If zcode is a list that contains 3 real numbers { ex, ey, ez } then x,y are further modified by doing x=ez/z*x-ex and y=ez/z*y-ey creating a perspective projection.
If zstring is provided, z clipping will happen using the z value (see below).
If zcode="N" or is a list that starts with "N", then each z is normalized to be between 0 and 255.

About ZString
TRIANGLE_P([G]) returns a string adapted for z clipping.

To use Z clipping, call TRIANGLE_P to create a Z clipping string (initialized at 255 for each pixels). You can then call TRIANGLE_P with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE_P will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate.

Example

```
TRIANGLE_P(0,20,150,50,100,100,#FFh,#FF00h,#FF0000h,128)
```

discussion

trig2exp

#link

Syntax

```
trig2exp(Expr)
```

Description

Replaces trigonometric functions in Expr with complex exponentials(without linearization).

Example

```
trig2exp(sin(x)) returns (exp(i*x)-(1/exp(i*x)))/(2*i)
```

discussion

trigcos

#link

Syntax

```
trigcos(Expr)
```

Description

Simplifies the argument Expr using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x) (privileging cosine)

Example

```
trigcos(sin(x)^4+sin(x)^2) returns cos(x)^4-3*cos(x)^2+2
```

discussion

trigexpand

#link

Syntax

```
trigexpand(Expr)
```

Description

Expands trigonometric functions.

Example

```
trigexpand(sin(3*x)) returns (4*cos(x)^2-1)*sin(x)
```

discussion

trigsin

#link

Syntax

`trigsin(Expr)`

Description

Simplifies the argument Expr using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x) (privileging sine)

Example

`trigsin(cos(x)^4+sin(x)^2) returns sin(x)^4-sin(x)^2+1`

discussion

#link

trigtan

Syntax

`trigtan(Expr)`

Description

Simplifies the argument Expr using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x) (privileging tangent)

Example

```
trigtan(cos(x)^4+sin(x)^2) returns
(tan(x)^4+tan(x)^2+1)/(tan(x)^4+2*tan(x)^2+1)
```

discussion

#link

TRN

Syntax

`TRN(matrix)`

Description

Transpose. Transposes matrix. If Complex mode is on and the matrix contains complex elements, then TRN finds the conjugate transpose.

Example

`TRN([[1,2],[3,4]]) returns [[1,3],[2,4]]`

discussion

#link

trunc

Syntax

`trunc(Real, [Integer]) or trunc(List, [Integer])`

Description

Truncates a value to n decimal places (by default n=0).Accepts complex numbers.

Example

```
trunc(4.3) returns 4
trunc({3.25, 8.71, 9.01},1) returns {3.2, 8.7, 9.}
```

discussion

TRUNCATE

#link

Syntax

```
TRUNCATE(value, [places])
```

Description

Truncates value to system display settings. If optional places is given, truncates value to places decimal places. If places is negative, truncates to significant digits instead.

Example

```
TRUNCATE(2.3678,2) returns 2.36
```

discussion

tsimplify

#link

Syntax

```
tsimplify(Expr)
```

Description

Returns an expression with transcendentals rewritten as complex exponentials

Example

```
tsimplify(exp(2*x)+exp(x)) returns exp(x)^2+exp(x)
```

discussion

type

#link

Syntax

```
type(Expr)
```

Description

Returns n in [1..12] that defines the type of the argument

Example

```
type("abc") returns DOM_STRING
```

discussion

TYPE

#link

Syntax

```
TYPE(object)
```

Description

Returns the type of the object:
0: Real
1: Integer
2: String
3: Complex
4: Matrix
5: Error
6: List
8: Function
9: Unit
14.?: cas object. the fractional part is the cas type

discussion

#link

UFACTOR

Syntax

```
UFACTOR(Value_Unit1, 1_Unit2)
```

Description

Unit factor conversion.
Converts a measurement using a compound unit into a measurement expressed in constituent units.

Example

```
a Coulomb—a measure of electric charge—is a compound unit derived from
the SI base units of Ampere and second: 1 C = 1 A * 1 s.  Using UFACTOR,
you can express a measurement in Coulombs as a product of Amperes and
time.

UFACTOR(100_C,1_A)) returns 100_A*s
UFACTOR(100_C, 1_min) returns 1.66666666667_min*A
```

discussion

#link

unapply

Syntax

```
unapply(Expr,Var)
```

Description

Returns a function defined by an expression.

Example

```
unapply(2*x^2,x) returns  (x)->2*x^2
```

discussion

UNCHECK

#link

Syntax

```
UNCHECK(n)
```

Description

Unchecks (deselects) the corresponding symbolic definition field in the current app. The integer n must be between 0 and 9 for most apps. For Statistics 1-Var and Statistics 2-Var apps, n must be between 1 and 5.

For example, UNCHECK(3) would uncheck F3 if the current app is Function.

discussion

UNTIL

#link

Syntax

```
REPEAT command(s) UNTIL test;
```

Description

executes command(s) UNTIL the test is true.

```
A:=5;
REPEAT
    PRINT(A);
   A:= A-1;
UNTIL A<1;
```

will print 5 4 3 2 1

discussion

USIMPLIFY

#link

Syntax

```
USIMPLIFY(Value_Unitsexpr)
```

Description

Unit simplification.
Simplifies Value in a complex unit expression Unitsexpr to an equivalent value in a simpler unit expression.

Example

```
a Joule is defined as 1 kg*m^2/s^2.


USIMPLIFY(5_kg*m2/s2) returns 5_J
```

discussion

valuation

#link

Syntax

```
valuation(Poly(P))
```

Description

Returns the valuation (degree of the term of lowest degree) of the polynomial P .

Example

```
valuation(x^4+x^3) returns 3
```

discussion

vandermonde

#link

Syntax

```
vandermonde(Vect(V))
```

Description

Returns the Vandermonde matrix=[V^0,V^1,..]

Example

```
vandermonde([1,2,3]) returns [[1,1,1],[1,2,4],[1,3,9]]
```

discussion

variance

#link

Syntax

```
variance(Lst||Mtrx,[Lst])
```

Description

Returns the variance of a list with the second argument as the weight, or the list of variance of the columns of a matrix.

Example

```
variance([3,4,2]) returns 2/3
```

discussion

vector

#link

Syntax

```
vector(Pnt,Pnt || Pnt,Vect)
```

Description

Defines a vector(origin is 0 if 1 arg) with two points or two components or two affix (for

2D) or with a point and a vector or with a point (its extrmity and its origin is [0,0,0]). discussion

vertices

#link

Syntax

```
vertices(Polygon or Polyedr(P))
```

Description

Returns the list of the vertices of the polygon or polyhedron P.

discussion

vertices_abca

#link

Syntax

```
vertices_abca(Polygon or Polyedr(P))
```

Description

Returns the closed list [A,B,...A] of the vertices of the polygon or polyhedron P.

discussion

VIEW

#link

Syntax

```
VIEW "text", Function()
```

Description

VIEW. Allows a programmer to customize the Views menu. Causes "text" to apear when VIEW key is pressed and Function to be executed when the OK menu key (or ENTER key) is pressed.

discussion

vpotential

#link

Syntax

```
vpotential(Vect(V),LstVar)
```

Description

Returns U such as curl(U)=V

Example

```
vpotential([2*x*y+3,x^2-4*z,-2*y*z],[x,y,z]) returns [0,-2*x*y*z,-x^3/3+4*x*z+3*y]
```

discussion

WAIT

#link

Syntax

```
WAIT(n)
```

Description

Halts program execution for the specified number of seconds.
If n is omitted or 0, halts execution until the user presses a key and returns the keycode (or -1 after 1 minute).

If n is -1, halts executions until the user presses a key or there is a mouse event.
If a key is pressed, the keycode is returned.
After a 1 minute timeout, returns -1

If a mouse event happends, a list of the form { type, [x, y], [dx, dy] } is returned. Normally x/y is the event position unless otherwise indicated.
Type can be:
0: Mouse Down
1: Mouse Move
2: Mouse Up (x/y is not provided)
3: Mouse Click (note, if a click is detected, there is no MouseUp)
5: Mouse Stretch. x/y is the delta since the last event. dx/dy is the delta since the ORIGINAL mouse down...
6: Mouse Rotate, x is original angle, y is new angle in 32nd of a circle.
7: Mouse Long Click, This means that the mouse stayed down for 1 second...

Example

```
WAIT(5) halts program execution for 5 seconds.
```

discussion

when

#link

Syntax

```
when(Cond,Expr1,Expr2)
```

Description

If condition (even symbolic) returns expr1 else returns expr2 (? is the infixed version of when).

discussion

WHILE

#link

Syntax

```
WHILE test DO command(s) END;
```

Description

executes command(s) WHILE the test is true.

```
A:=5;
WHILE A>1 DO
    PRINT(A);
   A:= A-1;
END;
```

will print 5 4 3 2 1

discussion

#link

white

Syntax

`white(Opt)`

Description

Option of the display command to display with color.

discussion

#link

XOR

Syntax

`Value1 XOR Value2`

Description

Exclusive OR.
Returns 1 if either Value1 or Value2 is non-zero but not both; otherwise, returns 0.

Example

`3 XOR 2 returns 0`

discussion

#link

XPON

Syntax

`XPON(value)`

Description

Exponent. Returns the exponent of value.

Example

`XPON(123.4) returns 2`

discussion

#link

yellow

Syntax

```
('display')=[color]
```

Description

For example, suppose you have drawn a circle in the Geometry app. In Symbolic view, the circle's definition might be GC:=circle(GA,GB-GA). If you wanted that circle to be, say, red, you could modify that definition to read:

Example

```
GC:=circle(GA,GB-GA, ('display')=red)
```

discussion

zeros

#link

Syntax

```
zeros(Expr,[Var])
```

Description

Returns the zeros (reals or complex according to the CAS settings) of the expression Expr for the variable Var (or the matrix where the lines are the solutions of the system : Expr1=0,Expr2=0...).

Example

```
zeros(x^2+4) returns [] in real mode and [-2*i,2*i] in complex mode
```

discussion

Zeta

#link

Syntax

```
Zeta(Real(a))
```

Description

Returns if a>1 sum(1/n^a,n,1,∞)

Example

```
Zeta(2) returns π^2/6
```

discussion

zip

#link

Syntax

```
zip(Fnc2d(f),Lst(l1),Lst(l2),[Val(default)])
```

Description

Returns a list whose j-th entry is f(l1[j],l2[j]): without default value its length is the minimum of the lengths of the two input lists and else the shorter list is padded with the default value.

Example

```
zip('+',[a,b,c,d], [1,2,3,4]) returns [a+1,b+2,c+3,d+4]
```

discussion

ztrans

#link

Syntax

```
ztrans(Expr,[Var],[ZtransVar])
```

Description

Z transform of a sequence.

Example

```
ztrans(a^x)
```

discussion