

TI-Nspire™ CX CAS Reference Guide

Important Information

Except as otherwise expressly stated in the License that accompanies a program, Texas Instruments makes no warranty, either express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis. In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the amount set forth in the license for the program. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

© 2020 Texas Instruments Incorporated

Actual products may vary slightly from provided images.

Contents

A	. 8 . 17 . 20 . 44 . 57 . 76 . 86 . 94 109
A B C C D E F C C C C C C C C C C C C C C C C C C	. 8 . 17 . 20 . 44 . 57 . 76 . 86 . 94 109
B	. 17 . 20 . 44 . 57 . 67 . 76 . 86 . 94 117
C	. 20 . 44 . 57 . 76 . 86 . 94 109
D	. 44 . 57 . 76 . 86 . 94 109
E	. 57 . 67 . 76 . 86 . 94 109
F	. 67 . 76 . 86 . 94 109
G	. 76 . 86 . 94 109 117
I	.86 . 94 109 117
L M	. 94 109 117
M N O P Q R S T U V W X	109 117
N	117
O	
P	
Q	129
R	
S	
T	
U	
V	
W	
X	
z	
	199
Symbols 2	207
TI-Nspire™ CX II - Draw Commands2	233
Graphics Programming	233
Graphics Screen	
Default View and Settings	
Graphics Screen Errors Messages	
Invalid Commands While in Graphics Mode	
C	
D	237
F	240
G	
P	
S	
U	

Empty (Void) Elements	248
Shortcuts for Entering Maths Expressions	250
EOS™ (Equation Operating System) Hierarchy	252
TI-Nspire CX II - TI-Basic Programming Features	254
Auto-indentation in Programming Editor Improved Error Messages for TI-Basic	
Constants and Values	257
Error Codes and Messages	258
Warning Codes and Messages	266
General Information	268
Online Help	268
Contact TI Support	268
Service and Warranty Information	268
Index	260

Expression Templates

Expression templates give you an easy way to enter maths expressions in standard mathematical notation. When you insert a template, it appears on the entry line with small blocks at positions where you can enter elements. A cursor shows which element you can enter.

Use the arrow keys or press tab to move the cursor to each element's position, and type a value or expression for the element. Press enter or ctrl enter to evaluate the expression.

Fraction template		ctrl 🗦 keys
<u> </u>	Example:	
Note: See also / (divide), page 209.	12 8:2	$\frac{3}{4}$

Exponent template		^ key
nΩ	Example:	
ш	2 ³	8
Note: Type the first value, press , and then type the exponent. To return the cursor		
to the baseline, press right arrow ().		

Note: See also ^ (power), page 210.

Square root template		ctrl x² keys
Note: See also $\sqrt{\ }$ () (square root), page 220.	Example: $ \sqrt{4} $ $ \sqrt{9,a,4} $ $ \sqrt{4} $ $ \sqrt{4} $ $ \sqrt{9,16,4} $	$ \begin{array}{c} 2 \\ \hline $

Nth root template



Note: See also root(), page 151.

Example:

3√8	2
³ √{8,27, <i>b</i> }	$\left[\frac{1}{3},\frac{1}{3}\right]$

e exponent template

ex keys



Natural exponential *e* raised to a power

Note: See also e^(), page 57.

Example:

e ¹	е
e 1.	2.71828182846

Log template

ctrl 10x kev



Calculates log to a specified base. For a default of base 10, omit the base.

Note: See also log(), page 105.

Example:

$\log_4(2.)$	0.5

Piecewise template (2-piece)

Catalogue >

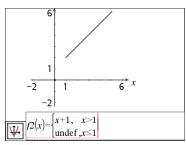




Lets you create expressions and conditions for a two-piece piecewise function. To add a piece, click in the template and repeat the template.

Note: See also piecewise(), page 130.

Example:



Piecewise template (N-piece)





Lets you create expressions and conditions for an

N-piece piecewise function. Prompts for N.

Create Piecewise Function Piecewise Function Number of function pieces 3 \$ OK Cancel

Note: See also piecewise(), page 130.

Example:

See the example for Piecewise template (2-piece).

System of 2 equations template



Creates a system of two equations. To add a row to an existing system, click in the template and repeat the template.

Note: See also system(), page 179.

Example:

$$solve \begin{cases} x+y=0 \\ x-y=5 \end{cases}, x,y \qquad x=\frac{5}{2} \text{ and } y=\frac{-5}{2}$$

$$solve \begin{cases} y=x^2-2 \\ x+2 \cdot y=-1 \end{cases}, x,y \end{cases}$$

$$x=\frac{-3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$$

System of N equations template

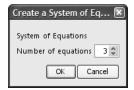




Lets you create a system of Neguations. Prompts for N.

Example:

See the example for System of equations template (2-equation).



Note: See also system(), page 179.

Absolute value template





Note: See also abs(), page 8.

Example:

Absolute value template Catalogue > {2,3,4,64} 2,-3,4,-43 dd°mm'ss.ss" template Catalogue > 0[]![]!! Example: 30°15'10" Lets you enter angles in dd°mm'ss.ss" $10891 \cdot \pi$ format, where dd is the number of decimal 64800 degrees, mm is the number of minutes, and ss.ss is the number of seconds. Catalogue > Matrix template (2 x 2) Example: 2 $2 \cdot a$ $3 \cdot a \quad 4 \cdot a$ Creates a 2 x 2 matrix. Matrix template (1 x 2) Catalogue > Example: [00]crossP[[1 2],[3 4]] $[0 \ 0 \ -2]$ Matrix template (2 x 1) Catalogue > Example: 5 | .0.01 0.05 8 0.08

Example:

diag||1

2 3

7

Catalogue >

 $[4 \ 2 \ 9]$

and columns.

Matrix template (m x n)

The template appears after you are

prompted to specify the number of rows

Matrix template (m x n)





Note: If you create a matrix with a large number of rows and columns, it may take a few moments to appear.

Sum template (Σ)

Catalogue >



25



Note: See also Σ () (sumSeq), page 221.

Example:

7	
	[]
	$\langle n \rangle$
n=3	

Product template (Π)

Catalogue >





Note: See also Π () (prodSeq), page 220.

Example:

	5		
		Г	$\lfloor 1 \rfloor$
			$\lfloor n \rfloor$
n	=	1	

First derivative template

Catalogue >





The first derivative template can also be used to calculate first derivative at a point.

Note: See also d() (derivative), page 218.

Example:



 $3 \cdot x^2$

$$\frac{d}{dx}(x^3)|_{x=3}$$

Second derivative template

Catalogue >

$$\frac{d^2}{d\Box^2}(\Box)$$

The second derivative template can also be used to calculate second derivative at a point.

Note: See also d() (derivative), page 218.

Example:

$\frac{d^2}{dx^2}(x^3)$	6.:
2	

 $\frac{d^2}{dx^2} \left(x^3\right) | x = 3$ 18

Nth derivative template

Catalogue >



The *n*th derivative template can be used to calculate the nth derivative.

Note: See also d() (derivative), page 218.

Example:



Definite integral template







Note: See also() integral(), page 218.

Example:

$$\int_{a}^{b} x^{2} dx \qquad \frac{b^{3}}{3} \frac{a^{3}}{3}$$

Indefinite integral template

Catalogue >



Note: See also ∫() integral(), page 218.

Example:

$$\int x^2 dx \qquad \qquad \underline{x}^2$$

Limit template

Catalogue >





Example:

$$\overline{\lim_{x\to 5} (2 \cdot x + 3)} \qquad 13$$



Use — or (—) for left hand limit. Use + for right hand limit.

Note: See also limit(), page 96.

Alphabetical Listing

Items whose names are not alphabetic (such as +, ! and >) are listed at the end of this section, starting page 207. Unless otherwise specified, all examples in this section were performed in the default reset mode, and all variables are assumed to be undefined.

Α

abs()		Catalogue > 🗐
abs(Expr1) ⇒expression	$\left\{\frac{\pi}{2}, \frac{-\pi}{3}\right\}$	$\left\{\frac{\pi}{n},\frac{\pi}{n}\right\}$
abs(List1)⇒list	$\frac{ [2'3] }{ 2-3\cdot i }$	$\frac{\left(\begin{array}{c}2^33\end{array}\right)}{\sqrt{13}}$
$abs(Matrix l) \Rightarrow matrix$		
Returns the absolute value of the argument.	$\frac{ x+y\cdot i }{}$	$\sqrt{x^2+y^2}$

Note: See also Absolute value template, page 3.

If the argument is a complex number, returns the number's modulus.

Note: All undefined variables are treated as real variables.

amortTbl()	Catalogue > 🗐
amortTbl(NPmt,N,I,PV, [Pmt], [FV],	amortTbl(12,60,10,5000,12,12)

amortTbl(NPmt,N,I,PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [roundValue]) $\Rightarrow matrix$

Amortisation function that returns a matrix as an amortisation table for a set of TVM arguments.

NPmt is the number of payments to be included in the table. The table starts with the first payment.

N, I, PV, Pmt, FV, PpY, CpY and PmtAt are described in the table of TVM arguments, page 192.

- If you omit Pmt, it defaults to Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt).
- If you omit FV, it defaults to FV=0.

mortTbl(12,60,10,5000,,,12,12)					
	0	0.	0.	5000.	
	1	$^{-}41.67$	-64.57	4935.43	
	2	-41.13	-65.11	4870.32	
	3	-40.59	-65.65	4804.67	
	4	$^{-40.04}$	-66.2	4738.47	
	5	-39.49	-66.75	4671.72	
	6	-38.93	-67.31	4604.41	
	7	-38.37	-67.87	4536.54	
	8	-37.8	$^{-}68.44$	4468.1	
	9	-37.23	-69.01	4399.09	
	10	-36.66	-69.58	4329.51	
	11	-36.08	-70.16	4259.35	
	12	-35.49	-70.75	4188.6	

Catalogue > 🕮

The defaults for PpY, CpY and PmtAtare the same as for the TVM functions.

roundValue specifies the number of decimal places for rounding. Default=2.

The columns in the result matrix are in this order: Payment number, amount paid to interest, amount paid to principal, and balance.

The balance displayed in row n is the balance after payment n.

You can use the output matrix as input for the other amortisation functions Σ **Int()** and Σ Prn(), page 222, and bal(), page 17.

Catalogue > 🕮 and

BooleanExpr1 and $BooleanExpr2 \Rightarrow Boolean\ expression$

 $x \ge 3$ and $x \ge 4$ $\{x \ge 3, x \le 0\}$ and $\{x \ge 4, x \le -2\}$

BooleanList1 and BooleanList2⇒Boolean list

BooleanMatrix1 and BooleanMatrix2⇒Boolean matrix

Returns true or false or a simplified form of the original entry.

*Integer1***and***Integer2*⇒*integer*

Compares two real integers bit-by-bit using an and operation. Internally, both integers are converted to signed, 64-bit binary numbers. When corresponding bits are compared, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value represents the bit results and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

In Hex base mode:

0h7AC36 and 0h3D5F

0h2C16

Important: Zero, not the letter O.

In Bin base mode:

0b100101 and 0b100

0b100

In Dec base mode:

37 and 0b100 4

Note: A binary entry can have up to 64 digits (not counting the 0b prefix). A hexadecimal entry can have up to 16 digits.

angle() Catalogue > [3]

$angle(Expr1) \Rightarrow expression$

Returns the angle of the argument, interpreting the argument as a complex number.

Note: All undefined variables are treated as real variables.

In Degree angle mode:

$$angle(0+2\cdot i) 90$$

In Gradian angle mode:

$$angle(0+3\cdot i)$$
 100

In Radian angle mode:

$$\frac{\pi}{4}$$

$$\operatorname{angle}(z) \qquad \frac{\pi}{2}$$

$$\operatorname{angle}(x+i\cdot y) \qquad \frac{\pi \cdot \operatorname{sign}(z)-1}{2}$$

$$\frac{\pi \cdot \operatorname{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)$$

angle
$$\left\{ 1+2\cdot i, 3+0\cdot i, 0-4\cdot i \right\}$$
 $\left\{ \frac{\pi}{2} - \tan^{-1} \left(\frac{1}{2} \right), 0, \frac{-\pi}{2} \right\}$

$angle(List1) \Rightarrow list$

$angle(Matrix 1) \Rightarrow matrix$

Returns a list or matrix of angles of the elements in *List1* or *Matrix1*, interpreting each element as a complex number that represents a two-dimensional rectangular coordinate point.

ANOVA Catalogue > 🗐

ANOVA List1,List2[,List3,...,List20][,Flag]

ANOVA

Performs a one-way analysis of variance for comparing the means of two to 20 populations. A summary of results is stored in the *stat.results* variable (page 174).

Flag=0 for Data, Flag=1 for Stats

Output variable	Description
stat.F	Value of the F statistic
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.df	Degrees of freedom of the groups
stat.SS	Sum of squares of the groups
stat.MS	Mean squares for the groups
stat.dfError	Degrees of freedom of the errors
stat.SSError	Sum of squares of the errors
stat.MSError	Mean square for the errors
stat.sp	Pooled standard deviation
stat.xbarlist	Mean of the input of the lists
stat.CLowerList	95% confidence intervals for the mean of each input list
stat.CUpperList	95% confidence intervals for the mean of each input list

ANOVA2way

Catalogue > 23

ANOVA2way List1,List2[,List3,...,List10][,levRow]

Computes a two-way analysis of variance for comparing the means of two to 10 populations. A summary of results is stored in the *stat.results* variable (page 174).

LevRow=0 for Block

LevRow=2,3,...,Len-1, for Two Factor, where Len=length(List1)=length(List2) = ... = length(List10) and Len / LevRow $\in \{2,3,...\}$

Outputs: Block Design

Output variable	Description
stat.F	F statistic of the column factor

Output variable	Description
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.df	Degrees of freedom of the column factor
stat.SS	Sum of squares of the column factor
stat.MS	Mean squares for column factor
stat.FBlock	F statistic for factor
stat.PValBlock	Least probability at which the null hypothesis can be rejected
stat.dfBlock	Degrees of freedom for factor
stat.SSBlock	Sum of squares for factor
stat.MSBlock	Mean squares for factor
stat.dfError	Degrees of freedom of the errors
stat.SSError	Sum of squares of the errors
stat.MSError	Mean squares for the errors
stat.s	Standard deviation of the error

COLUMN FACTOR Outputs

Output variable	Description
stat.FcoI	F statistic of the column factor
stat.PValCoI	Probability value of the column factor
stat.dfCoI	Degrees of freedom of the column factor
stat.SSCol	Sum of squares of the column factor
stat.MSCol	Mean squares for column factor

ROW FACTOR Outputs

Output variable	Description
stat.FRow	F statistic of the row factor
stat.PValRow	Probability value of the row factor
stat.dfRow	Degrees of freedom of the row factor
stat.SSRow	Sum of squares of the row factor
stat.MSRow	Mean squares for row factor

INTERACTION Outputs

Output variable	Description
stat.FInteract	F statistic of the interaction
stat.PValInteract	Probability value of the interaction
stat.dfInteract	Degrees of freedom of the interaction
stat.SSInteract	Sum of squares of the interaction
stat.MSInteract	Mean squares for interaction

ERROR Outputs

Output variable	Description
stat.dfError	Degrees of freedom of the errors
stat.SSError	Sum of squares of the errors
stat.MSError	Mean squares for the errors
s	Standard deviation of the error

Ans		ctrl (-) keys
Ans ⇒ <i>value</i>	56	56
Returns the result of the most recently evaluated expression.	56+4	60
	60+4	64

approx() Catalogue > 23

$approx(Expr1) \Rightarrow expression$

Returns the evaluation of the argument as an expression containing decimal values, when possible, regardless of the current Auto or Approximate mode.

This is equivalent to entering the argument and pressing ctrl enter.

 $approx(List1) \Rightarrow list$

 $approx(Matrix 1) \Rightarrow matrix$

Returns a list or *matrix* where each element has been evaluated to a decimal value, when possible.

	catalogue > Q
$\operatorname{approx}\left(\frac{1}{3}\right)$	0.333333
$\operatorname{approx}\left\{\left\{\frac{1}{3},\frac{1}{9}\right\}\right\}$	{0.333333,0.111111}
$\operatorname{approx}(\{\sin(\pi),\cos(\pi)$	}) {0.,-1.}
$approx([\sqrt{2} \sqrt{3}])$	[1.41421 1.73205]
$\operatorname{approx}\left[\begin{bmatrix} \frac{1}{3} & \frac{1}{9} \end{bmatrix}\right]$	[0.333333 0.111111]
77 7 5 7 5	
$approx(\{sin(\pi),cos(\pi)\}$	
$approx([\sqrt{2} \sqrt{3}])$	[1.41421 1.73205]

Catalogue > [3] ▶approxFraction() *Expr* ▶*approxFraction* 0.833333 $+\frac{1}{3}$ +tan(π) $([Tol]) \Rightarrow expression$ 0.833333333333333 ▶ approxFraction(5. E-14) $List \triangleright approxFraction([Tol]) \Rightarrow list$ $Matrix \triangleright approxFraction([Tol]) \Rightarrow matrix$ {π,1.5} ▶approxFraction(5.ε-14) Returns the input as a fraction, using a tolerance of Tol. If Tol is omitted, a tolerance of 5.E-14 is used. Note: You can insert this function from the computer keyboard by typing @>approxFraction(...). approxRational() Catalogue > [3] $approxRational(Expr[. Tol]) \Rightarrow expression$

approximational(Expr[, 101]) = expression	approxRational $(0.333,5\cdot10^{-5})$ 333
approxRational($List[, Tol]$) $\Rightarrow list$	1000
approximation (=)	approxRational({0.2,0.33,4.125},5.e-14)
approxRational($Matrix[, Tol]$) $\Rightarrow matrix$	$\left\{\frac{1}{5}, \frac{33}{100}, \frac{33}{8}\right\}$
Returns the argument as a fraction using a tolerance of <i>Tol</i> . If <i>Tol</i> is omitted, a	5,100,8

arccos()	See cos ⁻¹ (), page 31.

arccosh()		See cosh ⁻¹ (),	page 32.

arccoth()	See coth⁻¹(), page 34.

tolerance of 5.E-14 is used.

arccsch()

See csch⁻¹(), page 37.

arcLen()

 $arcLen(Expr1, Var, Start, End) \Rightarrow expression$

Returns the arc length of Expr1 from Start to End with respect to variable Var.

Arc length is calculated as an integral assuming a function mode definition.

 $arcLen(List1, Var, Start, End) \Rightarrow list$

Returns a list of the arc lengths of each element of *List1* from *Start* to *End* with respect to *Var*.

Catalogue > ্রি

$$\frac{\operatorname{arcLen}(\cos(x), x, 0, \pi)}{\operatorname{arcLen}(f(x), x, a, b)} \frac{3.8202}{\int_{a}^{b} \sqrt{\left(\frac{d}{dx}(f(x))\right)^{2} + 1} dx}$$

$$\frac{\operatorname{arcLen}(\{\sin(x),\cos(x)\},x,0,\pi)}{\{3.8202,3.8202\}}$$

arcsec()

See sec⁻¹(), page 155.

arcsech()

See sech⁻¹(), page 155.

arcsin()

See sin⁻¹(), page 165.

arcsinh()

See sinh⁻¹(), page 166.

arctan()

See tan-1(), page 180.

augment() Catalogue > 🗐

 $augment(List1, List2) \Rightarrow list$

the end of List1.

augment({1,-3,2},{5,4}) {1,-3,2,5,4} Returns a new list that is *List2* appended to

 $augment(Matrix1, Matrix2) \Rightarrow matrix$

Returns a new matrix that is *Matrix2* appended to Matrix 1. When the "." character is used, the matrices must have equal row dimensions, and Matrix2 is appended to *Matrix1* as new columns. Does not alter Matrix 1 or Matrix 2.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1$		1	2 4
$\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow m2$			5 6
$\frac{1}{\text{augment}(m1, m2)}$	1	2	5
	_3	4	6]

avgRC() Catalogue > 23

avgRC(Expr1, Var [=Value])*Step*])⇒*expression*

avgRC(Expr1, Var [=Value][,List1]**)**⇒list

 $avgRC(List1, Var [=Value] [, Step]) \Rightarrow list$

avgRC(Matrix 1, Var [=Value])Step]**)**⇒matrix

Returns the forward-difference quotient (average rate of change).

Expr1 can be a user-defined function name (see Func).

When *Value* is specified, it overrides any prior variable assignment or any current "|" substitution for the variable.

Step is the step value. If Step is omitted, it defaults to 0.001.

Note that the similar function centralDiff() uses the central-difference quotient.

	cutulogue > s _{ta}
$\operatorname{avgRC}(f(x),x,h)$	f(x+h)-f(x)
	h
$\operatorname{avgRC}(\sin(x),x,h) x=2$	$\sin(h+2)-\sin(2)$
	h
$\operatorname{avgRC}(x^2-x+2,x)$	2.·(x-0.4995)
$\operatorname{avgRC}(x^2-x+2,x,0.1)$	2.·(<i>x</i> -0.45)
$\operatorname{avgRC}(x^2-x+2,x,3)$	2·(x+1)

bal() Catalogue > [3]

bal(NPmt,N,I,PV,[Pmt],[FV],[PpY],[CpY], [PmtAt], [roundValue]) $\Rightarrow value$

bal(NPmt,amortTable)⇒value

Amortisation function that calculates schedule balance after a specified payment.

N, I, PV, Pmt, FV, PpY, CpY and PmtAtare described in the table of TVM arguments, page 192.

NPmt specifies the payment number after which you want the data calculated.

N, I, PV, Pmt, FV, PpY, CpY and PmtAt are described in the table of TVM arguments, page 192.

- If you omit Pmt, it defaults to Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt).
- If you omit FV, it defaults to FV=0.
- The defaults for PpY, CpY and PmtAtare the same as for the TVM functions.

roundValue specifies the number of decimal places for rounding. Default=2.

bal(NPmt,amortTable) calculates the balance after payment number NPmt. based on amortisation table amort Table. The amortTable argument must be a matrix in the form described under amortTbl(), page 8.

Note: See also Σ **Int()** and Σ **Prn()**, page 222.

computer keyboard by typing @>Base2.

bal(5,6,5.75,50	000	,,12,12)		833.11
tbl:=amortTbl	6,6	,5.75,50	00,,12,12)	
	0	0.	0.	5000.
	1	-23.35	-825.63	4174.37
	2	$^{-}19.49$	-829.49	3344.88
	3	-15.62	-833.36	2511.52
	4	-11.73	-837.25	1674.27
	5	-7.82	-841.16	833.11
	6	-3.89	-845.09	-11.98
bal(4,tbl)				1674.27

▶Base2		Catalogue > 🕎
Integer1 ▶Base2⇒integer	256▶Base2	0b100000000
Note: You can insert this operator from the	0h1F▶Base2	0b11111

Converts *Integer 1* to a binary number. Binary or hexadecimal numbers always have a 0b or 0h prefix, respectively. Use a zero, not the letter O, followed by b or h.

0b binaryNumber

Oh hexadecimalNumber

A binary number can have up to 64 digits. A hexadecimal number can have up to 16.

Without a prefix, *Integer 1* is treated as decimal (base 10). The result is displayed in binary, regardless of the Base mode.

Negative numbers are displayed in "two's complement" form. For example,

-1 is displayed as Ohfffffffffffff in Hex base mode 0b111...111 (64 1's) in Binary base mode

-263 is displayed as 0h8000000000000000 in Hex base mode 0b100...000 (63 zeroes) in Binary base mode

If you enter a decimal integer that is outside the range of a signed, 64-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range. Consider the following examples of values outside the range.

263 becomes -263 and is displayed as 0h8000000000000000 in Hex base mode 0b100...000 (63 zeroes) in Binary base mode

264 becomes 0 and is displayed as

0h0 in Hex base mode

0b0 in Binary base mode

-263 - 1 becomes 263 - 1 and is displayed as

0b111...111 (64 1's) in Binary base mode

▶Base10 Catalogue > \bigcirc Integer l ▶Base10⇒integer 0b10011▶Base10 19 Note: You can insert this operator from the 0h1F▶Base10 31

Converts *Integer 1* to a decimal (base 10) number. A binary or hexadecimal entry must always have a 0b or 0h prefix, respectively.

computer keyboard by typing @>Base10.

0b binaryNumber

Oh hexadecimalNumber

Zero, not the letter O, followed by b or h.

A binary number can have up to 64 digits. A hexadecimal number can have up to 16.

Without a prefix, *Integer1* is treated as decimal. The result is displayed in decimal, regardless of the Base mode.

▶Base16	Catalogue >		
<i>Integer1</i> ▶Base16⇒ <i>integer</i>	256▶Base16	0h100	
Note: You can insert this operator from the	0b111100001111▶Base16	OhFOF	

Note: You can insert this operator from the computer keyboard by typing @>Base16.

Converts *Integer1* to a hexadecimal number. Binary or hexadecimal numbers always have a 0b or 0h prefix, respectively.

0b binaryNumber

Oh hexadecimal Number

Zero, not the letter O, followed by b or h.

A binary number can have up to 64 digits. A hexadecimal number can have up to 16.

Without a prefix, *Integer1* is treated as decimal (base 10). The result is displayed in hexadecimal, regardless of the Base mode.

▶Base16

Catalogue > 🗐

If you enter a decimal integer that is too large for a signed, 64-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range. For more information, see **>Base2**, page 17.

binomCdf()

Catalogue > 📳

 $binomCdf(n,p) \Rightarrow list$

binomCdf(*n*,*p*,*lowBound*,*upBound***)**⇒*number* if *lowBound* and *upBound* are numbers, *list* if *lowBound* and *upBound* are lists

binomCdf(n,p,upBound)for P($0 \le X \le upBound$) $\Rightarrow number$ if upBound is a number, list if upBound is a list

Computes a cumulative probability for the discrete binomial distribution with n number of trials and probability p of success on each trial.

For $P(X \le upBound)$, set lowBound=0

binomPdf()

Catalogue > 🗐

 $binomPdf(n,p) \Rightarrow list$

binomPdf $(n,p,XVal) \Rightarrow number$ if XVal is a number, list if XVal is a list

Computes a probability for the discrete binomial distribution with n number of trials and probability p of success on each trial.

C

ceiling()

Catalogue > 🗐

 $ceiling(Expr1) \Rightarrow integer$

ceiling(.456)

1

Returns the nearest integer that is \geq the argument.

The argument can be a real or a complex number.

Note: See also floor().

ceiling()

ceiling(Listl) $\Rightarrow list$ ceiling(Matrixl) $\Rightarrow matrix$

Returns a list or matrix of the ceiling of each element.

(Cata	logue	>	ij.

$ceiling({-3.1,1,2.5})$	{-3.,1,3.}
ceiling $\begin{bmatrix} 0 & -3.2 \cdot i \end{bmatrix}$	0 -3.·i
1.3 4	[2. 4]

centralDiff()

centralDiff(Expr1,Var[=Value][,Step]**)** \Rightarrow expression

centralDiff(Expr1,Var[,Step]**)** $|Var=Value \Rightarrow expression$

centralDiff(Expr1,Var [=Value][,List]) \Rightarrow list

centralDiff(List1,Var [=Value][,Step]) \Rightarrow list

centralDiff(*Matrix1*,*Var* [=*Value*][,*Step*]) ⇒ *matrix*

Returns the numerical derivative using the central difference quotient formula.

When *Value* is specified, it overrides any prior variable assignment or any current "|" substitution for the variable.

Step is the step value. If Step is omitted, it defaults to 0.001.

When using *List1* or *Matrix1*, the operation gets mapped across the values in the list or across the matrix elements.

Note: See also avgRC() and d().

Catalogue > 🗐

$$\frac{\left(\cos(x),x,h\right)}{\frac{-\left(\cos(x-h)-\cos(x+h)\right)}{2\cdot h}}$$

$$\frac{\left(\cos(x-h)-\cos(x+h)\right)}{2\cdot h}$$

$$\lim_{h\to 0}\left(\operatorname{centralDiff}\left(\cos(x),x,h\right)\right) \qquad -\sin(x)$$

$$\operatorname{centralDiff}\left(x^3,x,0.01\right)$$

$$\frac{3\cdot \left(x^2+0.000033\right)}{\operatorname{centralDiff}\left(\cos(x),x\right)|x=\frac{\pi}{2}}$$

$$\operatorname{centralDiff}\left(x^2,x,\left\{0.01,0.1\right\}\right)$$

$$\left\{2\cdot x,2\cdot x\right\}$$

 $cFactor(Expr1[,Var]) \Rightarrow expression$ $cFactor(List1[,Var]) \Rightarrow list$ $cFactor(Matrix 1[, Var]) \Rightarrow matrix$

cFactor(*Expr1*) returns *Expr1* factored with respect to all of its variables over a common denominator.

Expr1 is factored as much as possible toward linear rational factors even if this introduces new non-real numbers. This alternative is appropriate if you want factorization with respect to more than one variable.

cFactor(Expr1,Var**)** returns Expr1 factored with respect to variable Var.

Expr1 is factored as much as possible toward factors that are linear in Var, with perhaps non-real constants, even if it introduces irrational constants or subexpressions that are irrational in other variables.

The factors and their terms are sorted with Var as the main variable. Similar powers of Var are collected in each factor. Include Var if factorization is needed with respect to only that variable and you are willing to accept irrational expressions in any other variables to increase factorization with respect to Var. There might be some incidental factoring with respect to other variables.

For the Auto setting of the Auto or **Approximate** mode, including *Var* also permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including Varmight yield more complete factorization.

Note: See also factor().

$$\begin{array}{c} \operatorname{cFactor}(a^3 \cdot x^2 + a \cdot x^2 + a^3 + a.x) \\ a \cdot \left(a^2 + 1\right) \cdot \left(x - t\right) \cdot \left(x + t\right) \\ \operatorname{cFactor}\left(x^2 + \frac{4}{9}\right) & \frac{\left(3 \cdot x - 2 \cdot t\right) \cdot \left(3 \cdot x + 2 \cdot t\right)}{9} \\ \operatorname{cFactor}(x^2 + 3) & x^2 + 3 \\ \operatorname{cFactor}(x^2 + a) & x^2 + a \end{array}$$

cFactor(
$$x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3$$
)
 $x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3$
cFactor($x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3, x$)
($x-0.964673$)·($x+0.611649$)·($x+2.12543$)·($x+3.12543$)·($x+3.$

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

char()

Catalogue > 🗓

$char(Integer) \Rightarrow character$

char(38) "&" char(65) "A"

Returns a character string containing the character numbered *Integer* from the handheld character set. The valid range for *Integer* is 0–65535.

charPoly()

Catalogue > 🕎

charPoly(squareMatrix, Var**)** \Rightarrow $polynomial\ expression$

charPoly(squareMatrix,Expr**)** \Rightarrow $polynomial\ expression$

charPoly(squareMatrix1,Matrix2**)** ⇒ polynomial expression

Returns the characteristic polynomial of squareMatrix. The characteristic polynomial of $n\times n$ matrix A, denoted by p_A (λ), is the polynomial defined by

$$p_A(\lambda) = \det(\lambda \cdot I - A)$$

where I denotes the $n \times n$ identity matrix.

squareMatrix1 and squareMatrix2 must have the equal dimensions.

		-080-
$m := \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$	\[\begin{pmatrix} 1 & 3 \\ 2 & -1 \\ -2 & 2 \end{pmatrix} \]	0]
m:= 2 -1 0	2 -1	0
[-2 2 5]	-2 2	5]
charPoly(m,x)	$-x^3 + 5 \cdot x^2 + 7 \cdot x$	-35
charPoly (m,x^2+1)	$-x^6+2\cdot x^4+14\cdot x^2$	-24
charPoly(m,m)		0

χ**22way**

Catalogue > 🗐

χ**22way** obsMatrix

chi22way obsMatrix

Computes a χ^2 test for association on the two-way table of counts in the observed matrix obsMatrix. A summary of results is stored in the stat.results variable. (page 174)

For information on the effect of empty elements in a matrix, see "Empty (Void) Elements," page 248.

Output variable	Description
stat.χ ²	Chi square stat: sum (observed - expected) ² /expected

Output variable	Description
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.df	Degrees of freedom for the chi square statistics
stat.ExpMat	Matrix of expected elemental count table, assuming null hypothesis
stat.CompMat	Matrix of elemental chi square statistic contributions

χ^2 Cdf() Catalogue > 🕄

 χ^2 Cdf(lowBound,upBound,df) \Rightarrow number if lowBound and upBound are numbers, list if lowBound and upBound are lists

 $chi2Cdf(lowBound,upBound,df) \Rightarrow number if$ lowBound and upBound are numbers, list if lowBound and upBound are lists

Computes the χ^2 distribution probability between lowBound and upBound for the specified degrees of freedom df.

For $P(X \le upBound)$, set lowBound = 0.

For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

χ2GOF Catalogue > 23

γ**2GOF** obsList,expList,df

chi2GOF obsList,expList,df

Performs a test to confirm that sample data is from a population that conforms to a specified distribution. obsList is a list of counts and must contain integers. A summary of results is stored in the stat.results variable. (See page 174.)

For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

Output variable	Description
stat.χ ²	Chi square stat: sum((observed - expected)²/expected
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected

Output variable	Description
stat.df	Degrees of freedom for the chi square statistics
stat.CompList	Elemental chi square statistic contributions

χ²Pdf() Catalogue > [2]

 $\chi^{2}Pdf(XVal,df) \Rightarrow number$ if XVal is a number, list if XVal is a list

chi2Pdf(XVal,df) \Rightarrow *number* if XVal is a number, *list* if XVal is a list

Computes the probability density function (pdf) for the χ^2 distribution at a specified XVal value for the specified degrees of freedom df.

For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

ClearAZ		Catalogue > 🗐
ClearAZ	$5 \rightarrow b$	5
Clears all single-character variables in the	b	5
current problem space.	ClearAZ	Done
If one or more of the variables are locked, this command displays an error message and deletes only the unlocked variables. See unlock, page 195.	b	b

ClrErr Catalogue > 🕡

CIrErr

Clears the error status and sets system variable *errCode* to zero.

The Else clause of the Try...Else...EndTry block should use ClrErr or PassErr. If the error is to be processed or ignored, use ClrErr. If what to do with the error is not known, use PassErr to send it to the next error handler. If there are no more pending Try...Else...EndTry error handlers, the error dialogue box will be displayed as normal.

Note: See also PassErr, page 129, and Try, page 188.

For an example of **CIrErr**, See Example 2 under the **Try** command, page 188.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

colAugment()

Catalogue > 🗐

 $colAugment(Matrix1, Matrix2) \Rightarrow matrix$

Returns a new matrix that is *Matrix2* appended to *Matrix1*. The matrices must have equal column dimensions, and *Matrix2* is appended to *Matrix1* as new rows. Does not alter *Matrix1* or *Matrix2*.

$\begin{bmatrix} 1 & 2 \end{bmatrix} \rightarrow m1$	1 2
[3 4]	[3 4]
$\begin{bmatrix} 5 & 6 \end{bmatrix} \rightarrow m2$	[5 6]
colAugment(m1, m2)	1 2
	3 4
	[5 6]

colDim[0 1 2]

colDim()

Catalogue > [1]

 $colDim(Matrix) \Rightarrow expression$

Returns the number of columns contained in *Matrix*.

Note: See also rowDim().

eataiogae 🔻

colNorm()

 $colNorm(Matrix) \Rightarrow expression$

Returns the maximum of the sums of the absolute values of the elements in the columns in *Matrix*.

Note: Undefined matrix elements are not allowed. See also **rowNorm()**.

	Catalogue > ৠু
2]	[1 -2 2]

 $\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix} \rightarrow mat \qquad \begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix}$ $colNorm(mat) \qquad \qquad 9$

comDenom()

Catalogue > 🗐

 $comDenom(Expr1[,Var]) \Rightarrow expression \\ comDenom(List1[,Var]) \Rightarrow list \\ comDenom(Matrix1[,Var]) \Rightarrow matrix$

comDenom(Expr1) returns a reduced ratio of a fully expanded numerator over a fully expanded denominator.

comDenom
$$\left(\frac{y^2+y}{(x+1)^2} + y^2 + y\right)$$

 $\frac{x^2 \cdot y^2 + x^2 \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot x \cdot y + 2 \cdot y^2 + 2 \cdot y}{x^2 + 2 \cdot x + 1}$

comDenom()

Catalogue > 2

comDenom(Expr1, Var) returns a reduced ratio of numerator and denominator expanded with respect to Var. The terms and their factors are sorted with *Var* as the main variable. Similar powers of *Var* are collected. There might be some incidental factoring of the collected coefficients. Compared to omitting *Var*, this often saves time, memory, and screen space, while making the expression more comprehensible. It also makes subsequent operations on the result faster and less likely to exhaust memory.

If Var does not occur in Expr1, comDenom (Expr1.Var) returns a reduced ratio of an unexpanded numerator over an unexpanded denominator. Such results usually save even more time, memory, and screen space. Such partially factored results also make subsequent operations on the result much faster and much less likely to exhaust memory.

Even when there is no denominator, the comden function is often a fast way to achieve partial factorization if factor() is too slow or if it exhausts memory.

Hint: Enter this comden() function definition and routinely try it as an alternative to comDenom() and factor().

$$\frac{x^{2} \cdot y \cdot (y+1) + y^{2} + y, x}{\frac{x^{2} \cdot y \cdot (y+1) + 2 \cdot x \cdot y \cdot (y+1) + 2 \cdot y \cdot (y+1)}{x^{2} + 2 \cdot x + 1}}$$

$$\frac{y^{2} \cdot y \cdot (y+1) + 2 \cdot x \cdot y \cdot (y+1) + 2 \cdot y \cdot (y+1)}{x^{2} + 2 \cdot x + 1}$$

$$\frac{y^{2} \cdot (x^{2} + 2 \cdot x + 2) + y \cdot (x^{2} + 2 \cdot x + 2)}{x^{2} + 2 \cdot x + 1}$$

Define *comden(exprn)*=comDenom(*exprn,abc*) $\left\{\frac{y^2+y}{(x+1)^2}+y^2+y\right\} = \frac{(x^2+2\cdot x+2)\cdot y\cdot (y+1)}{(x+1)^2}$

$$\frac{comden (1234 \cdot x^2 \cdot (y^3 - y) + 2468 \cdot x \cdot (y^2 - 1))}{1234 \cdot x \cdot (x \cdot y + 2) \cdot (y^2 - 1)}$$

completeSquare ()

Catalogue > 🕮

 $completeSquare(ExprOrEqn, Var) \Rightarrow$ expression or equation

completeSquare(ExprOrEqn, Var^Power) ⇒ expression or equation

completeSquare(ExprOrEqn, Var1, Var2 [,...]) \Rightarrow expression or equation

completeSquare(ExprOrEqn, {Var1, Var2 [,...] \Rightarrow expression or equation

Converts a quadratic polynomial expression of the form a•x2+b•x+c into the form a•(xh)2+k

completeSquare(
$$x^2+2\cdot x+3x$$
) $(x+1)^2+2$
completeSquare($x^2+2\cdot x=3x$) $(x+1)^2=4$
completeSquare($x^6+2\cdot x^3+3x^3$) $(x^3+1)^2+2$
completeSquare($x^2+4\cdot x+y^2+6\cdot y+3=0,x,y$) $(x+2)^2+(y+3)^2=10$

completeSquare ()

Catalogue > 23

- or -

Converts a quadratic equation of the form $a \cdot x^2 + b \cdot x + c = d$ into the form $a \cdot (x - h)^2 = k$

The first argument must be a quadratic expression or equation in standard form with respect to the second argument.

The Second argument must be a single univariate term or a single univariate term raised to a rational power, for example x, y^2 , or $z^{(1/3)}$.

The third and fourth syntax attempt to complete the square with respect to variables Var1, Var2 [,...]).

completeSquare $\left\{3 \cdot x^2 + 2 \cdot y + 7 \cdot y^2 + 4 \cdot x = 3, \left\{ x, y \right\} \right\}$ $3 \cdot \left\{ x + \frac{2}{3} \right\}^2 + 7 \cdot \left[y + \frac{1}{7} \right]^2 = \frac{94}{21}$

completeSquare $(x^2+2\cdot x\cdot y,x,y)$ $(x+y)^2-y$

conj() Catalogue > 🗊

 $conj(Expr1) \Rightarrow expression$

 $conj(List1) \Rightarrow list$ $conj(Matrix1) \Rightarrow matrix$

Returns the complex conjugate of the argument.

Note: All undefined variables are treated as real variables.

$conj(1+2\cdot i)$	1-2·i
$ \begin{array}{c c} \hline \operatorname{conj} \begin{bmatrix} 2 & 1 - 3 \cdot i \\ -i & -7 \end{bmatrix} \end{array} $	$\begin{bmatrix} 2 & 1+3 \cdot i \\ i & -7 \end{bmatrix}$
conj(z)	Z
$conj(x+i\cdot y)$	<i>x</i> - <i>y</i> · <i>i</i>

constructMat()

constructMat

(Expr,Var1,Var2,numRows,numCols) ⇒ matrix

Returns a matrix based on the arguments.

Expr is an expression in variables Var1 and Var2. Elements in the resulting matrix are formed by evaluating Expr for each incremented value of Var1 and Var2.

Var1 is automatically incremented from 1 through numRows. Within each row, Var2 is incremented from 1 through numCols.

Catalogue > 23

constructMat
$$\left(\frac{1}{i+j}, i, j, 3, 4\right)$$
 $\begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$

CopyVar

Catalogue > 👰

CopyVar Var1, Var2

CopyVar Var1., Var2.

CopyVar Var1, Var2 copies the value of variable Var1 to variable Var2, creating Var2 if necessary. Variable Var1 must have a value.

If Var1 is the name of an existing userdefined function, copies the definition of that function to function Var2. Function Var1 must be defined.

Var1 must meet the variable-naming requirements or must be an indirection expression that simplifies to a variable name meeting the requirements.

CopyVar *Var1.*, *Var2*. copies all members of the *Var1*. variable group to the *Var2*. group, creating *Var2*. if necessary.

Var1. must be the name of an existing variable group, such as the statistics stat.nn results, or variables created using the LibShortcut() function. If Var2. already exists, this command replaces all members that are common to both groups and adds the members that do not already exist. If one or more members of Var2. are locked, all members of Var2. are left unchanged.

Define $a(x) = \frac{1}{x}$	Done
Define $b(x)=x^2$	Done
CopyVar $a,c: c(4)$	1
	4
CopyVar b,c: c(4)	16

aa.a:=45				4 5
<i>aa.b</i> :=6.78			6.	78
CopyVar aa.,bb.			Do	
getVarInfo()	aa.a	"NUM" "NUM" "NUM" "NUM"	"[]"	0
	aa.b	"NUM"	"[]"	0
	bb.a	"NUM"	"[]"	0
	bb.b	"NUM"	"[]"	0

corrMat()

Catalogue > 🕎

 $\textbf{corrMat}(List1,\!List2[,\!...[,\!List20]])$

Computes the correlation matrix for the augmented matrix [List1, List2, ..., List20].

▶ cos

Catalogue > 📳

Expr ▶cos

Note: You can insert this operator from the computer keyboard by typing @>cos.

Represents *Expr* in terms of cosine. This is a display conversion operator. It can be used only at the end of the entry line.

$$(\sin(x))^2 \blacktriangleright \cos \qquad 1 - (\cos(x))^2$$

► cos reduces all powers of sin(...) modulo 1—cos(...)^2 so that any remaining powers of cos(...) have exponents in the range (0, 2). Thus

have exponents in the range (0, 2). Thus, the result will be free of sin(...) if and only if sin(...) occurs in the given expression only to even powers.

Note: This conversion operator is not supported in Degree or Gradian Angle modes. Before using it, make sure that the Angle mode is set to Radians and that *Expr* does not contain explicit references to degree or gradian angles.

cos()	ig	k	(6	е	;)	Į	,
-------	----	---	----	---	---	---	---	---

 $cos(Expr1) \Rightarrow expression$

 $cos(List1) \Rightarrow list$

cos(*Expr1*) returns the cosine of the argument as an expression.

cos(List1) returns a list of the cosines of all elements in List1.

Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode setting. You can use $^{\circ}$, G , or r to override the angle mode temporarily.

In Degree angle mode:

$\frac{1}{\cos\left(\frac{\pi}{r}\right)}$	$\sqrt{2}$
\4 /	2
cos(45)	$\sqrt{2}$
	2
cos({0,60,90})	$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$

In Gradian angle mode:

$$\cos(\{0,50,100\})$$
 $\left\{1,\frac{\sqrt{2}}{2},0\right\}$

In Radian angle mode:

$\cos\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
cos(45°)	$\sqrt{2}$
	2

 $\cos(squareMatrix l) \Rightarrow squareMatrix$

Returns the matrix cosine of squareMatrix1. This is not the same as calculating the cosine of each element. In Radian angle mode:

cos()



When a scalar function f(A) operates on squareMatrix I (A), the result is calculated by the algorithm:

Compute the eigenvalues (λ_i) and eigenvectors (V_i) of A.

squareMatrix1 must be diagonalizable. Also, it cannot have symbolic variables that have not been assigned a value.

Form the matrices:

$$B = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \text{ and } X = [V_1, V_2, \dots, V_n]$$

Then A = X B X-1 and f(A) = X f(B) X-1. For example, cos(A) = X cos(B) X-1 where:

$$cos(B) =$$

$$\begin{bmatrix} \cos(\lambda_1) & 0 & \dots & 0 \\ 0 & \cos(\lambda_2) & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \cos(\lambda_n) \end{bmatrix}$$

All computations are performed using floating-point arithmetic.

$$\begin{bmatrix} 1 & 3 & 1 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$$

cos-1()

 $\cos -1(Expr1) \Rightarrow expression$

 $\cos -1(List1) \Rightarrow list$

 \cos -1(ExprI) returns the angle whose cosine is ExprI as an expression.

 $\cos -1(List I)$ returns a list of the inverse cosines of each element of List I.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arccos (...).

In Degree angle mode:

 $\cos^{-1}(1)$ 0

In Gradian angle mode:

cos⁻¹(0) 100

In Radian angle mode:

 $\cos^{-1}(\{0,0.2,0.5\})$ $\left\{\frac{\pi}{2},1.36944,1.0472\right\}$

trig key

cos-1()



 $cos-1(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix inverse cosine of *squareMatrix1*. This is not the same as calculating the inverse cosine of each element. For information about the calculation method, refer to **cos()**.

square Matrix 1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and Rectangular Complex Format:

1.73485+0.064606•*i* -1.49086+2.10514 -0.725533+1.51594•*i* 0.623491+0.77836• -2.08316+2.63205•*i* 1.79018-1.27182•

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

cosh()

In Degree angle mode:

 $cosh(Expr1) \Rightarrow expression$

 $cosh(List1) \Rightarrow list$

cosh(*Expr1*) returns the hyperbolic cosine of the argument as an expression.

 $\cosh(List 1)$ returns a list of the hyperbolic cosines of each element of List 1.

 $cosh(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix hyperbolic cosine of *squareMatrix1*. This is not the same as calculating the hyperbolic cosine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

TI Degi ee angle mode.

 $\cosh\left(\left(\frac{\pi}{4}\right)^r\right)$ $\cosh(45)$

In Radian angle mode:

cosh-1()

Catalogue > 🕎

Catalogue > 🗐

 $cosh-1(Expr1) \Rightarrow expression$

 $cosh-1(List1) \Rightarrow list$

cosh-1(*Expr1*) returns the inverse hyperbolic cosine of the argument as an expression.

 $\cosh^{3}(1)$ 0 $\cosh^{3}(\{1,2.1,3\})$ $\{0,1.37286,\cosh^{3}(3)\}$

cosh-1(*List1*) returns a list of the inverse hyperbolic cosines of each element of *List1*.

Note: You can insert this function from the keyboard by typing arccosh (...).

 $cosh-1(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix inverse hyperbolic cosine of *squareMatrix1*. This is not the same as calculating the inverse hyperbolic cosine of each element. For information about the calculation method, refer to **cos** ().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and In Rectangular Complex Format:

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

cot()

 $cot(Expr1) \Rightarrow expression$

 $cot(List1) \Rightarrow list$

Returns the cotangent of Expr1 or returns a list of the cotangents of all elements in List1.

Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode setting. You can use $^{\circ}$, G , or r to override the angle mode temporarily.

In Degree angle mode:

cot(45)

In Gradian angle mode:

cot(50)

In Radian angle mode:

$$cot(\{1,2.1,3\})$$
 $\left\{\frac{1}{\tan(1)}, 0.584848, \frac{1}{\tan(3)}\right\}$

cot-1()

 $\cot^{-1}(Exprl) \Rightarrow expression$

 $\cot -1(List 1) \Rightarrow list$

Returns the angle whose cotangent is Expr1 or returns a list containing the inverse cotangents of each element of List1. In Degree angle mode:

cot-1(1)

45.

trig key

trig key

1

1

In Gradian angle mode:

cot⁻¹(1) 50.

cot-1()

trig key

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arccot(...).

In Radian angle mode:

cot ⁻¹ (1)	<u>π</u>
	$\overline{4}$

coth()

 $coth(Expr1) \Rightarrow expression$

 $coth(List1) \Rightarrow list$

Returns the hyperbolic cotangent of Expr1 or returns a list of the hyperbolic cotangents of all elements of List1.

Catalogue > 23

coth-1()

 $coth-1(Expr1) \Rightarrow expression$

 $coth-1(List1) \Rightarrow list$

Returns the inverse hyperbolic cotangent of Expr1 or returns a list containing the inverse hyperbolic cotangents of each element of List1.

Note: You can insert this function from the keyboard by typing arcoth (...).

Catalogue > 🗐

 $\frac{\coth^{3}(3.5) \qquad 0.293893}{\coth^{3}(\left\{-2,2.1,6\right\})} \\
\left\{\frac{-\ln(3)}{2}, 0.518046, \frac{\ln\left(\frac{7}{5}\right)}{2}\right\}$

count()

count(Value lor List 1[,Value 2 or List 2[,...]]) $\Rightarrow value$

Returns the accumulated count of all elements in the arguments that evaluate to numeric values.

Each argument can be an expression, value, list, or matrix. You can mix data types and use arguments of various dimensions.

For a list, matrix, or range of cells, each element is evaluated to determine if it should be included in the count.

Catalogue > 🗐

$$\frac{\text{count}(2,4,6)}{\text{count}(2,4,6)} \qquad \qquad 3 \\
\frac{\text{count}(2,4,6), \begin{bmatrix} 8 & 10 \\ 12 & 14 \end{bmatrix})}{\text{count}(\frac{1}{2},3+4\cdot i,\text{undef},\text{"hello"},x+5.,\text{sign}(0))} \\
2$$

In the last example, only 1/2 and 3+4*i are counted. The remaining arguments, assuming x is undefined, do not evaluate to numeric values.

Within the Lists & Spreadsheet application, you can use a range of cells in place of any argument.

Empty (void) elements are ignored. For more information on empty elements, see page 248.

countif()

Catalogue > 🗐

 $countif(List,Criteria) \Rightarrow value$

Returns the accumulated count of all elements in *List* that meet the specified *Criteria*.

Criteria can be:

- A value, expression, or string. For example, 3 counts only those elements in List that simplify to the value 3.
- A Boolean expression containing the symbol ? as a place holder for each element. For example, ?<5 counts only those elements in List that are less than

Within the Lists & Spreadsheet application, you can use a range of cells in place of *List*.

Empty (void) elements in the list are ignored. For more information on empty elements, see page 248.

Note: See also sumif(), page 178, and frequency(), page 74.

$$countIf(\{1,3,"abc",undef,3,1\},3)$$

Counts the number of elements equal to 3.

Counts the number of elements equal to "def."

countIf
$$(x^{-2}, x^{-1}, 1, x, x^{2}, x)$$

Counts the number of elements equal to x; this example assumes the variable x is undefined

countIf(
$$\{1,3,5,7,9\}$$
,?<5) 2

Counts 1 and 3.

Counts 3, 5, and 7.

Counts 1, 3, 7, and 9.

cPolyRoots()

Catalogue > 🗐

 $cPolyRoots(Poly,Var) \Rightarrow list$

 $(roighoods(roig, var) \rightarrow iisi$

 $cPolyRoots(ListOfCoeffs) \Rightarrow list$

The first syntax, cPolyRoots(Poly,Var), returns a list of complex roots of polynomial Poly with respect to variable Var.

Poly must be a polynomial in one variable.

The second syntax, **cPolyRoots** (*ListOfCoeffs*), returns a list of complex roots for the coefficients in *ListOfCoeffs*.

Note: See also polyRoots(), page 134.

polyRoots(v³+1,v)	{-1}
$cPolyRoots(y^3+1,y)$	
$\left\{-1,\frac{1}{2},\frac{1}{2}\right\}$	$\frac{\sqrt{3}}{2} \cdot i, \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i$
$polyRoots(x^2+2•x+1,x)$	{-1,-1}
cPolyRoots({1,2,1})	{-1,-1}

crossP()

Catalogue > 👰

 $crossP(List1, List2) \Rightarrow list$

Returns the cross product of List1 and List2 as a list.

List1 and List2 must have equal dimension, and the dimension must be either 2 or 3.

 $crossP(Vector1, Vector2) \Rightarrow vector$

Returns a row or column vector (depending on the arguments) that is the cross product of *Vector I* and *Vector 2*.

Both *Vector1* and *Vector2* must be row vectors, or both must be column vectors. Both vectors must have equal dimension, and the dimension must be either 2 or 3.

$\overline{\operatorname{crossP}(\{a1,b1\},\{a2,b2\})}$
$\{0,0,a1 \cdot b2 - a2 \cdot b1\}$
crossP({0.1,2.2,-5},{1,-0.5,0})
{-2.5,-5.,-2.25}

csc()

trig key

 $csc(Expr1) \Rightarrow expression$

 $csc(List1) \Rightarrow list$

Returns the cosecant of Expr1 or returns a list containing the cosecants of all elements in List1.

In Degree angle mode:

csc(45)

 $\sqrt{2}$

In Gradian angle mode:

csc(50)

 $\sqrt{2}$

In Radian angle mode:

$$\csc\left\{\left\{1,\frac{\pi}{2},\frac{\pi}{3}\right\}\right\} \qquad \left\{\frac{1}{\sin(1)},1,\frac{2\cdot\sqrt{3}}{3}\right\}$$

csc-1() trig key

 $csc-1(Expr1) \Rightarrow expression$

 $csc-1(List1) \Rightarrow list \qquad csc'(1)$

Returns the angle whose cosecant is *Expr1* or returns a list containing the inverse cosecants of each element of *List1*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arcsc(...).

In Degree angle mode:

csc⁻¹(1) 90.

In Gradian angle mode:

csc⁻¹(1) 100.

In Radian angle mode:

$$\frac{\operatorname{csc}^{-1}(\{1,4,6\})}{\left\{\frac{\pi}{2},\sin^{-1}\left(\frac{1}{4}\right),\sin^{-1}\left(\frac{1}{6}\right)\right\}}$$

csch()

 $csch(Expr1) \Rightarrow expression$

 $csch(List1) \Rightarrow list$

all elements of List 1.

Returns the hyperbolic cosecant of *Expr1* or returns a list of the hyperbolic cosecants of

 $\frac{1}{\sinh(3)}$ $\frac{1}{\sinh(3)}$ $\frac{1}{\cosh(\{1,2.1,4\})}$ $\frac{1}{1-0.248641}$

Catalogue > 🗐

csch-1() Catalogue > 💱

 $\operatorname{csch-1}(Expr1) \Rightarrow expression$

 $csch-1(List1) \Rightarrow list$

Returns the inverse hyperbolic cosecant of Expr1 or returns a list containing the inverse hyperbolic cosecants of each element of List1.

Note: You can insert this function from the keyboard by typing arcsch (...).

 $\frac{ \cosh^{3}(1) \qquad \qquad \sinh^{3}(1)}{ \cosh^{3}(\{1,2.1,3\}) } \\ \left\{ \sinh^{3}(1),0.459815,\sinh^{3}\left(\frac{1}{2}\right) \right\}$

cSolve()

Catalogue > 23

 $cSolve(Equation, Var) \Rightarrow Boolean$ expression

 $cSolve(Equation, Var=Guess) \Rightarrow Boolean$ expression

 $cSolve(Inequality, Var) \Rightarrow Boolean$ expression

cSolve
$$(x^3 = 1, x)$$

 $x = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ or } x = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } x = -1$
solve $(x^3 = -1, x)$ $x = -1$

Returns candidate complex solutions of an equation or inequality for Var. The goal is to produce candidates for all real and nonreal solutions. Even if *Equation* is real, cSolve() allows non-real results in Real result Complex Format.

cSolve() temporarily sets the domain to complex during the solution even if the current domain is real. In the complex domain, fractional powers having odd denominators use the principal rather than the real branch. Consequently, solutions from **solve()** to equations involving such fractional powers are not necessarily a subset of those from cSolve().

cSolve() starts with exact symbolic methods. cSolve() also uses iterative approximate complex polynomial factoring, if necessary.

Note: See also cZeros(), solve(), and zeros().

cSolve
$$\left(x^{\frac{1}{3}} = -1, x\right)$$
 false colve $\left(x^{\frac{1}{3}} = -1, x\right)$ $x = -1$

In Display Digits mode of Fix 2:

exact(cSolve(
$$x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3=0,x$$
))
 $x\cdot (x^4+4\cdot x^3+5\cdot x^2-6)=3$
cSolve(Ans,x)
 $x=1.11+1.07\cdot i$ or $x=1.11-1.07\cdot i$ or $x=2.19$

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

cSolve(Eqn1 and Eqn2 [and...],VarOrGuess1, VarOrGuess2 [, ...]) \Rightarrow Boolean expression

cSolve(SystemOfEgns, VarOrGuess1, $VarOrGuess2[,...]) \Rightarrow$ Boolean expression

Returns candidate complex solutions to the simultaneous algebraic equations, where each *varOrGuess* specifies a variable that you want to solve for.

Optionally, you can specify an initial guess for a variable. Each *varOrGuess* must have the form:

variable – or –

variable = real or non-real number

For example, x is valid and so is x=3+i.

If all of the equations are polynomials and if you do NOT specify any initial guesses, **cSolve()** uses the lexical Gröbner/Buchberger elimination method to attempt to determine **all** complex solutions.

Complex solutions can include both real and non-real solutions, as in the example to the right.

Simultaneous polynomial equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

You can also include solution variables that do not appear in the equations. These solutions show how families of solutions might contain arbitrary constants of the form $\mathbf{c}k$, where k is an integer suffix from 1 through 255.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or <code>varOrGuess</code> list.

cSolve
$$\left(u \cdot v - u = v \text{ and } v^2 = -u, \left\{u, v\right\}\right)$$

 $u = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } u = \frac{1}{2} - \frac{\sqrt{3}}{2}$

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

cSolve
$$\left(u \cdot v - u = c \cdot v \text{ and } v^2 = -u, \{u, v\}\right)$$

$$u = \frac{-\left(\sqrt{4 \cdot c - 1} \cdot i + 1\right)^2}{4} \text{ and } v = \frac{\sqrt{4 \cdot c - 1} \cdot i + 1}{2} \text{ o}$$

cSolve
$$\left(u \cdot v - u = v \text{ and } v^2 = -u, \left\{u, v, w\right\}\right)$$

 $u = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ and } w = \textbf{c43} \text{ or}^{\textbf{r}}$

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in all solution variables, cSolve() uses Gaussian elimination to attempt to determine all solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables, cSolve() determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

A non-real guess is often necessary to determine a non-real solution. For convergence, a guess might have to be rather close to a solution.

$$\operatorname{cSolve}\!\left(\!u\!+\!\nu\!=\!e^{\mathcal{W}} \text{ and } u\!-\!\nu\!=\!i,\!\left\{u,v\right\}\!\right) \\ u\!=\!\frac{e^{\mathcal{W}}\!+\!i}{2} \text{ and } v\!=\!\frac{e^{\mathcal{W}}\!-\!i}{2}$$

cSolve
$$\left(e^{z} = w \text{ and } w = z^{2}, \{w, z\}\right)$$

 $w = 0.494866 \text{ and } z = 0.703467$

cSolve
$$\left(e^Z = w \text{ and } w = z^2, \{w, z = 1 + i\}\right)$$

 $w = 0.149606 + 4.8919 \cdot i \text{ and } z = 1.58805 + 1.5402$

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

CubicReg

Catalogue > 🗐

CubicReg X, Y[, [Freq] [, Category, Include]]

Computes the cubic polynomial regression $y=a*x^3+b*x^2+c*x+d$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable. (See page 174.)

All the lists must have equal dimension except for *Include*.

X and Y are lists of independent and dependent

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.



For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

Output variable	Description
stat.RegEqn	Regression equation: a•x³+b•x²+c•x+d
stat.a, stat.b, stat.c, stat.d	Regression coefficients
stat.R ²	Coefficient of determination
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified <i>X List</i> actually used in the regression based on restrictions of <i>Freq</i> , <i>Category List</i> , and <i>Include Categories</i>
stat.YReg	List of data points in the modified Y List actually used in the regression based on restrictions of Freq, Category List, and Include Categories
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

cumulativeSum()

Catalogue > 🗊

 $cumulativeSum(List1) \Rightarrow list$

 $cumulativeSum(\{1,2,3,4\}) \qquad \qquad \{1,3,6,10\}$

Returns a list of the cumulative sums of the elements in List I, starting at element 1.

 $cumulativeSum(Matrix1) \Rightarrow matrix$

Returns a matrix of the cumulative sums of the elements in *Matrix I*. Each element is the cumulative sum of the column from top to bottom.

An empty (void) element in *List1* or *Matrix1* produces a void element in the resulting list or matrix. For more information on empty elements, see page 248.

1	2		1	2
3	4	→ m1	3	4
5	6		5	6
cur	nul	ativeSum $(m1)$	1	2
			4	6
			9	12]

Cycle

Catalogue > 😰

Cycle

Transfers control immediately to the next iteration of the current loop (**For**, **While**, or **Loop**).

Function listing that sums the integers from 1 to 100 skipping 50.

Cycle

Catalogue > 🗐

Cycle is not allowed outside the three looping structures (**For**, **While**, or **Loop**).

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define g()=Func	Done
Local temp,i	
$0 \rightarrow temp$	
For <i>i</i> ,1,100,1	
If $i=50$	
Cycle	
$temp+i \rightarrow temp$	
EndFor	
Return temp	
EndFunc	
g()	5000

► Cylind

Catalogue > 🗐

Vector ▶ Cylind

Note: You can insert this operator from the computer keyboard by typing @>Cylind.

Displays the row or column vector in cylindrical form $[r, \angle \theta, z]$.

Vector must have exactly three elements. It can be either a row or a column.

[2 2 3] Cylind
$$2 \cdot \sqrt{2} \angle \frac{\pi}{4}$$

cZeros()

Catalogue > 💱

 $cZeros(Expr, Var) \Rightarrow list$

Returns a list of candidate real and non-real values of Var that make Expr=0. cZeros() does this by computing

exp list(cSolve(Expr=0, Var), Var). Otherwise, cZeros() is similar to zeros().

Note: See also cSolve(), solve(), and zeros().

cZeros({Expr1, Expr2 [, ...] }, {
$$VarOrGuess1, VarOrGuess2$$
 [, ...] }) \Rightarrow matrix

Returns candidate positions where the expressions are zero simultaneously. Each VarOrGuess specifies an unknown whose value you seek.

In Display Digits mode of Fix 3:

cZeros
$$(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$$

{-1.114+1.073·**i**,-1.114-1.073·**i**,-2.125,-0.612,0.

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

Optionally, you can specify an initial guess for a variable. Each *VarOrGuess* must have the form:

variable

variable = real or non-real number

For example, x is valid and so is x=3+i.

If all of the expressions are polynomials and you do NOT specify any initial guesses, cZeros() uses the lexical

Gröbner/Buchberger elimination method to attempt to determine **all** complex zeros.

Complex zeros can include both real and non-real zeros, as in the example to the right.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the *VarOrGuess* list. To extract a row, index the matrix by [row].

Simultaneous polynomials can have extra variables that have no values, but represent given numeric values that could be substituted later.

You can also include unknown variables that do not appear in the expressions. These zeros show how families of zeros might contain arbitrary constants of the form $\mathbf{c}k$, where k is an integer suffix from 1 through 255.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or *VarOrGuess* list.

czeros(
$$\{u \cdot v - u - v, v^2 + u\}, \{u, v\}$$
)
$$\begin{bmatrix}
0 & 0 & 0 \\
\frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \\
\frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i
\end{bmatrix}$$

Extract row 2:

cZeros(
$$\{u \cdot v - u - c \cdot v^2, v^2 + u\}, \{u, v\}$$
)
$$\begin{bmatrix} 0 & 0 \\ -(c-1)^2 & -(c-1) \end{bmatrix}$$

cZeros(
$$\{u \cdot v - u - v, v^2 + u\}$$
, $\{u, v, w\}$)
cZero($\{u \cdot (v - 1) - v, u + v^2\}$, $\{u, v, w\}$)

$$\begin{bmatrix} \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & cd \\ \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & cd \end{bmatrix}$$

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in all unknowns. cZeros() uses Gaussian elimination to attempt to determine all zeros.

If a system is neither polynomial in all of its variables nor linear in its unknowns, cZeros () determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

A non-real guess is often necessary to determine a non-real zero. For convergence, a guess might have to be rather close to a zero.

cZeros(
$$\{u+v-e^{W},u-v-i\}$$
, $\{u,v\}$)
$$\left[\frac{e^{W}+i}{2} \quad \frac{e^{W}-i}{2}\right]$$

cZerog
$$({e^z - w, w - z^2}, {w,z})$$

[0.494866 -0.703467]

cZeros(
$$\{e^{-z}-w,w-z^2\}$$
, $\{w,z=1+i\}$)
[0.149606+4.8919· i 1.58805+1.54022· i]

ח

method.

dbd()		Catalogue > 🕎
$dbd(date1, date2) \Rightarrow value$	dbd(12.3103,1.0104)	1
Returns the number of days between $date 1$	dbd(1.0107,6.0107)	151
and $date2$ using the actual-day-count	dbd(3112.03.101.04)	1

date 1 and date 2 can be numbers or lists of numbers within the range of the dates on the standard calendar. If both date 1 and date2 are lists, they must be the same length.

date 1 and date 2 must be between the years 1950 through 2049.

You can enter the dates in either of two formats. The decimal placement differentiates between the date formats.

MM.DDYY (format used commonly in the United States)

DDMM.YY (format use commonly in Europe)

dbd(12.3103,1.0104)	1
dbd(1.0107,6.0107)	151
dbd(3112.03,101.04)	1
dbd(101.07,106.07)	151

▶DD

Catalogue > 🗐

Expr1 ▶DD⇒value

List1 ▶DD⇒*list*

$Matrix 1 \triangleright DD \Rightarrow matrix$

Note: You can insert this operator from the computer keyboard by typing @>DD.

Returns the decimal equivalent of the argument expressed in degrees. The argument is a number, list, or matrix that is interpreted by the Angle mode setting in gradians, radians or degrees.

In Degree angle mode:

(1.5°)▶DD	1.5°
(45°22'14.3")▶DD	45.3706°
({45°22'14.3",60°0'0"})▶DI)
	{45.3706°,60°}

In Gradian angle mode:

1▶DD	9
	10

In Radian angle mode:

(1.5)▶DD	85.9437°

▶Decimal

Expression1 ▶Decimal⇒expression

List1 ▶Decimal⇒*expression*

$Matrix1 \triangleright Decimal \Rightarrow expression$

Note: You can insert this operator from the computer keyboard by typing @>Decimal.

Displays the argument in decimal form. This operator can be used only at the end of the entry line.

Catalogue > 🗐

$\frac{1}{3}$ Decimal	0.333333

Define

Define Var = Expression

Define Function(Param1, Param2, ...) = Expression

Defines the variable Var or the user-defined function Function.

Catalogue > 🗐

Define $g(x,y)=2\cdot x-3\cdot y$	Done
g(1,2)	-4
$1 \rightarrow a: 2 \rightarrow b: g(a,b)$	-4
Define $h(x)$ =when($x < 2, 2 \cdot x - 3, -2 \cdot x + 3$)	Done
h(-3)	-9
h(4)	-5

Define

Parameters, such as *Param1*, provide place holders for passing arguments to the function. When calling a user-defined function, you must supply arguments (for example, values or variables) that correspond to the parameters. When called, the function evaluates *Expression* using the supplied arguments.

Var and Function cannot be the name of a system variable or built-in function or command.

Note: This form of **Define** is equivalent to executing the expression: expression → Function(Param1,Param2).

Define Function(Param1, Param2, ...) = Func

Block

EndFunc

Define Program(Param1, Param2, ...) =Prgm

Block

EndPrgm

In this form, the user-defined function or programme can execute a block of multiple statements.

Block can be either a single statement or a series of statements on separate lines. *Block* also can include expressions and instructions (such as If, Then, Else and For).

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Note: See also Define LibPriv, page 46, and Define LibPub, page 47.

Define $g(x,y)$ =Func	Done
If $x>y$ Then	
Return x	
Else	
Return y	
EndIf	
EndFunc	
g(3,-7)	3

```
If x>y Then
               Disp x," greater than ",y
               Disp x," not greater than ",y
               EndIf
               EndPrgm
                                       Done
g(3,-7)
                           3 greater than -7
```

Define g(x,y)=Prgm

Define LibPriv

Catalogue > 🗐

Done

Define LibPriv Var = Expression

Define LibPriv Function(Param1, Param2, ...) =

Catalogue > 23

Expression

Define LibPriv Function(Param1, Param2, ...) = Func Block

EndFunc

Define LibPriv Program(Param1, Param2, ...) = Prgm Block

EndPrgm

Operates the same as **Define**, except defines a private library variable, function, or programme. Private functions and programs do not appear in the Catalogue.

Note: See also Define, page 45, and Define LibPub, page 47.

Define LibPub

Catalogue > 🗐

Define LibPub Var = Expression

Define LibPub Function(Param1, Param2, ...) = Expression

Define LibPub Function(Param1, Param2, ...) = FuncBlock

EndFunc

Define LibPub Program(Param1, Param2, ...) = Prgm Block

EndPrgm

Operates the same as **Define**, except defines a public library variable, function, or programme. Public functions and programs appear in the Catalogue after the library has been saved and refreshed.

Note: See also Define, page 45, and Define LibPriv, page 46.

deltaList()

See Δ List(), page 102.

DelVar		Catalogue > 🗐
DelVar Var1[, Var2] [, Var3]	$2 \rightarrow a$	2
DelVar Var.	$(a+2)^2$	16
Dolotos the specified variable or variable	DelVar a	Done
Deletes the specified variable or variable group from memory.	$(a+2)^2$	$(a+2)^2$

If one or more of the variables are locked. this command displays an error message and deletes only the unlocked variables. See unLock, page 195.

DelVar Var. deletes all members of the Var. variable group (such as the statistics stat.nn results or variables created using the LibShortcut() function). The dot (.) in this form of the **DelVar** command limits it to deleting a variable group; the simple variable *Var* is not affected.

aa.a:=45			45
aa.b:=5.67			5.67
aa.c:=78.9			78.9
getVarInfo()	aa.a	"NUM"	"[]"]
	aa.b	"NUM"	"[]"
	aa.c	"NUM"	"[]"]
DelVar aa.			Done
getVarInfo()		"NONE'	

delVoid() Catalogue > 🗐 $delVoid(List1) \Rightarrow list$ delVoid({1,void,3})

Returns a list that has the contents of *List1* with all empty (void) elements removed.

For more information on empty elements, see page 248.

derivative() See d(), page 218. **deSolve(**lstOr2ndOrderODE, Var, depVar) $\Rightarrow a$ general solution

Returns an equation that explicitly or implicitly specifies a general solution to the 1st- or 2nd-order ordinary differential equation (ODE). In the ODE:

- Use a prime symbol (press ?!-) to denote the 1st derivative of the dependent variable with respect to the independent variable.
- Use two prime symbols to denote the corresponding second derivative.

The prime symbol is used for derivatives within deSolve() only. In other cases, use **d** ().

The general solution of a 1st-order equation contains an arbitrary constant of the form ck, where k is an integer suffix from 1 through 255. The solution of a 2nd-order equation contains two such constants.

Apply **solve()** to an implicit solution if you want to try to convert it to one or more equivalent explicit solutions.

When comparing your results with textbook or manual solutions, be aware that different methods introduce arbitrary constants at different points in the calculation, which may produce different general solutions.

deSolve(IstOrderODE and initCond, Var, depVar) $\Rightarrow a particular solution$

Returns a particular solution that satisfies *IstOrderODE* and *initCond*. This is usually easier than determining a general solution, substituting initial values, solving for the arbitrary constant, and then substituting that value into the general solution.

initCond is an equation of the form:

depVar (initialIndependentValue) = initialDependentValue

$$\frac{v = (c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6}{vight(Ans) \rightarrow temp \quad (c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6}{\frac{d^2}{dx^2}(temp) + 2 \cdot \frac{d}{dx}(temp) + temp - x^2}$$
DelVar temp
$$\frac{d^2}{dx^2}(temp) = \frac{d^2}{dx^2}(temp) = \frac{d^2}{dx^2}(temp)$$

$$deSolve(y) = (cos(y))^2 \cdot x, x, y \qquad tan(y) = \frac{x^2}{2} + c4$$

solve(
$$Ans,v$$
) $y=tan^{-1}\left(\frac{x^2+2\cdot c4}{2}\right)+n3\cdot 2$
 $Ans|c4=c-1 \text{ and } n3=0$ $y=tan^{-1}\left(\frac{x^2+2\cdot (c-1)}{2}\right)$

$$\sin(y) = (y \cdot e^{x} + \cos(y)) \cdot y' \to ode$$

$$\sin(y) = (e^{x} \cdot y + \cos(y)) \cdot y'$$

$$\det \text{Solve}(ode \text{ and } y(0) = 0, x, y) \to soln$$

$$\frac{-(2 \cdot \sin(y) + y^{2})}{2} = -(e^{x} - 1) \cdot e^{-x} \cdot \sin(y)$$

soln x=0 and $y=0$	true
ode y'=impDif(soln,x,y)	true
DelVar ode.soln	Done

The initialIndependentValue and initialDependentValue can be variables such as x0 and y0 that have no stored values. Implicit differentiation can help verify implicit solutions.

deSolve

(2ndOrderODEandinitCondlandinitCond2, Var, depVar)⇒a particular solution

Returns a particular solution that satisfies 2nd Order ODE and has a specified value of the dependent variable and its first derivative at one point.

For *initCond1*, use the form:

depVar (initialIndependentValue) = initialDependentValue

For *initCond2*, use the form:

depVar (initialIndependentValue) = initial 1stDerivative Value

deSolve

(2ndOrderODEandbndCond1andbndCond2, Var, depVar)⇒a particular solution

Returns a particular solution that satisfies 2ndOrderODE and has specified values at two different points.

deSolve
$$\left(w'' - \frac{2 \cdot w'}{x} + \left(9 + \frac{2}{x^2}\right) \cdot w = x \cdot e^x \text{ and } w \left(\frac{\pi}{6}\right) = 0 \text{ and } w \left(\frac{\pi}{3}\right) = 0, x, w\right)$$

$$w = \frac{x \cdot e^x}{\left(\ln(e)\right)^2 + 9} + \frac{e^{\frac{\pi}{3}} \cdot x \cdot \cos(3 \cdot x)}{\left(\ln(e)\right)^2 + 9} - \frac{e^{\frac{\pi}{6}} \cdot x \cdot \sin(3 \cdot x)}{\left(\ln(e)\right)^2 + 9}$$

deSolve
$$y''=y$$
 $\frac{-1}{2}$ and $y(0)=0$ and $y'(0)=0$, ty $\frac{3}{2 \cdot y} \frac{1}{4} = t$

solve
$$\left(\frac{\frac{3}{2 \cdot y} \cdot \frac{3}{4}}{3} = t_{y}\right)$$

$$y = \frac{\frac{1}{3 \cdot 3} \cdot \frac{2}{3} \cdot \frac{4}{1}}{4} \text{ and } t \ge 0$$

deSolve(
$$y''=x$$
 and $y(0)=1$ and $y'(2)=3,x,y$)
 $y = \frac{x^3}{6} + x + 1$
deSolve($y''=2 \cdot y'$ and $y(3)=1$ and $y'(4)=2,x,y$)
 $y = \mathbf{e}^{2 \cdot x - 8} - \mathbf{e}^{-2} + 1$

det()

Catalogue > 🗓

det(squareMatrix[, Tolerance])⇒expression

Returns the determinant of squareMatrix.

Optionally, any matrix element is treated as zero if its absolute value is less than *Tolerance*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *Tolerance* is ignored.

- If you use <u>ctrl</u> <u>enter</u> or set the **Auto or Approximate** mode to Approximate, computations are done using floating-point arithmetic.
- If Tolerance is omitted or not used, the default tolerance is calculated as:

5E-14 ·max(dim(squareMatrix)) · rowNorm(squareMatrix)

_		
$\det\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	$a \cdot d -$	b·c
$\det\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$		-2
$\det \left[\text{identity}(3) - x \cdot \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & 1 \\ -6 & -2 & 7 \end{bmatrix} \right] $ $- \left(98 \cdot x^3 - 55 \cdot x^2 \right)$	2	۱.
-\98·x ² -55·x	$+12 \cdot x$	-1)
$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow mat1$	1. E 20	1
	0	1]
det(mat1)		0
det(mat1,.1)	1.1	E20

diag()	C	Catalogue > 🗊
$diag(List) \Rightarrow matrix$	diag([2 4 6])	2 0 0
diag(rowMatrix)⇒matrix		$\begin{bmatrix} 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$

 $diag(columnMatrix) \Rightarrow matrix$

Returns a matrix with the values in the argument list or matrix in its main diagonal.

diag(squareMatrix)⇒rowMatrix

Returns a row matrix containing the elements from the main diagonal of *squareMatrix*.

squareMatrix must be square.

4	6	8	4	6	8
1	2	3	1	2	3
5	7	9	_5	7	9]
dia	g(A	ns)	[4	2	9]

dim()		Catalogue > 🕎
dim(<i>List</i>)⇒ <i>integer</i>	$\dim(\{0,1,2\})$	3

Returns the dimension of *List*.

dim()

Catalogue > 🕮

$dim(Matrix) \Rightarrow list$

Returns the dimensions of matrix as a twoelement list {rows, columns}.

dim(*String*)⇒*integer*

Returns the number of characters contained in character string *String*.

$\dim \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{bmatrix}$	{3,2}
dim("Hello")	5
dim("Hello "&"there")	11

Disp

Catalogue > 🕮

Disp exprOrString1 [, exprOrString2] ...

Displays the arguments in the *Calculator* history. The arguments are displayed in succession, with thin spaces as separators.

Useful mainly in programs and functions to ensure the display of intermediate calculations.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

	0
	For i,start,end
	Disp i ," ",char (i)
	EndFor
	EndPrgm
	Done
chars(240,243)	
	240 ð
	241 ñ
	242 ò
	243 ó
	Done

Define chars(start.end)=Prgm

DispAt

Catalogue > 🕮

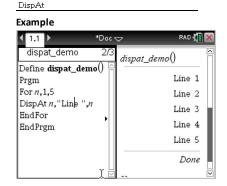
DispAt int, expr1 [,expr2 ...] ...

DispAt allows you to specify the line where the specified expression or string will be displayed on the screen.

The line number can be specified as an expression.

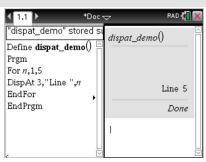
Please note that the line number is not for the entire screen but for the area immediately following the command/programme.

This command allows dashboard-like output from programmes where the value of an expression or from a sensor reading is updated on the same line.



DispAtand Disp can be used within the same programme.

Note: The maximum number is set to 8 since that matches a screen-full of lines on the handheld screen - as long as the lines don't have 2D maths expressions. The exact number of lines depends on the content of the displayed information.



Illustrative examples:

Define z()=	Output	
Prgm	z()	
For n,1,3	Iteration 1	:
DispAt 1,"N: ",n	Line	1: N:1
Disp "Hello"	Line 2	2: Hello
EndFor		
EndPrgm	Iteration 2	:
	Line	1: N:2
	Line 2	2: Hello
	Line 3	3: Hello
	Iteration 3	:
	Line	1: N:3
	Line 2	2: Hello
	Line 3	3: Hello
	Line 4	1: Hello
Define z1()=	z1()	
Prgm	Line	1: N:3
For n,1,3	Line 2	2: Hello
DispAt 1,"N: ",n	Line 3	3: Hello
EndFor	Line 4	1: Hello
	Line 5	5: Hello
For n,1,4		
Disp "Hello"		
EndFor		
EndPrgm		

Error conditions:

Error Message	Description
DispAt line number must be between 1 and 8	Expression evaluates the line number outside the range 1-8 (inclusive)
Too few arguments	The function or command is missing one or more arguments.
No arguments	Same as current 'syntax error' dialogue
Too many arguments	Limit argument. Same error as Disp.
Invalid data type	First argument must be a number.
Void: DispAt void	"Hello World" Datatype error is thrown for the void (if the callback is defined)
Conversion operator: DispAt 2_ft @> _m, "Hello World"	CAS: Datatype Error is thrown (if the callback is defined)
	Numeric: Conversion will be evaluated and if the result is a valid argument, DispAt print the string at the result line.

DMS	Catalogue > 🗓	Ź
-----	---------------	---

Expr DMS

List ▶DMS

Matrix ▶DMS

Note: You can insert this operator from the computer keyboard by typing @>DMS.

Interprets the argument as an angle and displays the equivalent DMS (DDDDDD°MM'SS.ss") number. See °, ', " (page 225) for DMS (degree, minutes, seconds) format.

Note: ▶DMS will convert from radians to degrees when used in radian mode. If the input is followed by a degree symbol o, no conversion will occur. You can use ▶DMS only at the end of an entry line.

In Degree angle mode:

(45.371)▶DMS	45°22'15.6"
({45.371,60})▶DMS	{45°22'15.6",60°}

domain()

Catalogue > 😰

 $domain(Expr1, Var) \Rightarrow expression$

Returns the domain of Expr1 with respect to Var.

domain() can be used to examine domains of functions. It is restricted to real and finite domain.

This functionality has limitations due to shortcomings of computer algebra simplification and solver algorithms.

Certain functions cannot be used as arguments for domain(), regardless of whether they appear explicitly or within user-defined variables and functions. In the following example, the expression cannot be simplified because \int () is a disallowed function.

domain
$$\begin{bmatrix} x \\ \frac{1}{t} dt, x \end{bmatrix}$$
 + domain $\begin{bmatrix} x \\ \frac{1}{t} dt, x \end{bmatrix}$

$$\operatorname{domain}\left(\frac{1}{x+y}, \nu\right) \qquad -\infty < y < x \text{ or } -x < y < \infty$$

$$\operatorname{domain}\left(\frac{x+1}{x^2+2\cdot x}, x\right) \qquad x \neq -2 \text{ and } x \neq 0$$

$$\operatorname{domain}\left(\left(\sqrt{x}\right)^2, x\right) \qquad 0 \leq x < \infty$$

$$\operatorname{domain}\left(\frac{1}{x+y}, \nu\right) \qquad -\infty < y < x \text{ or } -x < y < \infty$$

dominantTerm()

dominantTerm(Expr1, Var [, Point]) $\Rightarrow expression$

dominantTerm(Expr1, Var [, Point]) | $Var > Point \Rightarrow expression$

dominantTerm(*Expr1*, *Var* [, *Point*]) | *Var<Point* ⇒*expression*

Returns the dominant term of a power series representation of ExprI expanded about Point. The dominant term is the one whose magnitude grows most rapidly near Var = Point. The resulting power of (Var - Point) can have a negative and/or fractional exponent. The coefficient of this power can include logarithms of (Var - Point) and other functions of Var that are dominated by all powers of (Var - Point) having the same exponent sign.

Catalogue > 📳

$$\frac{x^7}{30}$$

$$\frac{x^7}{30}$$

$$\operatorname{dominantTerm}\left(\frac{1-\cos(x-1)}{(x-1)^3},x,1\right) \qquad \frac{1}{2\cdot(x-1)}$$

$$\operatorname{dominantTerm}\left(x^{-2}\cdot\tan\left(x^{-3}\right),x\right) \qquad \frac{1}{\frac{5}{x^3}}$$

$$\operatorname{dominantTerm}\left(\ln\left(x^x-1\right)\cdot x^{-2},x\right) \qquad \frac{\ln(x\cdot\ln(x))}{x^2}$$

dominantTerm()

Catalogue > 🕮

Point defaults to 0. *Point* can be ∞ or $-\infty$. in which cases the dominant term will be the term having the largest exponent of Var rather than the smallest exponent of Var.

dominantTerm(...) returns "dominantTerm (...)" if it is unable to determine such a representation, such as for essential singularities such as sin(1/z) at z=0, $e^{-1/z}$ at z=0, or e^z at z = ∞ or $-\infty$.

If the series or one of its derivatives has a iump discontinuity at *Point*, the result is likely to contain sub-expressions of the form sign(...) or abs(...) for a real expansion variable or (-1)floor(...angle(...)...) for a complex expansion variable, which is one ending with ". If you intend to use the dominant term only for values on one side of *Point*. then append to dominantTerm(...) the appropriate one of "| Var > Point", "| Var < Point", "| " $Var \ge Point"$, or " $Var \le$ Point" to obtain a simpler result.

dominantTerm() distributes over 1stargument lists and matrices.

dominantTerm() is useful when you want to know the simplest possible expression that is asymptotic to another expression as $Var \rightarrow Point$. dominantTerm() is also useful when it isn't obvious what the degree of the first non-zero term of a series will be, and you don't want to iteratively guess either interactively or by a programme loop.

Note: See also series(), page 158.

$\operatorname{dominantTerm}\left(e^{\frac{-1}{z}},z\right)$	
dominantTerr	$ \operatorname{m}\left(e^{\frac{-1}{z}},z,0\right) $
dominantTerm $\left(1+\frac{1}{n}\right)^n, n, \infty$	е
dominantTerm $\left(\tan^{-1}\left(\frac{1}{x}\right), x, 0\right)$	$\frac{\pi \cdot \operatorname{sign}(x)}{2}$
dominantTerm $\left(\tan^{-1}\left(\frac{1}{x}\right),x\right) x>0$	$\frac{\pi}{2}$

dotP() Catalogue > 🕮

 $dotP(List1, List2) \Rightarrow expression$

Returns the "dot" product of two lists.

 $dotP(Vector1, Vector2) \Rightarrow expression$

Returns the "dot" product of two vectors.

Both must be row vectors, or both must be column vectors.

$\overline{\det(\{a,b,c\},\{d,e,f\})}$	$a \cdot d + b \cdot e + c \cdot f$
$dotP(\{1,2\},\{5,6\})$	17
$\frac{1}{\det P([a \ b \ c],[d \ e \ f])}$	$a \cdot d + b \cdot e + c \cdot f$
dotP([1 2 3],[4 5 6])	32

2.

e^()		e ^x key
$e^{(Exprl)} \Rightarrow expression$	e^1	е
Returns ${\it e}$ raised to the ${\it Expr1}$ power.	e ^{1.}	2.71828
Note: See also <i>e</i> exponent template, page	-3 ²	e 9

Note: Pressing e^x to display e^x is different from pressing the character E on the keyboard.

You can enter a complex number in $re^{i\theta}$ polar form. However, use this form in Radian angle mode only; it causes a Domain error in Degree or Gradian angle mode.

$$e^{(List 1)} \Rightarrow list$$

Returns e raised to the power of each element in List 1.

 $e^{(squareMatrix1)} \Rightarrow squareMatrix$

Returns the matrix exponential of squareMatrix I. This is not the same as calculating e raised to the power of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

$e^{\{1,1.,0.5\}}$	{ e ,2.71828,1.64872 }

	1	5	3			456.509	
		2		680.546	488.795	396.521	
e	6	-2	1	524.929	371.222	307.879	

eff() Catalogue > \mathbb{Z} eff(nominalRate,CpY) \Rightarrow value eff(5.75,12) 5.90398

Financial function that converts the nominal interest rate nominalRate to an annual effective rate, given CpY as the number of compounding periods per year.

nominalRate must be a real number, and CpY must be a real number > 0.

Note: See also nom(), page 121.

$eigVc(squareMatrix) \Rightarrow matrix$

Returns a matrix containing the eigenvectors for a real or complex squareMatrix, where each column in the result corresponds to an eigenvalue. Note that an eigenvector is not unique; it may be scaled by any constant factor. The eigenvectors are normalized, meaning that:

if
$$V = [x_1, x_2, ..., x_n]$$

then
$$x_1^2 + x_2^2 + ... + x_n^2 = 1$$

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The *squareMatrix* is then reduced to upper Hessenberg form and the eigenvectors are computed via a Schur factorization.

In Rectangular Complex Format:

_						
[-1	2	5	[-	1	2	5
3	-6	$9 \rightarrow m1$		3	-6	9
2	-5	7]	Į:	2	-5	7
eig	Vc(n	11)				

-0.800906 0.767947

0.484029 0.573804+0.052258·i 0.5738) 0.352512 0.262687+0.096286·i 0.2626

To see the entire result, press and then use ◀ and ▶ to move the cursor.

eigVI()

 $eigVl(squareMatrix) \Rightarrow list$

Returns a list of the eigenvalues of a real or complex squareMatrix.

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The squareMatrix is then reduced to upper Hessenberg form and the eigenvalues are computed from the upper Hessenberg matrix.

Catalogue > 23

-1	2	5]	-1	2	5
3	-6		3	-6	9
2	-5	7	2	-5	7

In Rectangular complex format mode:

eigVl(m1)

-4.40941,2.20471+0.763006·*i*,2.20471-0.

To see the entire result, press - and then use ◀ and ▶ to move the cursor.

Else

See If, page 86.

Catalogue > 🔯 ElseIf If BooleanExpr1 Then Define g(x)=Func Block1 If $x \le -5$ Then Elself BooleanExpr2 Then Return 5 Block2 ElseIf x > -5 and x < 0 Then Return -x Elself Boolean ExprN Then ElseIf $x \ge 0$ and $x \ne 10$ Then BlockNReturn x EndIf ElseIf x=10 Then Return 3 EndIf Note for entering the example: For EndFunc instructions on entering multi-line Done programme and function definitions, refer to the Calculator section of your product guidebook. **EndFor** See For, page 72. **EndFunc** See Func, page 75. **EndIf** See If, page 86. EndLoop See Loop, page 108. **EndPrgm** See Prgm, page 135.

EndTry

EndWhile

See Try, page 188.

See While, page 198.

euler ()

Catalogue > 🕮

euler(Expr, Var, depVar, {Var0, VarMax}, depVar0, $VarStep[, eulerStep]) \Rightarrow matrix$

euler(SystemOfExpr, Var, ListOfDepVars, $\{Var0, VarMax\},$ ListOfDepVars0, $VarStep[, eulerStep]) \Rightarrow matrix$

euler(ListOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, $VarStep[, eulerStep]) \Rightarrow matrix$

Uses the Euler method to solve the system $\frac{d \ depVar}{} = Expr(Var, depVar)$

with depVar(Var0)=depVar0 on the interval [Var0.VarMax]. Returns a matrix whose first row defines the Var output values and whose second row defines the value of the first solution component at the corresponding *Var* values, and so on.

Expr is the right-hand side that defines the ordinary differential equation (ODE).

SystemOfExpr is the system of right-hand sides that define the system of ODEs (corresponds to order of dependent variables in *ListOfDepVars*).

ListOfExpr is a list of right-hand sides that define the system of ODEs (corresponds to the order of dependent variables in ListOfDepVars).

Var is the independent variable.

ListOfDepVars is a list of dependent variables.

{Var0, VarMax} is a two-element list that tells the function to integrate from Var0 to VarMax.

ListOfDepVars0 is a list of initial values for dependent variables.

Differential equation: y'=0.001*y*(100-y) and y(0)=10

euler
$$(0.001 \cdot y \cdot (100 - y), t, y, \{0,100\}, 10, 1)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9 & 11.8712 & 12.9174 & 14.042 \end{bmatrix}$$

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

Compare above result with CAS exact solution obtained using deSolve() and seqGen():

$$\begin{aligned} \text{deSolve} & \big(y' = 0.001 \cdot y \cdot \big(100 - y \big) \text{ and } y(0) = 10.t.y \big) \\ & y = \frac{100 \cdot \big(1.10517 \big)^t}{\big(1.10517 \big)^t + 9.} \end{aligned}$$

seqGen
$$\left(\frac{100.\cdot(1.10517)^{t}}{(1.10517)^{t}+9.},t_{y},\{0,100\}\right)$$

 $\left\{10.,10.9367,11.9494,13.0423,14.2189\right\}$

System of equations:

$$\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}$$

with v1(0)=2 and v2(0)=5

$$\begin{aligned} \text{eulef} & \left\{ \begin{matrix} yI + 0.1 \cdot yI \cdot y2 \\ 3 \cdot y2 - yI \cdot y2 \end{matrix} \right. t, \left\{ yI \cdot y2 \right\}, \left\{ 0,5 \right\}, \left\{ 2,5 \right\}, 1 \right) \\ & \left[\begin{matrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{matrix} \right] \end{aligned}$$

Catalogue > 23

euler ()

VarStep is a nonzero number such that sign (VarStep) = sign(VarMax-Var0) and solutions are returned at $Var0+i \cdot VarStep$ for all i=0,1,2,... such that $Var0+i \cdot VarStep$ is in [var0,VarMax] (there may not be a solution value at VarMax).

eulerStep is a positive integer (defaults to 1) that defines the number of euler steps between output values. The actual step size used by the euler method is VarStep / eulerStep.

eval () Hub Menu

 $eval(Expr) \Rightarrow string$

eval() is valid only in the TI-Innovator™ Hub Command argument of programming commands Get, GetStr and Send. The software evaluates expression Expr and replaces the eval() statement with the result as a character string.

The argument *Expr* must simplify to a real number.

Set the blue element of the RGB LED to half intensity.

lum:=127	127
Send "SET COLOR.BLUE eval(lum)"	Done

Reset the blue element to OFF.

Send "SET COLOR.BLUE OFF"	Done
---------------------------	------

eval() argument must simplify to a real number.

Send "SET LED eval("4") TO ON"
"Error: Invalid data type"

Programme to fade-in the red element

Define fadein()=
Prgm
For i,0,255,10
Send "SET COLOR.RED eval(i)"
Wait 0.1
EndFor
Send "SET COLOR.RED OFF"
EndPrgm

Execute the programme.

fadein()	Done
----------	------

eval () Hub Menu

Although eval() does not display its result, you can view the resulting Hub command string after executing the command by inspecting any of the following special variables.

iostr.SendAns iostr.GetAns iostr.GetStrAns

Note: See also Get (page 77), GetStr (page 83), and Send (page 156).



exact()
exact(Expr1 [, Tolerance]) ⇒ expression

equivalent of the argument.

exact(List1 [, Tolerance]) ⇒ list exact(Matrix1 [, Tolerance]) ⇒ matrix Uses Exact mode arithmetic to return, when possible, the rational-number

Tolerance specifies the tolerance for the conversion; the default is 0 (zero).

exact(0.25)	<u>1</u>
	4
exact(0.333333)	333333
	1000000
exact(0.333333,0.001)	1
	3
$\operatorname{exact}(3.5 \cdot x + y)$	$\frac{7 \cdot x}{2} + y$
exact({0.2,0.33,4.125})	$\left\{\frac{1}{5}, \frac{33}{100}, \frac{33}{8}\right\}$
	5 100 8

Catalogue > 13

Exit Catalogue > 13

Exit

Exits the current For, While, or Loop block.

Exit is not allowed outside the three looping structures (**For**, **While**, or **Loop**).

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Function listing:

Define g()=Func Done
Local temp, i $0 \rightarrow temp$ For i, 1, 100, 1 $temp+i \rightarrow temp$ If $temp \ge 20$ Then
Exit
EndIf
EndFor
EndFunc g()21

▶ exp

Catalogue > 🗐

e× kev

Expr \triangleright exp

Represents Expr in terms of the natural exponential e. This is a display conversion operator. It can be used only at the end of the entry line.

Note: You can insert this operator from the computer keyboard by typing @>exp.

$\frac{d}{dx} \left(\mathbf{e}^{x} + \mathbf{e}^{-x} \right)$	2• sinh(x)
2· sinh(x)▶exp	$e^{x}-e^{-x}$

exp()

 $exp(Expr1) \Rightarrow expression$

Returns **e** raised to the *Expr1* power.

Note: See also e exponent template, page 2.

You can enter a complex number in $re^{i\theta}$ polar form. However, use this form in Radian angle mode only; it causes a Domain error in Degree or Gradian angle mode.

 $\exp(List1) \Rightarrow list$

Returns **e** raised to the power of each element in *List1*.

 $\exp(squareMatrix 1) \Rightarrow squareMatrix$

Returns the matrix exponential of squareMatrix I. This is not the same as calculating e raised to the power of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

e ¹	е
e ^{1.}	2.71828
e^{3^2}	e ⁹

 $e^{\{1,1.,0.5\}}$ {e,2.71828,1.64872}

	1	5	3	782.209	559.617	456.509
	4	2	1	680.546	488.795	396.521
е	6	-2	1	524.929	371.222	307.879

exp▶list()

 $exp \triangleright list(Expr, Var) \Rightarrow list$

Catalogue > 🗐

$$\frac{\text{solve}(x^2 - x - 2 = 0, x)}{\text{exp} \cdot \text{list}(\text{solve}(x^2 - x - 2 = 0, x), x)} \qquad x = 1 \text{ or } x = 2$$

exp ► list()

Examines *Expr* for equations that are separated by the word "or," and returns a list containing the right-hand sides of the equations of the form Var=Expr. This gives you an easy way to extract some solution values embedded in the results of the solve(), cSolve(), fMin(), and fMax() functions.

Note: exp ► list() is not necessary with the zeros() and cZeros() functions because they return a list of solution values directly.

You can insert this function from the keyboard by typing exp@>list(...).

expand()

 $expand(Expr1 [, Var]) \Rightarrow expression$ $expand(List1 [,Var]) \Rightarrow list$ $expand(Matrix1 [,Var]) \Rightarrow matrix$

expand(Expr1) returns Expr1 expanded with respect to all its variables. The expansion is polynomial expansion for polynomials and partial fraction expansion for rational expressions.

The goal of **expand()** is to transform Expr1into a sum and/or difference of simple terms. In contrast, the goal of factor() is to transform *Expr1* into a product and/or quotient of simple factors.

expand(Expr1, Var) returns Expr1expanded with respect to Var. Similar powers of *Var* are collected. The terms and their factors are sorted with *Var* as the main variable. There might be some incidental factoring or expansion of the collected coefficients. Compared to omitting Var, this often saves time, memory, and screen space, while making the expression more comprehensible.

Catalogue > 23

$$\frac{x^2 + 2 \cdot x \cdot y + 2 \cdot x + y^2 + 2 \cdot y + 1}{x^2 + 2 \cdot x \cdot y + 2 \cdot x + y^2 + 2 \cdot y + 1}$$

$$\frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2 + x \cdot y}$$

$$\frac{1}{x - 1} - \frac{1}{x} + \frac{1}{y - 1} - \frac{1}{y}$$

$$\frac{\operatorname{expand}((x+y+1)^2,y) - y^2 + 2 \cdot y \cdot (x+1) + (x+1)^2}{\operatorname{expand}((x+y+1)^2,x) - x^2 + 2 \cdot x \cdot (y+1) + (y+1)^2}$$

$$\operatorname{expand}\left(\frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2 + x \cdot y}, y\right)$$

$$\frac{1}{y-1} - \frac{1}{y} + \frac{1}{x \cdot (x-1)}$$

$$\operatorname{expand}(Ans,x) - \frac{1}{x-1} - \frac{1}{x} + \frac{1}{y \cdot (y-1)}$$

expand()

Catalogue > 23

Even when there is only one variable, using Var might make the denominator factorization used for partial fraction expansion more complete.

expand(Ans,x)

Hint: For rational expressions, propFrac() is a faster but less extreme alternative to expand().

Note: See also comDenom() for an expanded numerator over an expanded denominator.

expand(Expr1,[Var]) also distributes logarithms and fractional powers regardless of *Var*. For increased distribution of logarithms and fractional powers, inequality constraints might be necessary to guarantee that some factors are nonnegative.

expand(Expr1, [Var]) also distributes absolute values, sign(), and exponentials, regardless of Var.

Note: See also **tExpand()** for trigonometric

$\ln(2\cdot x\cdot y) + \sqrt{2\cdot x\cdot y}$	
expand(Ans)	$\ln(x \cdot y) + \sqrt{2} \cdot \sqrt{x \cdot y} + \ln(2)$
$expand(Ans) y\ge 0$	1
	$\ln(x) + \sqrt{2} \cdot \sqrt{x} \cdot \sqrt{y} + \ln(y) + \ln(2)$
$sign(x\cdot y)+ x\cdot y +c$	$e^{2 \cdot x + y}$
	$e^{2\cdot x+y} + \operatorname{sign}(x\cdot y) + x\cdot y $

 $\operatorname{sign}(x) \cdot \operatorname{sign}(y) + |x| \cdot |y| + (e^x)^2 \cdot e^y$

angle-sum and multiple-angle expansion.

expr() Catalogue > 🕮

 $expr(String) \Rightarrow expression$

Returns the character string contained in String as an expression and immediately executes it.

expr("1+2+x^2+x")	x^2+x+3
expr("expand((1+x)^2)")	$x^2 + 2 \cdot x + 1$

"Define cube(x)= x^3 " \rightarrow funcstr

expand(Ans)

"Define cube(x)= x^3 " expr(funcstr) Done cube(2)8

Catalogue > 🗐 **ExpReg**

ExpReg X, Y [, [Freq] [, Category, Include]]

Computes the exponential regression $v = a \cdot (b)^x$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable. (See page 174.)

Catalogue > 23

ExpReg

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

Output variable	Description
stat.RegEqn	Regression equation: a•(b) ^x
stat.a, stat.b	Regression coefficients
stat.r ²	Coefficient of linear determination for transformed data
stat.r	Correlation coefficient for transformed data (x, ln(y))
stat.Resid	Residuals associated with the exponential model
stat.ResidTrans	Residuals associated with linear fit of transformed data
stat.XReg	List of data points in the modified $X\ List$ actually used in the regression based on restrictions of $Freq$, $Category\ List$, and $Include\ Categories$
stat.YReg	List of data points in the modified <i>Y List</i> actually used in the regression based on restrictions of <i>Freq</i> , <i>Category List</i> , and <i>Include Categories</i>
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

factor() Catalogue > 23

 $factor(Expr1[, Var]) \Rightarrow expression$

 $factor(List1[Var]) \Rightarrow list$

 $factor(Matrix 1[Var]) \Rightarrow matrix$

factor(Expr1) returns Expr1 factored with respect to all of its variables over a common denominator.

Expr1 is factored as much as possible toward linear rational factors without introducing new non-real subexpressions. This alternative is appropriate if you want factorization with respect to more than one variable.

factor(Expr1,Var) returns Expr1 factored with respect to variable Var.

Expr1 is factored as much as possible toward real factors that are linear in Var. even if it introduces irrational constants or subexpressions that are irrational in other variables.

The factors and their terms are sorted with Var as the main variable. Similar powers of Var are collected in each factor. Include Var if factorization is needed with respect to only that variable and you are willing to accept irrational expressions in any other variables to increase factorization with respect to Var. There might be some incidental factoring with respect to other variables.

For the Auto setting of the Auto or **Approximate** mode, including *Var* permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including Varmight yield more complete factorization.

$$\begin{array}{c|c} \hline {\rm factor} \Big(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a \Big) \\ & a \cdot (a-1) \cdot (a+1) \cdot (x-1) \cdot (x+1) \\ \hline {\rm factor} \Big(x^2 + 1 \Big) & x^2 + 1 \\ \hline {\rm factor} \Big(x^2 - 4 \Big) & (x-2) \cdot (x+2) \\ \hline {\rm factor} \Big(x^2 - 3 \Big) & x^2 - 3 \\ \hline {\rm factor} \Big(x^2 - a \Big) & x^2 - a \\ \hline \end{array}$$

$$\frac{factor(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3)}{x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3}$$

$$factor(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$$

$$(x-0.964673)\cdot(x+0.611649)\cdot(x+2.12543)\cdot(x^4-611649)\cdot(x^4-611649)\cdot($$

Note: See also **comDenom()** for a fast way to achieve partial factoring when **factor()** is not fast enough or if it exhausts memory.

Note: See also **cFactor()** for factoring all the way to complex coefficients in pursuit of linear factors.

factor(rationalNumber) returns the rational number factored into primes. For composite numbers, the computing time grows exponentially with the number of digits in the second-largest factor. For example, factoring a 30-digit integer could take more than a day, and factoring a 100-digit number could take more than a century.

To stop a calculation manually,

- Handheld: Hold down the figure on key and press enter repeatedly.
- Windows®: Hold down the F12 key and press Enter repeatedly.
- Macintosh®: Hold down the F5 key and press Enter repeatedly.
- iPad[®]: The app displays a prompt. You can continue waiting or cancel.

If you merely want to determine if a number is prime, use **isPrime()** instead. It is much faster, particularly if *rationalNumber* is not prime and if the second-largest factor has more than five digits.

factor(152417172689)	123457 · 1234577
isPrime(152417172689)	false

FCdf() Catalogue > 43

FCdf

(lowBound,upBound,dfNumer,dfDenom)⇒number if lowBound and upBound are numbers, list if lowBound and upBound are lists

FCdf

(lowBound,upBound,dfNumer,dfDenom)⇒number if lowBound and upBound are numbers, list if lowBound and upBound are lists

Computes the F distribution probability between *lowBound* and *upBound* for the specified *dfNumer* (degrees of freedom) and *dfDenom*.

For $P(X \le upBound)$, set lowBound = 0.

Fill	Catalogue >	Q
------	-------------	---

Fill Expr, matrixVar⇒matrix

Replaces each element in variable matrix Var with Expr.

matrix Var must already exist.

Fill Expr, $listVar \Rightarrow list$

Replaces each element in variable listVar with Expr.

listVar must already exist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow amatrix$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
Fill 1.01,amatrix	Done
amatrix	1.01 1.01
	1.01 1.01

$\{1,2,3,4,5\} \rightarrow$	alist	{1,2,3,4,5}
Fill 1.01,alist		Done
alist	{1.01,1	.01,1.01,1.01,1.01

FiveNumSummary

Catalogue > 23

FiveNumSummary X[,[Freq][,Category,Include]]

Provides an abbreviated version of the 1-variable statistics on list X. A summary of results is stored in the stat.results variable (page 174).

X represents a list containing the data.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1.

Category is a list of numeric category codes for the corresponding X data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

An empty (void) element in any of the lists *X*, *Freq*, or *Category* results in a void for the corresponding element of all those lists. For more information on empty elements, see page 248.

Output variable	Description
stat.MinX	Minimum of x values.
stat.Q ₁ X	1st Quartile of x.
stat.MedianX	Median of x.
stat.Q ₃ X	3rd Quartile of x.
stat.MaxX	Maximum of x values.

noor()	floor()	Catalogue > 🗊
--------	---------	---------------

 $floor(Expr1) \Rightarrow integer$

floor(-2.14) -3.

Returns the greatest integer that is \leq the argument. This function is identical to int().

The argument can be a real or a complex number.

 $floor(List1) \Rightarrow list$

 $floor(Matrix 1) \Rightarrow matrix$

Returns a list or matrix of the floor of each element.

Note: See also ceiling() and int().

floor $\left\{ \frac{3}{2}, 0, -5.3 \right\}$	{1,0,-6.}
floor $\begin{bmatrix} 1.2 & 3.4 \\ 2.5 & 4.8 \end{bmatrix}$	1. 3. 2. 4.
(Lis non	L=

fMax() Catalogue > [3]

 $fMax(Expr, Var) \Rightarrow Boolean \ expression$

fMax(Expr, Var, lowBound)

fMax $\left(1-(x-a)^2-(x-b)^2,x\right)$ $x = \frac{a+b}{2}$ fMax $\left(.5 \cdot x^3 - x - 2,x\right)$ $x = \infty$

fMax(Expr, Var,lowBound,upBound)

fMax(*Expr*, *Var***)** | *lowBound*≤*Var* ≤*upBound*

Returns a Boolean expression specifying candidate values of Var that maximise Expr or locate its least upper bound.

You can use the constraint ("|") operator to restrict the solution interval and/or specify other constraints.

$$f_{\text{Max}}(0.5 \cdot x^3 - x - 2, x)|_{x \le 1}$$
 $x = 0.816497$

For the Approximate setting of the Auto or Approximate mode, fMax() iteratively searches for one approximate local maximum. This is often faster, particularly if you use the "|" operator to constrain the search to a relatively small interval that contains exactly one local maximum.

Note: See also fMin() and max().

fMin()

Catalogue > 🗐

 $\chi = -\infty$ or $\chi = \infty$

x=1.

 $fMin(1-(x-a)^2-(x-b)^2,x)$

 $fMin(0.5 \cdot x^3 - x - 2.x)|_{x \ge 1}$

fMin(*Expr*, *Var***)**⇒*Boolean expression*

fMin(Expr, Var,lowBound)

fMin(Expr, Var,lowBound,upBound)

fMin(*Expr***,** *Var***)** | *lowBound*≤*Var* ≤ *upBound*

Returns a Boolean expression specifying candidate values of Var that minimise Expr or locate its greatest lower bound.

You can use the constraint ("|") operator to restrict the solution interval and/or specify other constraints.

For the Approximate setting of the Auto or Approximate mode, fMin() iteratively searches for one approximate local minimum. This is often faster, particularly if you use the "|" operator to constrain the search to a relatively small interval that contains exactly one local minimum.

Note: See also fMax() and min().

For

Catalogue > 🗐

For Var, Low, High [, Step]

Block

EndFor

Executes the statements in Block iteratively for each value of Var, from Low to High, in increments of Step.

Var must not be a system variable.

Step can be positive or negative. The default value is 1.

Block can be either a single statement or a series of statements separated with the ":" character.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define $g()$ =Func	Done
Local tempsum, step, i	
$0 \rightarrow tempsum$	
$1 \rightarrow step$	
For <i>i</i> ,1,100, <i>step</i>	
$tempsum+i \rightarrow tempsum$	
EndFor	
EndFunc	
g()	5050

format()

format(*Expr*[, *formatString*])⇒*string*

Returns ${\it Expr}$ as a character string based on the format template.

Expr must simplify to a number.

formatString is a string and must be in the form: "F[n]", "S[n]", "E[n]", "G[n][c]", where [] indicate optional portions.

F[n]: Fixed format. n is the number of digits to display after the decimal point.

S[n]: Scientific format. n is the number of digits to display after the decimal point.

E[n]: Engineering format. n is the number of digits after the first significant digit. The exponent is adjusted to a multiple of three, and the decimal point is moved to the right by zero, one, or two digits.

format(1.234567, "f3")	"1.235"
format(1.234567,"s2")	"1.23E0"
format(1.234567,"e3")	"1.235e0"
format(1.234567, "g3")	"1.235"
format(1234.567, "g3")	"1,234.567"
format(1.234567, "g3,r:")	"1:235"

Catalogue > 🕮

G[n][c]: Same as fixed format but also separates digits to the left of the radix into groups of three. c specifies the group separator character and defaults to a comma. If c is a period, the radix will be shown as a comma.

[Rc]: Any of the above specifiers may be suffixed with the Rc radix flag, where c is a single character that specifies what to substitute for the radix point.

fPart() Catalogue > 🗓

 $fPart(Expr1) \Rightarrow expression$

fPart(-1.234) -0.234 fPart({1,-2.3,7.003}) {0,-0.3,0.003}

 $fPart(List 1) \Rightarrow list$

 $fPart(Matrix 1) \Rightarrow matrix$

Returns the fractional part of the argument.

For a list or matrix, returns the fractional parts of the elements.

The argument can be a real or a complex number.

FPdf() Catalogue > [3]

 $\mathsf{FPdf}(XVal, dfNumer, dfDenom) \Rightarrow number \text{ if } XVal \text{ is a number, } list \text{ if } XVal \text{ is a list}$

Computes the F distribution probability at XVal for the specified dfNumer (degrees of freedom) and dfDenom.

freqTable▶list()

Catalogue > 🗐

 $freqTable \gt list(List1, freqIntegerList) \Rightarrow list$

Returns a list containing the elements from List1 expanded according to the frequencies in freqIntegerList. This function can be used for building a frequency table for the Data & Statistics application.

List1 can be any valid list.

freqIntegerList must have the same dimension as List1 and must contain nonnegative integer elements only. Each element specifies the number of times the corresponding List1 element will be repeated in the result list. A value of zero excludes the corresponding List1 element.

Note: You can insert this function from the computer keyboard by typing freqTable@>list(...).

Empty (void) elements are ignored. For more information on empty elements, see page 248.

frequency()

Catalogue > 🕮

 $frequency(List1,binsList) \Rightarrow list$

Returns a list containing counts of the elements in *List1*. The counts are based on ranges (bins) that you define in *binsList*.

If binsList is $\{b(1), b(2), ..., b(n)\}$, the specified ranges are $\{? \le b(1), b(1) < ? \le b(2), ..., b(n-1) < ? \le b(n), b(n) > ?\}$. The resulting list is one element longer than binsList.

Each element of the result corresponds to the number of elements from List1 that are in the range of that bin. Expressed in terms of the **countif()** function, the result is { countif(list, ? \leq b(1)), countif(list, b(1)<? \leq b(2)), ..., countif(list, b(n-1)<? \leq b(n)), countif (list, b(n)>?)}.

Elements of List 1 that cannot be "placed in a bin" are ignored. Empty (void) elements are also ignored. For more information on empty elements, see page 248.

Within the Lists & Spreadsheet application, you can use a range of cells in place of both arguments.

Note: See also countif(), page 35.

 $\begin{aligned} \textit{datalist:} &= \left\{1,2,\mathbf{e},3,\pi,4,5,6,\text{"hello"},7\right\} \\ &= \left\{1,2,2.71828,3,3.14159,4,5,6,\text{"hello"},7\right\} \\ &= \text{frequency}(\textit{datalist},\left\{2.5,4.5\right\}) \end{aligned}$

Explanation of result:

- **2** elements from Datalist are ≤ 2.5
- **4** elements from Datalist are >2.5 and \leq 4.5
- 3 elements from Datalist are >4.5

The element "hello" is a string and cannot be placed in any of the defined bins.

FTest_2Samp List1,List2[,Freq1[,Freq2[,Hypoth]]]

FTest_2Samp List1,List2[,Freq1[,Freq2[,Hypoth]]]

(Data list input)

FTest_2Samp sx1,n1,sx2,n2[,Hypoth]

FTest_2Samp sx1,n1,sx2,n2[,Hypoth]

(Summary stats input)

Performs a two-sample F test. A summary of results is stored in the *stat.results* variable (page 174).

For H_a : $\sigma 1 > \sigma 2$, set Hypoth > 0

For H_a : $\sigma 1 \neq \sigma 2$ (default), set Hypoth = 0

For H_a : $\sigma 1 < \sigma 2$, set Hypoth < 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.F	Calculated F statistic for the data sequence
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.dfNumer	numerator degrees of freedom = n1-1
stat.dfDenom	denominator degrees of freedom = n2-1
stat.sx1, stat.sx2	Sample standard deviations of the data sequences in $\mathit{List}\ 1$ and $\mathit{List}\ 2$
stat.x1_bar	Sample means of the data sequences in List 1 and List 2
stat.x2_bar	
stat.n1, stat.n2	Size of the samples

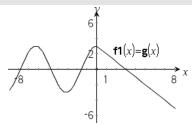
Func	Catalogi	ne > রীর
Func Block	Define a piecewise function:	
EndFunc	Define $g(x)$ =Func	Done
- 1. 6 1.6. 1.	If $x < 0$ Then	
Template for creating a user-defined	Return $3 \cdot \cos(x)$	
function.	Else	
	Return 3–x	
	EndIf	
	EndFunc	
	Result of graphing g(x)	

Func

Catalogue > 🕡

Block can be a single statement, a series of statements separated with the ":" character, or a series of statements on separate lines. The function can use the **Return** instruction to return a specific result.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.



G

gcd() Catalogue > [3]

gcd(Number1, Number2**)**⇒expression

gcd(18,33) 3

Returns the highest common factor of the two arguments. The **gcd** of two fractions is the **gcd** of their numerators divided by the **lcm** of their denominators.

In Auto or Approximate mode, the **gcd** of fractional floating-point numbers is 1.0.

$$gcd(List1, List2) \Rightarrow list$$

 $\gcd(\{12,14,16\},\{9,7,5\}) \qquad \{3,7,1\}$

Returns the highest common factors of the corresponding elements in *List1* and *List2*.

$$gcd(Matrix1, Matrix2) \Rightarrow matrix$$

Returns the highest common factors of the corresponding elements in *Matrix I* and *Matrix 2*.

$$\gcd\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}, \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix} \qquad \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

geomCdf()

Catalogue > 👰

geomCdf(*p,lowBound,upBound***)**⇒*number* if *lowBound* and *upBound* are numbers, *list* if *lowBound* and *upBound* are lists

geomCdf(p,upBound**)**for P($1 \le X \le upBound$) $\Rightarrow number$ if upBound is a number, list if upBound is a list

Computes a cumulative geometric probability from lowBound to upBound with the specified probability of success p.

Catalogue > 🕮

geomCdf()

For $P(X \le upBound)$, set lowBound = 1.

geomPdf() Catalogue > 🗐

geomPdf(p,XVal**)** \Rightarrow number if XVal is a number, *list* if XVal is a list

Computes a probability at XVal, the number of the trial on which the first success occurs, for the discrete geometric distribution with the specified probability of success p.

Get Hub Menu

Get[promptString,]var[, statusVar]

Get[promptString,] func(arg1, ...argn)
[, statusVar]

Programming command: Retrieves a value from a connected TI-Innovator^{\mathbf{M}} Hub and assigns the value to variable var.

The value must be requested:

- In advance, through a Send "READ ..." command.
 - or -
- By embedding a "READ ..." request as the optional promptString argument. This method lets you use a single command to request the value and retrieve it.

Implicit simplification takes place. For example, a received string of "123" is interpreted as a numeric value. To preserve the string, use **GetStr** instead of **Get**.

If you include the optional argument *statusVar*, it is assigned a value based on the success of the operation. A value of zero means that no data was received.

Example: Request the current value of the hub's built-in light-level sensor. Use **Get** to retrieve the value and assign it to variable *lightval*.

Send "READ BRIGHTNESS"	Done
Get lightval	Done
lightval	0.347922

Embed the READ request within the **Get** command.

Get "READ BRIGHTNESS",lightval	Done
lightval 0.	.378441

Get Hub Menu

In the second syntax, the *func()* argument allows a programme to store the received string as a function definition. This syntax operates as if the programme executed the command:

Define func(arg1, ...argn) = received string

The programme can then use the defined function *func*().

Note: You can use the **Get** command within a user-defined programme but not within a function.

Note: See also **GetStr**, page 83 and **Send**, page 156.

getDenom()

 $getDenom(Expr1) \Rightarrow expression$

Transforms the argument into an expression having a reduced common denominator, and then returns its denominator.

$\frac{\text{getDenom}\left(\frac{x+2}{y-3}\right)}{\text{getDenom}\left(\frac{2}{7}\right)} \qquad y-3$ $\frac{1}{\sqrt{2+y}} \qquad x\cdot y$

getKey()

getKey([0|1]) ⇒ returnString

Description:getKey() - allows a TI-Basic programme to get keyboard input - handheld, desktop and emulator on desktop.

Example:

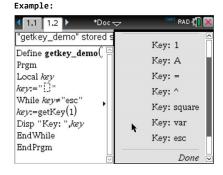
- keypressed := getKey() will return a key or an empty string if no key has been pressed. This call will return immediately.
- keypressed := getKey(1) will wait till a key is pressed. This call will pause execution of the programme till a key is pressed.

Catalogue > 👰

Catalogue > 🗐

getKey()

getDenom



Handling of key presses:

Handheld Device/Emulator Key	Desktop	Return Value
Esc	Esc	"esc"
Touchpad - Top click	n/a	"up"
On	n/a	"home"
Scratchapps	n/a	"scratchpad"
Touchpad - Left click	n/a	"left"
Touchpad - Centre click	n/a	"centre"
Touchpad - Right click	n/a	"right"
Doc	n/a	"doc"
		W. J. II
Tab	Tab	"tab"
Touchpad - Bottom click	Down Arrow	"down"
Menu	n/a	"menu"
Ctrl	Ctrl	no return
Shift	Shift	no return
Var	n/a	"var"
Del	n/a	"del"
=	=	"="
trig	n/a	"trig"
0 to 9	0-9	"0" "9"
Templates	n/a	"template"
Catalogue	n/a	"cat"
٨	^	"^"
X^2	n/a	"square"
	<u> </u>	square
/ (division key)	*	/ "*"
* (multiply key)		

Handheld Device/Emulator Key	Desktop	Return Value	
e^x	n/a	"exp"	
10^x	n/a	"10power"	
+	+	"+"	
-	-	"_"	
(("("	
))	")"	
		"."	
(-)	n/a	"-" (negate sign)	
Enter	Enter	"enter"	
ee	n/a	"E" (scientific notation E)	
a - z	a-z	alpha = letter pressed (lower case)	
		("a" - "z")	
shift a-z	shift a-z	alpha = letter pressed	
		"A" - "Z"	
		Note: ctrl-shift works to lock caps	
?!	n/a	"?!"	
pi	n/a	"pi"	
Flag	n/a	no return	
,	,	" "	
Return	n/a	"return"	
Space	Space	" " (space)	
Inaccessible	Special Character Keys like @,!,^, etc.	The character is returned	
n/a	Function Keys	No returned character	
n/a	Special desktop control keys	No returned character	
Inaccessible	Other desktop keys that are not available on the calculator while getkey() is	Same character you get in Notes (not in a maths box)	

Handheld Device/Emulator Key	Desktop	Return Value
	waiting for a keystroke. ({, },;, :,)	

Note: It is important to note that the presence of getKey() in a programme changes how certain events are handled by the system. Some of these are described below.

Terminate programme and Handle event - Exactly as if the user were to break out of programme by pressing the **ON** key

"Support" below means - System works as expected - programme continues to run.

Event	Device	Desktop - TI-Nspire™ Student Software
Quick Poll	Terminate programme, handle event	Same as the handheld (TI- Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-only)
Remote file mgmt	Terminate programme, handle event	Same as the handheld. (TI-Nspire™ Student
(Incl. sending 'Exit Press 2 Test' file from another handheld or desktop- handheld)		Software, TI-Nspire™ Navigator™ NC Teacher Software-only)
End Class	Terminate programme,	Support
	handle event	(TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-only)

Event	Device	Desktop - TI-Nspire™ All Versions
TI-Innovator™ Hub connect/disconnect	Support - Can successfully issue commands to the TI-Innovator™ Hub. After you exit the programme the TI-Innovator™ Hub is still working with the handheld.	Same as the handheld

getLangInfo()		Catalogue > 🗐
getLangInfo()⇒string	getLangInfo()	"en"

Returns a string that corresponds to the short name of the currently active language. You can, for example, use it in a programme or function to determine the current language.

English = "en"
Danish = "da"
German = "de"
Finnish = "fi"
French = "fr"
Italian = "it"
Dutch = "nl"
Belgian Dutch = "nl_BE"
Norwegian = "no"
Portuguese = "pt"
Spanish = "es"
Swedish = "sv"

getLockInfo()

Catalogue > 🗐

$getLockInfo(Var) \Rightarrow value$

Returns the current locked/unlocked state of variable Var.

value =0: Var is unlocked or does not exist.

value **=1**: *Var* is locked and cannot be modified or deleted.

See Lock, page 105, and unLock, page 195.

a:=65	65
Lock a	Done
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
a:=75	75
DelVar a	Done

getMode()

Catalog > 🔯

getMode(*ModeNameInteger***)**⇒*value*

 $getMode(0) \Rightarrow list$

getMode(*ModeNameInteger*) returns a value representing the current setting of the *ModeNameInteger* mode.

getMode(0) returns a list containing number pairs. Each pair consists of a mode integer and a setting integer.

getMode(0) {1,7,2,1,3,1,4,1,5,1,6,1	,7,1,8,1}
getMode(1)	7
getMode(8)	1

getMode()

For a listing of the modes and their settings, refer to the table below.

If you save the settings with getMode(0) → var, you can use setMode(var) in a function or programme to temporarily restore the settings within the execution of the function or programme only. See setMode (), page 159.

Mode Name	Mode Integer	Setting Integers
Display Digits	1	1=Float, 2=Float1, 3=Float2, 4=Float3, 5=Float4, 6=Float5, 7=Float6, 8=Float7, 9=Float8, 10=Float9, 11=Float10, 12=Float11, 13=Float12, 14=Fix0, 15=Fix1, 16=Fix2, 17=Fix3, 18=Fix4, 19=Fix5, 20=Fix6, 21=Fix7, 22=Fix8, 23=Fix9, 24=Fix10, 25=Fix11, 26=Fix12
Angle	2	1=Radian, 2=Degree, 3=Gradian
Exponential Format	3	1=Normal, 2=Scientific, 3=Engineering
Real or Complex	4	1=Real, 2=Rectangular, 3=Polar
Auto or Approx.	5	1=Auto, 2=Approximate, 3=Exact
Vector Format	6	1=Rectangular, 2=Cylindrical, 3=Spherical
Base	7	1=Decimal, 2=Hex, 3=Binary
Unit system	8	1=SI, 2=Eng/US

getNum()		Catalogue > 🕎
$getNum(Expr1) \Rightarrow expression$	$\frac{1}{\text{getNum}\left(\frac{x+2}{y-3}\right)}$	x+2
Transforms the argument into an expression having a reduced common denominator, and then returns its	$\frac{\sqrt{y-3}}{\text{getNum}\left(\frac{2}{7}\right)}$	2
numerator.	$getNum\left(\frac{1}{x} + \frac{1}{y}\right)$	<i>x</i> + <i>y</i>

GetStr	Hub Menu
<pre>GetStr[promptString,] var[, statusVar]</pre>	For examples, see Get .

GetStr Hub Menu

GetStr[promptString,] func(arg1, ...argn) [, statusVar]

Programming command: Operates identically to the **Get** command, except that the retrieved value is always interpreted as a string. By contrast, the **Get** command interprets the response as an expression unless it is enclosed in quotation marks ("").

Note: See also Get, page 77 and Send, page 156.

getType() Catalogue > 23 $getType(var) \Rightarrow string$ $\{1,2,3\} \rightarrow temp$ {1,2,3} Returns a string that indicates the data getType(temp) "LIST" type of variable var. $3 \cdot i \rightarrow temp$ 3. i getType(temp) "EXPR" If var has not been defined, returns the string "NONE". DelVar temp Done getType(temp) "NONE

getVarInfo() Catalogue > [1][3]

 $getVarInfo() \Rightarrow matrix \text{ or } string$

getVarInfo(*LibNameString***)**⇒*matrix* or *string*

getVarInfo() returns a matrix of information (variable name, type, library accessibility and locked/unlocked state) for all variables and library objects defined in the current problem.

If no variables are defined, getVarInfo() returns the string "NONE".

getVarInfo(*LibNameString*) returns a matrix of information for all library objects defined in library *LibNameString*. *LibNameString* must be a string (text enclosed in quotation marks) or a string variable.

If the library *LibNameString* does not exist, an error occurs.

getVarInfo()			"NOI	VE"
Define x=5			D	one
Lock x			D	one
Define LibPriv y	·={ 1.	,2,3}	D	one
Define LibPub $z(x)=3\cdot x^2-x$		D	one	
getVarInfo()	[x	"NUM"	"[]"	1
	У	"LIST"	"LibPriv "	0
	Z	"FUNC"	"LibPub "	0]

getVarInfo(tmp3)

"Error: Argument must be a string"

getVarInfo("tmp3")

[volcyl2 "NONE" "LibPub " 0]

getVarInfo()

Note the example to the left, in which the result of **getVarInfo()** is assigned to variable vs. Attempting to display row 2 or row 3 of vs returns an "Invalid list or matrix" error because at least one of elements in those rows (variable b, for example) revaluates to a matrix.

This error could also occur when using *Ans* to reevaluate a **getVarInfo()** result.

The system gives the above error because the current version of the software does not support a generalised matrix structure where an element of a matrix can be either a matrix or a list.

a:=1				1
$b := \begin{bmatrix} 1 & 2 \end{bmatrix}$			[1	2]
c:=[1 3 7]			[1 3	7]
vs:=getVarInfo()	a	"NUM"	"[]"	0]
	b	"MAT"	"[]"	0
	$\lfloor c$	"MAT"	"[]"	0]
vs[1]	[1	"NUM"	"[]"	0]
vs[1,1]				1
vs[2]	'Error: Iı	nvalid list	or matr	ix"
vs[2,1]			1	2

Catalogue > 🕮

Goto Catalogue > 🕎

Goto label Name

Transfers control to the label *labelName*.

labelName must be defined in the same function using a **Lbl** instruction.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Done
55

▶Grad Catalogue > 🗓 3

 $Expr1
ightharpoonup Grad \Rightarrow expression$

Converts *Expr1* to gradian angle measure.

Note: You can insert this operator from the computer keyboard by typing @>Grad.

In Degree angle mode:

(1.5)▶Grad (1.66667)⁹

In Radian angle mode:

(1.5)▶Grad (95.493)⁹

identity()		Catalogue > 🕡
$identity(Integer) \Rightarrow matrix$	identity(4)	1 0 0 0
Returns the identity matrix with a dimension of $Integer$.		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Integer must be a positive integer		[0 0 0 1]

Returns the identity matrix with a dimension of <i>Integer</i> .	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Integer must be a positive integer.	[0 0 0 1]
If	Catalogue > 🗐
If BooleanExpr Statement	Define $g(x)$ =Func Done If x <0 Then
If BooleanExpr Then Block EndIf	Return x^2 EndIf EndFunc
If $BooleanExpr$ evaluates to true, executes the single statement $Statement$ or the block of statements $Block$ before continuing execution.	g(-2) 4
If <i>BooleanExpr</i> evaluates to false, continues execution without executing the statement or block of statements.	
Block can be either a single statement or a sequence of statements separated with the ":" character.	
Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.	
If BooleanExpr Then Block1 Else	Define $g(x)$ =Func Done If x <0 Then Return $\neg x$
Block2	Else
If BooleanExpr evaluates to true, executes Block1 and then skips Block2.	Return x EndIf EndFunc
If BooleanExpr evaluates to false, skips Block1 but executes Block2.	$\frac{g(12)}{g(-12)}$ 12 12

	return x	
	Else	
	Return x	
	EndIf	
	EndFunc	
g(12)		12
g(-12)		12

Block1 and *Block2* can be a single statement.

 $\begin{array}{c} \textbf{If } BooleanExpr1 \textbf{ Then} \\ Block1 \\ \textbf{ElseIf } BooleanExpr2 \textbf{ Then} \\ Block2 \end{array}$

 $\begin{array}{c} \textbf{Elself} \ Boolean ExprN \ \textbf{Then} \\ Block N \end{array}$

EndIf

Allows for branching. If BooleanExpr1 evaluates to true, executes Block1. If BooleanExpr1 evaluates to false, evaluates BooleanExpr2, and so on.

Define $g(x)$ =Func	
If $x < -5$ Then	
Return 5	
ElseIf $x > -5$ and $x < 0$ Then	
Return ⁻ <i>x</i>	
ElseIf $x \ge 0$ and $x \ne 10$ Then	
Return x	
ElseIf $x=10$ Then	
Return 3	
EndIf	
EndFunc	
D)one

	Done
g(-4)	4
g(10)	3

ifFn()

ifFn(BooleanExpr,Value_If_true [,Value_If_false [,Value_If_unknown]]) ⇒ expression, list, or matrix

Evaluates the boolean expression BooleanExpr (or each element from BooleanExpr) and produces a result based on the following rules:

- BooleanExpr can test a single value, a list, or a matrix.
- If an element of *BooleanExpr* evaluates to true, returns the corresponding element from *Value If true*.
- If an element of BooleanExpr evaluates to false, returns the corresponding element from Value_If_false. If you omit Value_If_false, returns undef.
- If an element of BooleanExpr is neither true nor false, returns the corresponding element Value_If_unknown. If you omit Value_If_unknown, returns undef.
- If the second, third, or fourth argument of the ifFn() function is a single expression, the Boolean test is applied to every position in BooleanExpr.

ifFn({1,2,3}<2.5,{5,6,7},{8,9,10}) {5,6,10}

Catalogue > 🕮

Test value of **1** is less than 2.5, so its corresponding

Value_If_True element of **5** is copied to the result list.

Test value of **2** is less than 2.5, so its corresponding

Value_If_True element of 6 is copied to
the result list.

Test value of **3** is not less than 2.5, so its corresponding $Value_If_False$ element of **10** is copied to the result list.

$$ifFn({1,2,3}<2.5,4,{8,9,10})$$
 {4,4,10}

Value_If_true is a single value and corresponds to any selected position.

ifFn()

Catalogue > 🗐

Note: If the simplified *BooleanExpr* statement involves a list or matrix, all other list or matrix arguments must have the same dimension(s), and the result will have the same dimension(s).

$$\overline{\text{ifFn}(\{1,2,3\}<2.5,\{5,6,7\})}$$
 {5,6,undef}

Value_If_false is not specified. Undef is used.

$$\overline{ifFn({2,"a"}<2.5,{6,7},{9,10},"err") \atop {6,"err"}}$$

One element selected from $Value_If_true$.
One element selected from $Value_If_unknown$.

imag() Catalogue > [2]

 $imag(Expr1) \Rightarrow expression$

Returns the imaginary part of the argument.

Note: All undefined variables are treated as real variables. See also real(), page 144

 $imag(List1) \Rightarrow list$

Returns a list of the imaginary parts of the elements.

 $imag(Matrix 1) \Rightarrow matrix$

Returns a matrix of the imaginary parts of the elements.

$imag(1+2\cdot i)$	2
imag(z)	0
$imag(x+i\cdot y)$	у

$$imag(\{-3,4-i,i\})$$
 $\{0,-1,1\}$

$$\operatorname{imag}\begin{bmatrix} a & b \\ i \cdot c & i \cdot d \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$$

impDif() Catalogue > [3]

impDif(Equation, Var, dependVar[,Ord]**)** ⇒ expression

where the order Ord defaults to 1.

Computes the implicit derivative for equations in which one variable is defined implicitly in terms of another.

impDif
$$\left(x^2+y^2=100,x,y\right)$$
 $\frac{-x}{y}$

Indirection

See #(), page 223.

inString() Catalogue > 🕄

inString(srcString, subString[, Start]) ⇒
integer

inString("Hello there", "the") 7
inString("ABCEFG", "D") 0

Returns the character position in string *srcString* at which the first occurrence of string *subString* begins.

Start, if included, specifies the character position within *srcString* where the search begins. Default = 1 (the first character of *srcString*).

If srcString does not contain subString or Start is > the length of srcString, returns zero.

int()	Catalogue > 👰

 $int(Expr) \Rightarrow integer$

 $int(List1) \Rightarrow list$ $int(Matrix1) \Rightarrow matrix$

Returns the greatest integer that is less than or equal to the argument. This function is identical to **floor()**.

The argument can be a real or a complex number.

For a list or matrix, returns the greatest integer of each of the elements.

int(-2.5) -3. int([-1.234 0 0.37]) [-2. 0 0.]

intDiv() Catalogue > [2]

intDiv(Number1, Number2) \Rightarrow integer intDiv(List1, List2) \Rightarrow list intDiv(Matrix1, Matrix2) \Rightarrow matrix

Returns the signed integer part of $(Number1 \div Number2)$.

For lists and matrices, returns the signed integer part of (argument $1 \div$ argument 2) for each element pair.

intDiv(-7,2)	-3
intDiv(4,5)	0
intDiv({12,-14,-16},{5,4,-3})	{2,-3,5}

interpolate ()

Catalogue > 🗐

interpolate(xValue, xList, yList, yPrimeList) $\Rightarrow list$

This function does the following:

Given xList, yList= $\mathbf{f}(xList)$, and yPrimeList= $\mathbf{f}'(xList)$ for some unknown function \mathbf{f} , a cubic interpolant is used to approximate the function \mathbf{f} at xValue. It is assumed that xList is a list of monotonically increasing or decreasing numbers, but this function may return a value even when it is not. This function walks through xList looking for an interval [xList[i], xList[i+1]] that contains xValue. If it finds such an interval, it returns an interpolated value for $\mathbf{f}(xValue)$; otherwise, it returns \mathbf{undef} .

xList, yList, and yPrimeList must be of equal dimension ≥ 2 and contain expressions that simplify to numbers.

xValue can be an undefined variable, a number, or a list of numbers.

Differential equation: $y'=-3 \cdot y+6 \cdot t+5$ and y(0)=5

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

Use the interpolate() function to calculate the function values for the xvaluelist:

invχ2()

Catalogue > 🗐

invχ²(Area,df)

invChi2(Area,df)

Computes the Inverse cumulative χ^2 (chi-square) probability function specified by degree of freedom, $d\!f$ for a given Area under the curve.

invF()

Catalogue > 🗐

invF(Area,dfNumer,dfDenom)

invF(Area,dfNumer,dfDenom)

computes the Inverse cumulative F distribution function specified by dfNumer and dfDenom for a given Area under the curve.

invBinom() Catalogue > [[]]

invBinom

(CumulativeProb,NumTrials,Prob, OutputForm)⇒ scalar or matrix

Given the number of trials (NumTrials) and the probability of success of each trial (Prob), this function returns the minimum number of successes, k, such that the cumulative probability of k successes is greater than or equal to the given cumulative probability (CumulativeProb).

OutputForm=**0**, displays result as a scalar (default).

OutputForm=1, displays result as a matrix.

Example: Mary and Kevin are playing a dice game. Mary has to guess the maximum number of times 6 shows up in 30 rolls. If the number 6 shows up that many times or less, Mary wins. Furthermore, the smaller the number that she guesses, the greater her winnings. What is the smallest number Mary can guess if she wants the probability of winning to be greater than 77%?

invBinomN()

invBinomN(CumulativeProb,Prob, NumSuccess,OutputForm)⇒ scalar or matrix

Given the probability of success of each trial (Prob), and the number of successes (NumSuccess), this function returns the minimum number of trials, N, such that the cumulative probability of x successes is less than or equal to the given cumulative probability (CumulativeProb).

OutputForm=**0**, displays result as a scalar (default).

OutputForm=1, displays result as a matrix.

Catalogue > 📳

Example: Monique is practising goal shots for netball. She knows from experience that her chance of making any one shot is 70%. She plans to practise until she scores 50 goals. How many shots must she attempt to ensure that the probability of making at least 50 goals is more than 0.99?

invNorm()

Catalogue > 🕎

 $invNorm(Area[,\mu[,\sigma]])$

Computes the inverse cumulative normal distribution function for a given Area under the normal distribution curve specified by μ and σ .

invt() Catalogue > [3]

invt(Area,df)

Computes the inverse cumulative student-t probability function specified by degree of freedom, *df* for a given *Area* under the curve.

iPart() Catalogue > [3]

iPart(Number) ⇒ integer **iPart**(List1) ⇒ list**iPart**(Matrix1) ⇒ matrix

iPart(-1.234) -1. iPart $\left\{\frac{3}{2}$,-2.3,7.003 $\right\}$ $\left\{1,-2.,7.\right\}$

Returns the integer part of the argument.

For lists and matrices, returns the integer part of each element.

The argument can be a real or a complex number.

irr() Catalogue > 23

 $irr(CF0,CFList\ [,CFFreq]) \Rightarrow value$

Financial function that calculates internal rate of return of an investment.

CF0 is the initial cash flow at time 0; it must be a real number.

CFList is a list of cash flow amounts after the initial cash flow CFO.

CFFreq is an optional list in which each element specifies the frequency of occurrence for a grouped (consecutive) cash flow amount, which is the corresponding element of CFList. The default is 1; if you enter values, they must be positive integers < 10,000.

Note: See also mirr(), page 113.

list1:={6000,-8000,2000,-3000}		
{ (5000,-8000,2000,-3000}	
list2:={2,2,2,1}	{2,2,2,1}	
irr(5000,list1,list2)	-4.64484	

isPrime()

Catalogue > 😰

isPrime(Number**)** \Rightarrow Boolean constant expression

Returns true or false to indicate if number is a whole number ≥ 2 that is evenly divisible only by itself and 1.

If Number exceeds about 306 digits and has no factors \leq 1021, isPrime(Number) displays an error message.

If you merely want to determine if *Number* is prime, use **isPrime()** instead of **factor()**. It is much faster, particularly if *Number* is not prime and has a second-largest factor that exceeds about five digits.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

isPrime(5)	true
isPrime(6)	false

Function to find the next prime after a specified number:

Define nextprim	(n)=Func	Done
	Loop	
	$n+1 \rightarrow n$	
	If $isPrime(n)$	
	Return n	
	EndLoop	
	EndFunc	
nextprim(7)		11

isVoid()

Catalogue > 😰

isVoid(Var) ⇒ Boolean constant expression isVoid(Expr) ⇒ Boolean constant expression

isVoid(List) ⇒ list of Boolean constant expressions

Returns true or false to indicate if the argument is a void data type.

For more information on void elements, see page 248.

Lbl Catalogue > 2

Lbl lahelName

Defines a label with the name labelName within a function.

You can use a **Goto** labelName instruction to transfer control to the instruction immediately following the label.

labelName must meet the same naming requirements as a variable name.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define g()	=Func	Done
	Local temp,i	
	$0 \rightarrow temp$	
	$1 \rightarrow i$	
	Lbl top	
	$temp+i \rightarrow temp$	
	If $i < 10$ Then	
	$i+1 \rightarrow i$	
	Goto top	
	EndIf	
	Return temp	
	EndFunc	
g()		55

lcm() Catalogue > 🕮

lcm(*Number1***,** *Number2***)**⇒*expression*

 $lcm(List1, List2) \Rightarrow list$

 $lcm(Matrix 1, Matrix 2) \Rightarrow matrix$

Returns the least common multiple of the two arguments. The Icm of two fractions is the Icm of their numerators divided by the gcd of their denominators. The Icm of fractional floating-point numbers is their product.

For two lists or matrices, returns the least common multiples of the corresponding elements.

lcm(6,9)	18
$lcm\left\{\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right\}$	$\left\{\frac{2}{3},14,80\right\}$

left() Catalogue > 23

left(sourceString[, Num])⇒string

Returns the leftmost Num characters contained in character string sourceString.

If you omit *Num*, returns all of sourceString.

left() Catalogue > [1]3

$$left(List1[, Num]) \Rightarrow list$$

left(
$$\{1,3,-2,4\},3$$
) $\{1,3,-2\}$

Returns the leftmost Num elements contained in List 1.

If you omit Num, returns all of List1.

Returns the left-hand side of an equation or inequality.

$$left(x<3) x$$

libShortcut()

Catalogue > 😰

libShortcut(LibNameString, ShortcutNameString [, LibPrivFlag])⇒list of variables

Creates a variable group in the current problem that contains references to all the objects in the specified library document *libNameString*. Also adds the group members to the Variables menu. You can then refer to each object using its *ShortcutNameString*.

Set *LibPrivFlag*=**0** to exclude private library objects (default)

Set *LibPrivFlag*=1 to include private library objects

To copy a variable group, see **CopyVar**, page 29

To delete a variable group, see **DelVar**, page 48.

This example assumes a properly stored and refreshed library document named **linalg2** that contains objects defined as *clearmat*, gauss1 and gauss2.

 $\left\{ la. clearmat, la. gauss 1, la. gauss 2 \right\}$

limit() or lim()

limit(*Expr1***,** *Var***,** *Point* [,*Direction*]**)**⇒*expression*

limit(List1, Var, Point [, Direction])⇒list

limit(Matrix1, Var, Point [, Direction])⇒matrix

Returns the limit requested.

Note: See also Limit template, page 6.

Direction: negative=from left, positive=from right, otherwise=both. (If omitted, *Direction* defaults to both.)

Limits at positive ∞ and at negative ∞ are always converted to one-sided limits from the finite side.

Depending on the circumstances, **limit()** returns itself or undef when it cannot determine a unique limit. This does not necessarily mean that a unique limit does not exist. undef means that the result is either an unknown number with finite or infinite magnitude, or it is the entire set of such numbers.

limit() uses methods such as L'Hopital's rule, so there are unique limits that it cannot determine. If Expr1 contains undefined variables other than Var, you might have to constrain them to obtain a more concise result.

Limits can be very sensitive to rounding error. When possible, avoid the Approximate setting of the **Auto or Approximate** mode and approximate numbers when computing limits. Otherwise, limits that should be zero or have infinite magnitude probably will not, and limits that should have finite non-zero magnitude might not.

		_
$\lim_{x\to 5} (2\cdot x+3)$	1.	3
$\lim_{x \to 0^+} \left(\frac{1}{x} \right)$	0	0
$\lim_{x \to 0} \left(\frac{\sin(x)}{x} \right)$		1
$\lim_{h\to 0} \left(\frac{\sin(x+h) - \sin(x)}{h} \right)$	$\cos(x)$;)
$\lim_{n\to\infty} \left(\left(1 + \frac{1}{n} \right)^n \right)$	•	e

Catalogue > 🕮

$\lim (a^x)$	undef
$x \rightarrow \infty$	
$\lim_{x\to\infty} (a^x) a>1$	∞
$\frac{x \to \infty}{x + x}$	0
$\lim_{x\to\infty} (a^x) a>0 \text{ and } a<1$	Ü

LinRegBx

Catalogue > 🗐

LinRegBx X, Y[,[Freq][,Category,Include]]

LinRegBx

Computes the linear regressiony = $a+b \cdot xon$ lists Xand Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression Equation: a+b·x
stat.a, stat.b	Regression coefficients
stat.r ²	Coefficient of determination
stat.r	Correlation coefficient
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat.XReg and stat.YReg

Catalogue > 23 LinRegMx

LinRegMx X, Y[,[Freq][,Category,Include]]

LinRegMx

Computes the linear regression $y = m \cdot x + b$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for *Include*.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers \geq 0.

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression Equation: y = m ·x+b
stat.m, stat.b	Regression coefficients
stat.r ²	Coefficient of determination
stat.r	Correlation coefficient
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat.XReg and stat.YReg

LinRegtIntervals

Catalogue > 🗐

LinRegtIntervals X,Y[,F[,0[,CLev]]]

Catalogue > 23

LinRegtIntervals

For Slope. Computes a level C confidence interval for the slope.

LinRegtIntervals X,Y[F,1,Xval,CLev]

For Response. Computes a predicted y-value, a level C prediction interval for a single observation and a level C confidence interval for the mean response.

A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension.

X and Y are lists of independent and dependent variables.

F is an optional list of frequency values. Each element in *F* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression Equation: a+b·x
stat.a, stat.b	Regression coefficients
stat.df	Degrees of freedom
stat.r ²	Coefficient of determination
stat.r	Correlation coefficient
stat.Resid	Residuals from the regression

For Slope type only

Output variable	Description
[stat.CLower, stat.CUpper]	Confidence interval for the slope
stat.ME	Confidence interval margin of error
stat.SESlope	Standard error of slope
stat.s	Standard error about the line

For Response type only

Output variable	Description
[stat.CLower, stat.CUpper]	Confidence interval for the mean response
stat.ME	Confidence interval margin of error
stat.SE	Standard error of mean response
[stat.LowerPred,	Prediction interval for a single observation
stat.UpperPred]	
stat.MEPred	Prediction interval margin of error
stat.SEPred	Standard error for prediction
stat.ŷ	a + b · XVal

LinRegtTest

Catalogue > 🗐

LinRegtTest *X***,** *Y*[**,** *Freq*[**,** *Hypoth*]]

Computes a linear regression on the X and Y lists and a t test on the value of slope β and the correlation coefficient ρ for the equation y= α + βx . It tests the null hypothesis H_0 : β =0 (equivalently, ρ =0) against one of three alternative hypotheses.

All the lists must have equal dimension.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Hypoth is an optional value specifying one of three alternative hypotheses against which the null hypothesis (H_0 : β = ρ =0) will be tested.

For H_a : $\beta \neq 0$ and $\rho \neq 0$ (default), set Hypoth=0

For H_a : β <0 and ρ <0, set Hypoth<0

For H_a : β >0 and ρ >0, set Hypoth>0

A summary of results is stored in the *stat.results* variable (page 174).

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: $a + b \cdot x$
stat.t	t-Statistic for significance test
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.df	Degrees of freedom
stat.a, stat.b	Regression coefficients
stat.s	Standard error about the line
stat.SESlope	Standard error of slope
stat.r ²	Coefficient of determination
stat.r	Correlation coefficient
stat.Resid	Residuals from the regression

Catalogue > 23 linSolve()

linSolve(SystemOfLinearEgns, Var1, Var2, ...) $\Rightarrow list$

linSolve(LinearEqn1 and LinearEqn2 and ..., Var1, Var2, ...)⇒list

linSolve({LinearEqn1, LinearEqn2, ...}, $Var1, Var2, ... \implies \hat{l}ist$

linSolve(SystemOfLinearEqns, {Var1, Var2,...}) $\Rightarrow list$

linSolve(LinearEqn1 and LinearEqn2 and ..., {Var1, Var2, ...})⇒list

linSolve({LinearEqn1, LinearEgn2, ...}, $\{Var1, Var2, ...\}\} \Rightarrow list$

Returns a list of solutions for the variables Var1, Var2, ...

The first argument must evaluate to a system of linear equations or a single linear equation. Otherwise, an argument error occurs.

For example, evaluating linSolve(x=1 and x=2,x) produces an "Argument Error" result.

$$\begin{aligned} & \text{linSolve} \left\{ \begin{bmatrix} 2 \cdot x + 4 \cdot y = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{bmatrix} \left\{ x_{i} y \right\} \right\} & \left\{ \frac{37}{26}, \frac{1}{26} \right\} \\ & \text{linSolve} \left\{ \begin{bmatrix} 2 \cdot x = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{bmatrix} \left\{ x_{i} y \right\} \right\} & \left\{ \frac{3}{2}, \frac{1}{6} \right\} \\ & \text{linSolve} \left\{ \begin{bmatrix} apple + 4 \cdot pear = 23 \\ 5 \cdot apple - pear = 17 \end{bmatrix} \left\{ apple, pear \right\} \right\} \\ & \left\{ \frac{13}{3}, \frac{14}{3} \right\} \\ & \text{linSolve} \left\{ \begin{bmatrix} apple \cdot 4 + \frac{pear}{3} = 14 \\ -apple + pear = 6 \end{bmatrix} , \left\{ apple, pear \right\} \right\} \\ & \left\{ \frac{36}{13}, \frac{114}{13} \right\} \end{aligned}$$

Δ List()

list>mat()

Catalogue > 23

 $\Delta List(List1) \Rightarrow list$

ΔList({20,30,45,70})

{10,15,25

Note: You can insert this function from the keyboard by typing deltaList(...).

Returns a list containing the differences between consecutive elements in *List1*. Each element of *List1* is subtracted from the next element of List1. The resulting list is always one element shorter than the original *List1*.

Cata	logue	`	[a][2]

5 0

ln(10)

list mat(*List* [, elementsPerRow])⇒matrix

Returns a matrix filled row-by-row with the elements from List.

elementsPerRow, if included, specifies the number of elements per row. Default is the number of elements in *List* (one row).

If *List* does not fill the resulting matrix, zeroes are added.

Note: You can insert this function from the computer keyboard by typing list@>mat (...).

list▶mat({1,2,3})	[1 2 3]
list▶mat({1,2,3,4,5},2)	1 2
* * * * * * * * * * * * * * * * * * * *	2 4

Catalogue > [1] ▶ln

 $Expr \triangleright In \Rightarrow expression$

Causes the input *Expr* to be converted to an expression containing only natural logs (In).

Note: You can insert this operator from the computer keyboard by typing @>ln.

	cutalogue > 54.5
$\left(\log_{10}(x)\right) \triangleright \ln$	$\frac{\ln(x)}{\ln(10)}$

ex kevs

 $In(Expr1) \Rightarrow expression$

ln(2.)0.693147

 $In(List1) \Rightarrow list$

In()

Returns the natural logarithm of the argument.

If complex format mode is Real:

In()



For a list, returns the natural logarithms of the elements.

$$\ln(\{-3,1.2,5\})$$

"Error: Non-real calculation"

 $In(squareMatrix 1) \Rightarrow squareMatrix$

Returns the matrix natural logarithm of squareMatrix1. This is not the same as calculating the natural logarithm of each element. For information about the calculation method, refer to cos() on.

squareMatrix1 must be diagonalisable. The result always contains floating-point numbers.

If complex format mode is Rectangular:

$$\ln(\{-3,1.2,5\})$$
 $\{\ln(3)+\pi \cdot i,0.182322,\ln(5)\}$

In Radian angle mode and Rectangular complex format:

$$\ln \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

1.83145+1.73485·i 0.009193-1.49086 0.448761-0.725533·i 1.06491+0.623491 -0.266891-2.08316·*i* 1.12436+1.79018·

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

LnReg

Catalogue > 🕮

LnReg X, Y[, [Freq] [, Category, Include]]

Computes the logarithmic regression $y = a+b \cdot ln(x)$ on lists X and Y with frequency Freq. A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Frea is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.



For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: a+b ·ln(x)
stat.a, stat.b	Regression coefficients
stat.r ²	Coefficient of linear determination for transformed data
stat.r	Correlation coefficient for transformed data (ln(x), y)
stat.Resid	Residuals associated with the logarithmic model
stat.ResidTrans	Residuals associated with linear fit of transformed data
stat.XReg	List of data points in the modified $X\ List$ actually used in the regression based on restrictions of $Freq$, $Category\ List$ and $Include\ Categories$
stat.YReg	List of data points in the modified Y List actually used in the regression based on restrictions of Freq, Category List and Include Categories
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

Local Catalogue > 13

Local Var1[, Var2] [, Var3] ...

Declares the specified *vars* as local variables. Those variables exist only during evaluation of a function and are deleted when the function finishes execution.

Note: Local variables save memory because they only exist temporarily. Also, they do not disturb any existing global variable values. Local variables must be used for **For** loops and for temporarily saving values in a multi-line function since modifications on global variables are not allowed in a function.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

	$1 \rightarrow i$
	Loop
	If randInt $(1,6)$ =randInt $(1,6)$
	Goto end
	$i+1 \rightarrow i$
	EndLoop
	Lbl end
	Return i
	EndFunc
	Done
rollcount()	16
rollcount()	3

Local i

Define rollcount()=Func

Lock

Lock *Var1*[, *Var2*] [, *Var3*] ...

Lock Var.

Locks the specified variables or variable group. Locked variables cannot be modified or deleted.

You cannot lock or unlock the system variable Ans, and you cannot lock the system variable groups stat. or tvm.

Note: The Lock command clears the Undo/Redo history when applied to unlocked variables.

See unLock, page 195, andgetLockinfo(), page 82.

a:=65	65
Lock a	Done
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
a:=75	75
DelVar a	Done

Catalogue > 🕮

ctrl 10X keys

log()

 $log(Expr1[Expr2]) \Rightarrow expression$

 $log(List1[Expr2]) \Rightarrow list$

Returns the base-Expr2 logarithm of the first argument.

Note: See also Log template, page 2.

For a list, returns the base-Expr2 logarithm of the elements.

If the second argument is omitted, 10 is used as the base.

	out to keys
$\log_{10}(2.)$	0.30103
log (2.)	0.5
$\log_3(10) - \log_3(5)$	$\log_{3}(2)$

If complex format mode is Real:

$$\log_{10}(\{-3,1.2,5\}) \qquad \text{Error: } Non-\text{real } result$$

If complex format mode is Rectangular:

$$\frac{\log_{10}(\{-3,1.2,5\})}{\log_{10}(3)+1.36438 \cdot i,0.079181,\log_{10}(5)\}}$$

 $log(squareMatrix 1[,Expr]) \Rightarrow squareMatrix$

Returns the matrix base-*Expr* logarithm of squareMatrix1. This is not the same as calculating the base-*Expr* logarithm of each element. For information about the calculation method, refer to cos().

In Radian angle mode and Rectangular complex format:

log()

ctrl 10X keys

squareMatrixI must be diagonalisable. The result always contains floating-point numbers.

If the base argument is omitted, 10 is used as base.

$$\log_{10} \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

 $\begin{bmatrix} 0.795387 + 0.753438 \boldsymbol{\cdot} \mathbf{i} & 0.003993 - 0.6474 \boldsymbol{\cdot} \\ 0.194895 - 0.315095 \boldsymbol{\cdot} \mathbf{i} & 0.462485 + 0.2707 \boldsymbol{\cdot} \\ -0.115909 - 0.904706 \boldsymbol{\cdot} \mathbf{i} & 0.488304 + 0.77744 \boldsymbol{\cdot} \end{bmatrix}$

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

▶logbase

Catalogue > 🔯

 $Expr \triangleright logbase(Expr 1) \Rightarrow expression$

Causes the input Expression to be simplified to an expression using base Expr1.

Note: You can insert this operator from the computer keyboard by typing @>logbase (...).

$$\frac{\log_{3}(10) - \log_{5}(5) \triangleright \log \operatorname{base}(5)}{\log_{5}(3)}$$

Logistic

Catalogue > 👰

Logistic X, Y[, [Freq] [, Category, Include]]

Computes the logistic regressiony = $(c/(1+a \cdot e^{-bx}))$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for *Include*.

 \boldsymbol{X} and \boldsymbol{Y} are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

Logistic

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: c/(1+a ·e ^{-bx})
stat.a, stat.b, stat.c	Regression coefficients
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat.XReg and stat.YReg

LogisticD Catalogue > 🗐

LogisticD X, Y [, [Iterations], [Freq] [, Category, Include]]

Computes the logistic regression $y = (c/(1+a \cdot e^{-bx})+d)$ on lists X and Y with frequency Freq, using a specified number of *Iterations*. A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: c/(1+a ·e-bx)+d)
stat.a, stat.b, stat.c, stat.d	Regression coefficients
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

Loop	Catalogue > 🗐
------	---------------

Loop

Block

EndLoop

Repeatedly executes the statements in *Block*. Note that the loop will be executed endlessly, unless a Goto or Exit instruction is executed within *Block*.

Block is a sequence of statements separated with the ":" character.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define rollcount()=Func
Local i
$1 \rightarrow i$
Loop
If randInt $(1,6)$ =randInt $(1,6)$
Goto end
$i+1 \rightarrow i$
EndLoop
Lbl end
Return i
EndFunc
Done

	Done
rollcount()	16
rollcount()	3

0 0

1

1 0 0

0 1 0

0 0 1

LU Matrix, lMatrix, uMatrix, pMatrix [Tol]

Calculates the Doolittle LU (lower-upper) decomposition of a real or complex matrix. The lower triangular matrix is stored in *lMatrix*, the upper triangular matrix in uMatrix and the permutation matrix (which describes the row swaps done during the calculation) in *pMatrix*.

 $lMatrix \cdot uMatrix = pMatrix \cdot matrix$

Optionally, any matrix element is treated as zero if its absolute value is less than Tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, Tol is ignored.

- If you use ctri enter or set the Auto or Approximate mode to Approximate, computations are done using floatingpoint arithmetic.
- If Tol is omitted or not used, the default tolerance is calculated as: $5E-14 \cdot max(dim(Matrix)) \cdot rowNorm$ (Matrix)

The **LU** factorization algorithm uses partial pivoting with row interchanges.

$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 6 & 12 \\ 5 & 14 \end{bmatrix}$	18 31
[3 8 18]	18
LU m1,lower,upper,perm	Done
lower 1 0	0 (
$\left \frac{5}{6}\right $ 1	. 0
$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}$	- 1
<i>upper</i> [6 12	18
upper 6 12	

perm

$\begin{bmatrix} m & n \end{bmatrix} \rightarrow m1$		$\begin{bmatrix} m & n \end{bmatrix}$
$\begin{bmatrix} o & p \end{bmatrix}$		$\begin{bmatrix} o & p \end{bmatrix}$
LU m1,lower,upper,perm		Done
lower		1 0
		$\left[\frac{m}{o} 1\right]$
upper	0	p
	0	$n-\frac{m\cdot p}{o}$
perm		0 1
		11.0

M

mat list()

 $mat \ge list(Matrix) \Rightarrow list$

Returns a list filled with the elements in Matrix. The elements are copied from *Matrix* row by row.

Note: You can insert this function from the computer keyboard by typing mat@>list (...).

	Catalogue > 🏥
mat▶list([1 2 3])	{1,2,3}
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
4 5 6	[4 5 6]
mat▶list(<i>m1</i>)	$\{1,2,3,4,5,6\}$

max()

Catalogue > 💱

max(Expr1, Expr2)⇒expression

max(List1, List2)⇒list

 $\max(2.3,1.4)$ 2.3 $\max(\{1,2\},\{-4,3\})$ $\{1,3\}$

max(Matrix1, Matrix2)⇒matrix

Returns the maximum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the maximum value of each pair of corresponding elements.

 $max(List) \Rightarrow expression$

Returns the maximum element in *list*.

 $max(Matrix 1) \Rightarrow matrix$

Returns a row vector containing the maximum element of each column in *Matrix I*.

Empty (void) elements are ignored. For more information on empty elements, see page 248.

Note: See also fMax() and min().

$$\max(\{0,1,-7,1.3,0.5\})$$
 1.3

$$\max \begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 7 \end{bmatrix}$$

mean()

 $mean(List[,freqList]) \Rightarrow expression$

Returns the mean of the elements in List.

Each *freqList* element counts the number of consecutive occurrences of the corresponding element in *List*.

 $mean(Matrix 1[, freqMatrix]) \Rightarrow matrix$

Returns a row vector of the means of all the columns in *Matrix1*.

Each *freqMatrix* element counts the number of consecutive occurrences of the corresponding element in *Matrix1*.

Empty (void) elements are ignored. For more information on empty elements, see page 248.

Catalogue > 🗐

mean({0.2,0,1,-0.3,0.4})	0.26
mean({1,2,3},{3,2,1})	<u>5</u>
	3

In Rectangular vector format:

$ \text{mean} \begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix} $	[-0.133333
	$\begin{bmatrix} \frac{-2}{15} & \frac{5}{6} \end{bmatrix}$
	$\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

Catalogue > 23 median()

 $median(List[, freqList]) \Rightarrow expression$

median({0.2,0,1,-0.3,0.4}) 0.2

Returns the median of the elements in List.

Each freqList element counts the number of consecutive occurrences of the corresponding element in *List*.

 $median(Matrix 1[, freqMatrix]) \Rightarrow matrix$

Returns a row vector containing the medians of the columns in Matrix 1.

Each freqMatrix element counts the number of consecutive occurrences of the corresponding element in *Matrix 1*.

[0.4 - 0.3]0.2 0 median 1 -0.3 -0.5

Notes:

- All entries in the list or matrix must simplify to numbers.
- Empty (void) elements in the list or matrix are ignored. For more information on empty elements, see page 248.

MedMed Catalogue > 🕮

MedMed X,Y [, Freq] [, Category, Include]]

Computes the median-median liney = $(m \cdot x+b)$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Median-median line equation: m·x+b
stat.m, stat.b	Model coefficients
stat.Resid	Residuals from the median-median line
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y List actually used in the regression based on restrictions of Freq, Category List and Include Categories
stat.FreqReg	List of frequencies corresponding to stat.XReg and stat.YReg

mid() Catalogue > 23

mid(sourceString, Start[, Count])⇒string

Returns *Count* characters from character string *sourceString*, beginning with character number *Start*.

If *Count* is omitted or is greater than the dimension of *sourceString*, returns all characters from *sourceString*, beginning with character number *Start*.

Count must be ≥ 0 . If Count = 0, returns an empty string.

 $mid(sourceList, Start [, Count]) \Rightarrow list$

Returns *Count* elements from *sourceList*, beginning with element number *Start*.

If *Count* is omitted or is greater than the dimension of *sourceList*, returns all elements from *sourceList*, beginning with element number *Start*.

Count must be ≥ 0 . If Count = 0, returns an empty list.

 $mid(sourceStringList, Start[, Count]) \Rightarrow list$

mid("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	"[]"

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{[]}

$$\overline{ mid \big(\big\{ \text{"A","B","C","D"} \big\}, 2, 2 \big) } \\ \big\{ \text{"B","C"} \big\}$$

Returns Count strings from the list of strings sourceStringList, beginning with element number Start.

min() Catalogue > 🗐

 $min(Expr1, Expr2) \Rightarrow expression$

min(2.3,1.4) $\min(\{1,2\},\{-4,3\})$ -4.2

 $min(List1, List2) \Rightarrow list$

 $min(Matrix1, Matrix2) \Rightarrow matrix$

Returns the minimum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the minimum value of each pair of corresponding elements.

 $min(List) \Rightarrow expression$

Returns the minimum element of List.

 $min(Matrix 1) \Rightarrow matrix$

Returns a row vector containing the minimum element of each column in Matrix 1.

Note: See also fMin() and max().

min 1	-3	7]	[-4 -3	0.3]
\ [-4	0	0.3		

mirr() Catalogue > 🕮

mirr

(financeRate,reinvestRate,CF0,CFList [,CFFreq])

Financial function that returns the modified internal rate of return of an investment.

financeRate is the interest rate that you pay on the cash flow amounts.

reinvestRate is the interest rate at which the cash flows are reinvested.

CF0 is the initial cash flow at time 0: it must be a real number.

CFList is a list of cash flow amounts after the initial cash flow CFO.

list1:={6000,-8000,2000,-3000} {6000,-8000,2000,-3000} $list2:=\{2,2,2,1\}$ { 2.2.2.1 mirr(4.65,12,5000,list1,list2) 13,41608607 CFFreq is an optional list in which each element specifies the frequency of occurrence for a grouped (consecutive) cash flow amount, which is the corresponding element of CFList. The default is 1; if you enter values, they must be positive integers < 10,000.

Note: See also irr(), page 92.

mod()	Catal	ogue > 🕎
$mod(Expr1, Expr2) \Rightarrow expression$	mod(7,0)	7
$mod(List1, List2) \Rightarrow list$	mod(7,3) mod(-7,3)	1
mod(Matrix1, Matrix2)⇒matrix	mod(7,-3)	-2
Returns the first argument modulo the second argument as defined by the identities:	mod(-7,-3) $mod(\{12,-14,16\},\{9,7,-5\})$	$\frac{-1}{\left\{3,0,-4\right\}}$

mod(x,0) = x

mod(x,y) = x - y floor(x/y)

When the second argument is non-zero, the result is periodic in that argument. The result is either zero or has the same sign as the second argument.

If the arguments are two lists or two matrices, returns a list or matrix containing the modulo of each pair of corresponding elements.

Note: See also remain(), page 147

mRow()		Catalogue > 😰
$mRow(Expr, Matrix 1, Index) \Rightarrow matrix$	$\operatorname{mRow}\left(\frac{-1}{3},\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},2\right)$	1 2
Returns a copy of <i>Matrix1</i> with each element in row <i>Index</i> of <i>Matrix1</i> multiplied	\ 3 [3 4] \	$\left[\begin{array}{cc} -1 & \frac{-4}{3} \end{array}\right]$

by Expr.

mRowAdd()

Catalogue > [3]

mRowAdd(Expr, Matrix1, Index1, Index2) \Rightarrow matrix

mRowAdd[-3, 1 2 ,1,2] $mRowAdd[n, a b]_{1,1,2}$ h $a \cdot n + c \quad b \cdot n + d$

Returns a copy of *Matrix1* with each element in row *Index2* of *Matrix1* replaced with:

 $Expr \cdot row Index1 + row Index2$

Index2

Catalogue > 🕮 MultReg

MultReg Y, X1[,X2[,X3,...[,X10]]]

Calculates multiple linear regression of list Y on lists X1, X2, ..., X10. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression Equation: b0+b1 ·x1+b2 ·x2+
stat.b0, stat.b1,	Regression coefficients
stat.R ²	Coefficient of multiple determination
stat. ŷ List	ŷ List = b0+b1 ·x1+
stat.Resid	Residuals from the regression

MultRegIntervals

Catalogue > 🗐

MultRegintervals Y, X1[,X2[,X3,...[,X10]]], XValList[,CLevel]

Computes a predicted y-value, a level C prediction interval for a single observation, and a level C confidence interval for the mean response.

A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression Equation: b0+b1 ·x1+b2 ·x2+
stat.ŷ	A point estimate: $\hat{y} = b0 + b1 \cdot xl +$ for $XValList$
stat.dfError	Error degrees of freedom
stat.CLower, stat.CUpper	Confidence interval for a mean response
stat.ME	Confidence interval margin of error
stat.SE	Standard error of mean response
stat.LowerPred,	Prediction interval for a single observation
stat.UpperrPred	
stat.MEPred	Prediction interval margin of error
stat.SEPred	Standard error for prediction
stat.bList	List of regression coefficients, {b0,b1,b2,}
stat.Resid	Residuals from the regression

MultRegTests

Catalogue > 🗐

MultRegTests *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]]

Multiple linear regression test computes a multiple linear regression on the given data and provides the global F test statistic and t test statistics for the coefficients.

A summary of results is stored in the *stat.results* variable (page 174).

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Outputs

Output variable	Description	
stat.RegEqn	Regression Equation: b0+b1 ·x1+b2 ·x2+	
stat.F	GlobalF test statistic	

Output variable	Description
stat.PVal	P-value associated with global ${\cal F}$ statistic
stat.R ²	Coefficient of multiple determination
stat.AdjR ²	Adjusted coefficient of multiple determination
stat.s	Standard deviation of the error
stat.DW	Durbin-Watson statistic; used to determine whether first-order auto correlation is present in the model
stat.dfReg	Regression degrees of freedom
stat.SSReg	Regression sum of squares
stat.MSReg	Regression mean square
stat.dfError	Error degrees of freedom
stat.SSError	Error sum of squares
stat.MSError	Error mean square
stat.bList	{b0,b1,} List of coefficients
stat.tList	List of t statistics, one for each coefficient in the bList
stat.PList	List P-values for each t statistic
stat.SEList	List of standard errors for coefficients in bList
stat. ŷ List	\hat{y} List = b0+b1·x1+
stat.Resid	Residuals from the regression
stat.sResid	Standardized residuals; obtained by dividing a residual by its standard deviation
stat.CookDist	Cook's distance; measure of the influence of an observation based on the residual and leverage
stat.Leverage	Measure of how far the values of the independent variable are from their mean values

N

nand		ctrl = keys
BooleanExpr1nandBooleanExpr2 returns Boolean expression	x≥3 and x≥4	<i>x</i> ≥4
Boolean expression	$x \ge 3$ nand $x \ge 4$	x<4
BooleanList1nandBooleanList2 returns Boolean list		



BooleanMatrix1nandBooleanMatrix2 returns Boolean matrix

Returns the negation of a logical and operation on the two arguments. Returns true, false, or a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Integer1 nand*Integer2* ⇒ *integer*

Compares two real integers bit-by-bit using a nand operation. Internally, both integers are converted to signed, 64-bit binary numbers. When corresponding bits are compared, the result is 0 if both bits are 1; otherwise, the result is 1. The returned value represents the bit results, and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

3 and 4	0
3 nand 4	-1
{1,2,3} and {3,2,1}	{1,2,1}
{1,2,3} nand {3,2,1}	{-2,-3,-2}

Catalogue > 🗐

 $z \cdot (z-2) \cdot (z-1)$ 6

> $c! \cdot (z-c)!$ 1

10

nCr() $nCr(Expr1, Expr2) \Rightarrow expression$ nCr(z,3)

For integer Expr1 and Expr2 with $Expr1 \ge$ $Expr2 \ge 0$, nCr() is the number of

combinations of <i>Expr1</i> things taken <i>Expr2</i>	nCr(z,c)
at a time. (This is also known as a binomial	
coefficient.) Both arguments can be integers or symbolic expressions.	$\frac{Ans}{n\Pr(z,c)}$

Ans|z=5

$$nCr(Expr, 0) \Rightarrow 1$$

$$nCr(Expr, negInteger) \Rightarrow 0$$

$$nCr(Expr, posInteger) \Rightarrow Expr \cdot (Expr-1)...$$

nCr()

Catalogue > [3]

((Expr-nonInteger)! · nonInteger!)

 $nCr(List1, List2) \Rightarrow list$

$$nCr({5,4,3},{2,4,2})$$
 {10,1,3}

Returns a list of combinations based on the corresponding element pairs in the two lists. The arguments must be the same size list.

 $nCr(Matrix1, Matrix2) \Rightarrow matrix$

Returns a matrix of combinations based on the corresponding element pairs in the two matrices. The arguments must be the same size matrix.

nCr[6	5][2	2	1	15	10
$\backslash \! \lfloor 4$	3][2	2]	Į.	6	3

nDerivative()

Catalogue > 🗐

nDerivative(Expr1, Var = Value)[,Order])⇒value

nDerivative(Expr1,Var[,Order]) | Var=Value⇒value

Returns the numerical derivative calculated using auto differentiation methods.

When *Value* is specified, it overrides any prior variable assignment or any current "|" substitution for the variable.

Order of the derivative must be 1 or 2.

nDerivative($ x , x=1$)	1
nDerivative($ x ,x$) $ x=0$	undef
nDerivative $(\sqrt{x-1}, x) x=1$	undef

newList() Catalogue > 🕮 $newList(numElements) \Rightarrow list$ newList(4) 0,0,0,0

Returns a list with a dimension of mimElements. Fach element is zero.

dimension *numRows* by *numColumns*.

newMat()		Catalogue > 🗐
$\textbf{newMat}(numRows, numColumns) \Rightarrow matrix$	newMat(2,3)	0 0 0
Returns a matrix of zeroes with the		[0 0 0]

nfMax()

Catalogue > 🕄

 $nfMax(Expr, Var) \Rightarrow value$

 $nfMax(Expr, Var, lowBound) \Rightarrow value$

nfMax(Expr, Var, lowBound, upBound**)**⇒value

 $nfMax(Expr, Var) \mid lowBound \leq Var$ ≤upBounḋ⇒value

Returns a candidate numerical value of variable Var where the local maximum of Expr occurs.

If you supply *lowBound* and *upBound*, the function looks in the closed interval [lowBound,upBound] for the local maximum.

Note: See also fMax() and d().

nfMin()

Catalogue > 🔯

 $nfMin(Expr, Var) \Rightarrow value$

nfMin(*Expr*, *Var*, *lowBound***)**⇒*value*

nfMin(Expr, Var, lowBound, upBound**)**⇒value

nfMin(*Expr*, *Var***)** | *lowBound*≤*Var* ≤upBound⇒value

Returns a candidate numerical value of variable Var where the local minimum of Expr occurs.

If you supply *lowBound* and *upBound*, the function looks in the closed interval [lowBound,upBound] for the local minimum.

Note: See also fMin() and d().

nfMin
$$(x^2+2\cdot x+5,x)$$
 -1.
nfMin $(0.5\cdot x^3-x-2,x,-5,5)$ -5.

nInt()

Catalogue > 🕮

nint(Expr1, Var, Lower, *Upper*)⇒*expression*

1.49365 $nInt(e^{-x^2}, x, -1, 1)$

nInt()

If the integrand Expr1 contains no variable other than Var, and if Lower and Upper are constants, positive ∞ , or negative ∞ , then **nint()** returns an approximation of [(Expr1, Var, Lower, Upper). This approximation is a weighted average of some sample values of the integrand in the interval Lower<Var<Upper.

The goal is six significant digits. The adaptive algorithm terminates when it seems likely that the goal has been achieved, or when it seems unlikely that additional samples will yield a worthwhile improvement.

A warning is displayed ("Questionable accuracy") when it seems that the goal has not been achieved.

Nest nint() to do multiple numeric integration. Integration limits can depend on integration variables outside them.

Note: See also (1), page 218.

$$\frac{\operatorname{nInt}(\cos(x), x, \neg \pi, \pi+1. E-12)}{\int_{-\pi}^{\pi+10^{-12}} \cos(x) dx} - \sin\left(\frac{1}{10000000000000}\right)$$

$$\operatorname{nInt}\left(\operatorname{nInt}\left(\frac{\mathbf{e}^{-x \cdot y}}{\sqrt{x^2 - y^2}}, y, -x, x\right), x, 0, 1\right)$$
 3.30423

nom()

 $nom(effectiveRate, CpY) \Rightarrow value$

Financial function that converts the annual effective interest rate effectiveRate to a nominal rate, given CpY as the number of compounding periods per year.

effectiveRate must be a real number, and CpY must be a real number > 0.

Note: See also eff(), page 57.

Catalogue > 🕮

nom(5,90398,12) 5.75

nor		ctrl = keys
BooleanExpr1norBooleanExpr2 returns Boolean expression	<i>x</i> ≥3 or <i>x</i> ≥4	<i>x</i> ≥3

BooleanList Inor BooleanList 2 returns Boolean list

BooleanMatrix InorBooleanMatrix 2

 $x \ge 3$ nor $x \ge 4$ x < 3 returns Boolean matrix

Returns the negation of a logical **or** operation on the two arguments. Returns true, false, or a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

*Integer1***nor***Integer2*⇒*integer*

Compares two real integers bit-by-bit using a **nor** operation. Internally, both integers are converted to signed, 64-bit binary numbers. When corresponding bits are compared, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value represents the bit results and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

3 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

norm()		Catalogue > 🗊
$norm(Matrix) \Rightarrow expression$	$\operatorname{norm}\left[\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right]$	$\sqrt{a^2+b^2+c^2+d^2}$
norm(Vector)⇒expression Returns the Frobenius norm.	$\frac{\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)}{\operatorname{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)}$	√30
	norm([1 2])	$\sqrt{5}$
	$ \overbrace{\operatorname{norm}\begin{bmatrix}1\\2\end{bmatrix}} $	√5

normalLine()

 $normalLine(Expr1, Var, Point) \Rightarrow expression$

normalLine

 $(Expr1, Var=Point) \Rightarrow expression$

Returns the normal line to the curve represented by Expr1 at the point specified in Var=Point.

Make sure that the independent variable is not defined. For example, If f1(x):=5 and x:=3, then normalLine(f1(x),x,2) returns "false."

normalLine $(x^2,x,1)$	$\frac{3}{2}$ $\frac{x}{2}$
normalLine $((x-3)^2-4,x,3)$	<i>x</i> =3
normalLine $\left(x^{\frac{1}{3}}, x=0\right)$	0
normalLine $(\sqrt{ x }, x=0)$	undef

normCdf()

Catalogue > 🗐

Catalogue > 🕮

normCdf($lowBound,upBound[,\mu[,\sigma]]$) $\Rightarrow number$ if lowBound and upBound are numbers, list if lowBound and upBound are lists

Computes the normal distribution probability between lowBound and upBound for the specified µ (default=0) and σ (default=1).

For $P(X \le upBound)$, set $lowBound = -\infty$.

normPdf()

Catalogue > 23

normPdf($XVal[,\mu[,\sigma]]$) $\Rightarrow number$ if XVal is a number, *list* if *XVal* is a list

Computes the probability density function for the normal distribution at a specified XVal value for the specified μ and σ .

not

Catalogue > 23

not BooleanExpr⇒Boolean expression

Returns true, false, or a simplified form of the argument.

not *Integer1*⇒*integer*

not(2≥3) true not(x<2) $x \ge 2$ not not innocent innocent

In Hex base mode:

Important: Zero, not the letter O.

not

Catalogue > 👰

Returns the one's complement of a real integer. Internally, *Integer I* is converted to a signed, 64-bit binary number. The value of each bit is flipped (0 becomes 1 and vice versa) for the one's complement. Results are displayed according to the Base mode.

You can enter the integer in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, the integer is treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 64-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range. For more information, see >Base2, page 17.

not 0h7AC36 0hFFFFFFFFF	7853C9

In Bin base mode:

0b100101▶Base10	37
not 0b100101	
0b111111111111111111111111111111111111	11111111111
not 0b100101▶Base10	-38

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

Note: A binary entry can have up to 64 digits (not counting the 0b prefix). A hexadecimal entry can have up to 16 digits.

nPr()

$nPr(Expr1, Expr2) \Rightarrow expression$

For integer Expr1 and Expr2 with $Expr1 \ge Expr2 \ge 0$, nPr() is the number of permutations of Expr1 things taken Expr2 at a time. Both arguments can be integers or symbolic expressions.

$nPr(Expr, 0) \Rightarrow 1$

 $nPr(Expr, negInteger) \Rightarrow 1/((Expr+1) \cdot (Expr+2)...$

(expression-negInteger))

 $nPr(Expr, posInteger) \Rightarrow Expr \cdot (Expr-1)...$

(Expr-posInteger+1)

 $nPr(Expr, nonInteger) \Rightarrow Expr! / (Expr-nonInteger)!$

 $nPr(List1, List2) \Rightarrow list$

Returns a list of permutations based on the corresponding element pairs in the two lists. The arguments must be the same size list.

Catalogue > 🗐

nPr(z,3)	$z \cdot (z-2) \cdot (z-1)$
$\overline{Ans z=5}$	60
nPr(z,-3)	1
	$\overline{(z+1)\cdot(z+2)\cdot(z+3)}$
nPr(z,c)	z!
	$\overline{(z-c)!}$
$Ans \cdot nPr(z-c,-c)$	1

$$nPr({5,4,3},{2,4,2})$$
 {20,24,6}

nPr()

Catalogue > 🕮

12

 $nPr(Matrix 1, Matrix 2) \Rightarrow matrix$

Returns a matrix of permutations based on the corresponding element pairs in the two matrices. The arguments must be the same size matrix.

npv() Catalogue > 🗐

npv(InterestRate,CFO,CFList[,CFFreq])

Financial function that calculates net present value: the sum of the present values for the cash inflows and outflows. A positive result for npv indicates a profitable investment.

InterestRate is the rate by which to discount the cash flows (the cost of money) over one period.

CF0 is the initial cash flow at time 0: it must be a real number.

CFList is a list of cash flow amounts after the initial cash flow CF0.

CFFreq is a list in which each element specifies the frequency of occurrence for a grouped (consecutive) cash flow amount, which is the corresponding element of CFList. The default is 1; if you enter values, they must be positive integers < 10.000.

list1:={6000,-8000,2000,-300	
{6000,-800	00,2000,-3000}
list2:={2,2,2,1}	{2,2,2,1}
npv(10,5000, <i>list1</i> , <i>list2</i>)	4769.91

nSolve() Catalogue > 23

 $nSolve(Equation, Var[=Guess]) \Rightarrow number$ or error string

nSolve(Equation, Var[=Guess], lowBound) ⇒number or error string

nSolve(Equation, Var [=Guess], lowBound, upBound) $\Rightarrow number$ or error string

nSolve(Equation, Var[=Guess]) | lowBound $\leq Var \leq upBound \Rightarrow number or error string$

 $nSolve(x^2+5\cdot x-25=9.x)$ 3.84429 -2. 2.

Note: If there are multiple solutions, you can use a guess to help find a particular solution.

Iteratively searches for one approximate real numeric solution to *Equation* for its one variable. Specify the variable as:

variable

variable = real number

For example, x is valid and so is x=3.

nSolve() is often much faster than solve() or zeroes(), particularly if the "|" operator is used to constrain the search to a small interval containing exactly one simple solution.

nSolve() attempts to determine either one point where the residual is zero or two relatively close points where the residual has opposite signs and the magnitude of the residual is not excessive. If it cannot achieve this using a modest number of sample points, it returns the string "no solution found."

Note: See also cSolve(), cZeroes(), solve() and zeroes().

nSolve
$$(x^2+5\cdot x-25=9,x)|_{x<0}$$
 -8.84429
nSolve $\left(\frac{(1+r)^{24}-1}{r}=26,r\right)|_{r>0}$ and $r<0.25$
0.006886
nSolve $(x^2=-1,x)$ "No solution found"

0

OneVar

Catalogue > 🗐

OneVar [1,]X[,[Freq][,Category,Include]]

OneVar [*n*,]*X1*,*X2*[*X3*[,...[,*X20*]]]

Calculates 1-variable statistics on up to 20 lists. A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension except for *Include*.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of numeric category codes for the corresponding X values.

OneVar

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

An empty (void) element in any of the lists X, Freq or *Category* results in a void for the corresponding element of all those lists. An empty element in any of the lists XI through X20 results in a void for the corresponding element of all those lists. For more information on empty elements, see page 248.

Output variable	Description
stat.x	Mean of x values
stat.Σx	Sum of x values
stat. Σx^2	Sum of x ² values
stat.sx	Sample standard deviation of x
stat. x	Population standard deviation of x
stat.n	Number of data points
stat.MinX	Minimum of x values
stat.Q ₁ X	1st Quartile of x
stat.MedianX	Median of x
stat.Q ₃ X	3rd Quartile of x
stat.MaxX	Maximum of x values
stat.SSX	Sum of squares of deviations from the mean of x

or	Cat	alogue > 👰
BooleanExpr1orBooleanExpr2 returns Boolean expression	$x \ge 3$ or $x \ge 4$	<i>x</i> ≥3
BooleanList1 or BooleanList2 returns Boolean list	Define $g(x)$ =Func If $x \le 0$ or $x \ge 5$ Goto end	Done
BooleanMatrix1 or BooleanMatrix2 returns Boolean matrix	Return <i>x</i> ·3 Lbl <i>end</i>	
Returns true or false or a simplified form of the original entry.	EndFunc $ \frac{g(3)}{g(0)} \qquad \qquad A \text{ function did no} $	9 ot return <i>a value</i>
	g(0) A function did no	ot return a value

Returns true if either or both expressions simplify to true. Returns false only if both expressions evaluate to false.

Note: See xor.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Integer1 **or** *Integer2*⇒*integer*

Compares two real integers bit-by-bit using an or operation. Internally, both integers are converted to signed, 64-bit binary numbers. When corresponding bits are compared, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value represents the bit results and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 64-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range. For more information, see >Base2, page 17.

character in character string String, or a list of the first characters of each list element.

Note: See xor.

In Hex base mode:

0h7AC36 or 0h3D5F 0h7BD7F

Important: Zero, not the letter O.

In Bin base mode:

ord({ "alpha", "beta" })

{ 97,98 }

0b100101 or 0b100 0b100101

Note: A binary entry can have up to 64 digits (not counting the 0b prefix). A hexadecimal entry can have up to 16 digits.

ord()	С	atalogue > 👰
ord(String)⇒integer	ord("hello")	104
$ord(List1) \Rightarrow list$	char(104)	"h"
Returns the numeric code of the first	ord(char(24))	507.00

P>Rx()

PRx(rExpr, $\theta Expr$) $\Rightarrow expression$

 $P \rightarrow Rx(rList, \theta List) \Rightarrow list$

PRx(rMatrix, $\theta Matrix$) $\Rightarrow matrix$

Returns the equivalent x-coordinate of the (r, θ) pair.

Note: The θ argument is interpreted as either a degree, gradian or radian angle, according to the current angle mode. If the argument is an expression, you can use °, G or r to override the angle mode setting temporarily.

Note: You can insert this function from the computer keyboard by typing P@>Rx (...).

In Radian angle mode:

$$\begin{array}{c|c} \hline \text{P} + \text{Rx}(r, \theta) & \cos(\theta) \cdot r \\ \hline \text{P} + \text{Rx}(4,60^\circ) & 2 \\ \hline \\ \text{P} + \text{Rx}\left\{\left\{-3,10,1.3\right\}, \left\{\frac{\pi}{3}, \frac{-\pi}{4}, 0\right\}\right\} \\ & \left\{\frac{-3}{2}, 5 \cdot \sqrt{2}, 1.3\right\} \end{array}$$

Catalogue > 2

Catalogue > 🔯 P▶R_V()

 $P Ry(rExpr, \theta Expr) \Rightarrow expression$

 $P \rightarrow R y(rList, \theta List) \Rightarrow list$

 $PPRy(rMatrix, \theta Matrix) \Rightarrow matrix$

Returns the equivalent v-coordinate of the (r, θ) pair.

Note: The θ argument is interpreted as either a degree, radian or gradian angle, according to the current angle mode. If the argument is an expression, you can use °, G or r to override the angle mode setting temporarily.

Note: You can insert this function from the computer keyboard by typing P@>Ry (...).

In Radian angle mode:

$P \triangleright Ry(r,\theta)$	$\sin(\theta) \cdot r$
P▶Ry(4,60°)	2.√3
$P \triangleright \text{Ry}\left\{\left\{-3,10,1.3\right\}, \left\{\frac{\pi}{3}, \frac{-\pi}{4}, 0\right\}\right\}$	
$\left\{\frac{-3\cdot\sqrt{3}}{2}\right.$	·,-5·√2,0.

PassFrr Catalogue > 🕮

PassErr

Passes an error to the next level.

For an example of PassErr, See Example 2 under the Try command, page 188.

If system variable *errCode* is zero, **PassErr** does not do anything.

The Else clause of the Try...Else...EndTry block should use ClrErr or PassErr. If the error is to be processed or ignored, use ClrErr. If what to do with the error is not known, use PassErr to send it to the next error handler. If there are no more pending Try...Else...EndTry error handlers, the error dialogue box will be displayed as normal.

Note: See also CirErr, page 25, and Try, page 188.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

alogue > 🗓

piecewise(*Expr1* [, *Cond1* [, *Expr2* [, *Cond2* [, ...]]]])

Returns definitions for a piecewise function in the form of a list. You can also create piecewise definitions by using a template.

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, x \le 0 \end{cases}$ $\begin{array}{c} p(1) & 1 \\ p(-1) & \text{undef} \end{cases}$

Note: See also Piecewise template, page 2.

poissCdf() Catalogue > [2]

poissCdf(λ,lowBound,upBound)⇒number if lowBound and upBound are numbers, list if lowBound and upBound are lists

poissCdf(λ ,upBound)for P($0 \le X \le upBound$) $\Rightarrow number$ if upBound is a number, list if upBound is a list

Computes a cumulative probability for the discrete Poisson distribution with specified mean λ .

For $P(X \le upBound)$, set lowBound=0

poissPdf() Catalogue > [2]

poissPdf(λ ,XVal) \Rightarrow number if XVal is a number, *list* if XVal is a list

Computes a probability for the discrete Poisson distribution with the specified mean λ .

▶Polar

Catalogue > 🗐

Vector Polar

Note: You can insert this operator from the computer keyboard by typing @>Polar.

Displays *vector* in polar form $[r \angle \theta]$. The vector must be of dimension 2 and can be a row or a column.

Note: ▶Polar is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update ans.

Note: See also > Rect, page 144.

complex Value ▶Polar

Displays *complexVector* in polar form.

- Degree angle mode returns ($r\angle\theta$).
- Radian angle mode returns $re^{i\theta}$.

complex Value can have any complex form. However, an rei θ entry causes an error in Degree angle mode.

Note: You must use the parentheses for an $(r\angle\theta)$ polar entry.

[1 3.]▶Polar $3.16228 \ \angle 1.24905$ x y ▶Polar $\sqrt{x^2+y^2} \quad \angle \frac{\pi \cdot \operatorname{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)$

In Radian angle mode:

$$(3+4\cdot i) \blacktriangleright \text{Polar}$$

$$e^{i \cdot \left(\frac{\pi}{2} - \tan^{-1}\left(\frac{3}{4}\right)\right) \cdot 5}$$

$$\left(\left(4 \angle \frac{\pi}{3}\right)\right) \blacktriangleright \text{Polar}$$

$$e^{\frac{i \cdot \pi}{3} \cdot 4}$$

$$e^{\frac{i \cdot \pi}{3} \cdot 4}$$

In Gradian angle mode:

In Degree angle mode:

$$(3+4\cdot i)$$
 Polar $\left(5 \angle 90-\tan^{-1}\left(\frac{3}{4}\right)\right)$

polyCoeffs()

Catalogue > 23

 $polyCoeffs(Poly[,Var]) \Rightarrow list$

Returns a list of the coefficients of polynomial *Poly* with respect to variable Var.

polyCoeffs
$$\left(4\cdot x^2 - 3\cdot x + 2, x\right)$$
 $\left\{4, -3, 2\right\}$

Poly must be a polynomial expression in Var. We recommend that you do not omit Var unless Poly is an expression in a single variable.

polyCoeffs
$$((x-1)^2 \cdot (x+2)^3)$$
 {1,4,1,-10,-4,8}

Expands the polynomial and selects x for the omitted Var.

polyDegree()

$polyDegree(Poly [, Var]) \Rightarrow value$

Returns the degree of polynomial expression Poly with respect to variable Var. If you omit Var, the polyDegree() function selects a default from the variables contained in the polynomial Poly.

Poly must be a polynomial expression in Var. We recommend that you do not omit Var unless Poly is an expression in a single variable.

Catalogue > 🕮

polyDegree(5)	0
polyDegree($ln(2)+\pi,x$)	0

Constant polynomials

$$\frac{\text{polyDegree}(4 \cdot x^2 - 3 \cdot x + 2, x)}{\text{polyDegree}((x-1)^2 \cdot (x+2)^3)}$$

$$\frac{\text{polyDegree}((x+y^2+z^3)^2,x)}{\text{polyDegree}((x+y^2+z^3)^2,y)} \qquad \qquad \frac{2}{4}$$
$$\frac{10000}{\text{polyDegree}((x-1)^{10000},x)}$$

The degree can be extracted even though the coefficients cannot. This is because the degree can be extracted without expanding the polynomial.

polyEval()

Catalogue > [3]

 $polyEval(List1, Expr1) \Rightarrow expression$

 $polyEval(List1, List2) \Rightarrow expression$

Interprets the first argument as the coefficient of a descending-degree polynomial and returns the polynomial evaluated for the value of the second argument.

$polyEval(\{a,b,c\},x)$	$a \cdot x^2 + b \cdot x + c$
polyEval({1,2,3,4},2)	26
polyEval({1,2,3,4},{2,-7})	{26,-262}

polyGcd()

Catalogue > 23

 $polyGcd(Expr1,Expr2) \Rightarrow expression$

Returns highest common factor of the two arguments.

Expr1 and Expr2 must be polynomial expressions.

List, matrix and Boolean arguments are not allowed.

polyGcd(100,30)	10
$\frac{1}{\operatorname{polyGcd}(x^2-1,x-1)}$	<i>x</i> -1
$\overline{\text{polyGcd}(x^3 - 6 \cdot x^2 + 11 \cdot x - 6, x^2 - 6 \cdot x + 8)}$	
	x-2

polyQuotient()

Catalogue > 🕮

polyQuotient(Poly1,Poly2 [,Var]) $\Rightarrow expression$

Returns the quotient of polynomial *Poly1* divided by polynomial *Poly2* with respect to the specified variable Var.

Poly1 and Poly2 must be polynomial expressions in Var. We recommend that you do not omit Var unless Poly1 and *Poly2* are expressions in the same single variable

polyQuotient(x-1,x-3)	1
$polyQuotient(x-1,x^2-1)$	0
$polyQuotient(x^2-1,x-1)$	x+1
$\overline{\text{polyQuotient}(x^3 - 6 \cdot x^2 + 11 \cdot x - 6, x^2)}$	-6· <i>x</i> +8)
	x

polyQuotient(
$$(x-y)\cdot(y-z),x+y+z,x$$
) $y-z$
polyQuotient($(x-y)\cdot(y-z),x+y+z,y$)
 $2\cdot x-y+2\cdot z$

polyRemainder()

Catalogue > 💱

polyRemainder(
$$Poly1,Poly2$$
 [, Var]) $\Rightarrow expression$

Returns the remainder of polynomial Poly1 divided by polynomial Poly2 with respect to the specified variable Var.

Poly1 and Poly2 must be polynomial expressions in Var. We recommend that you do not omit Var unless Poly1 and Poly2 are expressions in the same single variable.

polyRemainder($x-1,x-3$)	2
$polyRemainder(x-1,x^2-1)$	<i>x</i> -1
$\frac{1}{\text{polyRemainder}(x^2-1,x-1)}$	0

polyRemainder(
$$(x-y)\cdot(y-z),x+y+z,x$$
)
 $-(y-z)\cdot(2\cdot y+z)$
 polyRemainder($(x-y)\cdot(y-z),x+y+z,y$)
 $-2\cdot x^2-5\cdot x\cdot z-2\cdot z^2$
 polyRemainder($(x-y)\cdot(y-z),x+y+z,z$)
 $(x-y)\cdot(x+2\cdot y)$

polyRoots()

Catalogue > 🗐

 $polyRoots(Poly,Var) \Rightarrow list$

 $polyRoots(ListOfCoeffs) \Rightarrow list$

The first syntax, polyRoots(Poly,Var), returns a list of real roots of polynomial Poly with respect to variable Var. If no real roots exist, returns an empty list: $\{\ \}$.

Poly must be a polynomial in one variable.

The second syntax, **polyRoots** (*ListOfCoeffs*), returns a list of real roots for the coefficients in *ListOfCoeffs*.

Note: See also cPolyRoots(), page 36.

polyRoots (y^3+1,y)	{-1}
cPolyRoots(y3+1,y)	
$\left\{-1,\frac{1}{2}\right\}$	$-\frac{\sqrt{3}}{2}\mathbf{i},\frac{1}{2}+\frac{\sqrt{3}}{2}\mathbf{i}$
$\overline{\text{polyRoots}(x^2 + 2\cdot x + 1, x)}$	{-1,-1}
polyRoots({1,2,1})	{-1,-1}

PowerReg

Catalogue > 🔯

PowerReg X,Y [, Freq] [, Category, Include]]

Computes the power regressiony = $(a \cdot (x)^b)$ on lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for *Include*.

X and Y are lists of independent and dependent variables.

Catalogue > [3]

PowerReg

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: a \cdot (x) ^b
stat.a, stat.b	Regression coefficients
stat.r ²	Coefficient of linear determination for transformed data
stat.r	Correlation coefficient for transformed data (ln(x), ln(y))
stat.Resid	Residuals associated with the power model
stat.ResidTrans	Residuals associated with linear fit of transformed data
stat.XReg	List of data points in the modified $X\ List$ actually used in the regression based on restrictions of $Freq$, $Category\ List$ and $Include\ Categories$
stat.YReg	List of data points in the modified Y List actually used in the regression based on restrictions of Freq, Category List and Include Categories
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

Catalogue > 23 **Prgm**

Prgm Block **EndPrgm** Calculate GCD and display intermediate results.

Template for creating a user-defined programme. Must be used with the Define, Define LibPub or Define LibPriv command.

Prgm

Catalogue > 23

Done

Block can be a single statement, a series of statements separated with the ":" character or a series of statements on separate lines.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define $proggcd(a,b)$ =Prgn	ı
Loca	ıl d
Whi	e <i>b</i> ≠0
d:≕n	$\operatorname{rod}(a,b)$
a := b	
b:= d	•
Disp	a," ",b
End	While
Disp	"GCD=", a
Endl	Prgm

proggcd(4560,450)	
	450 60
	60 30
	30 0
	GCD=30
	Done

prodSeq() See Π (), page 220.

Product (PI) See Π (), page 220.

product()

Catalogue > 📳

 $product(List[, Start[, End]]) \Rightarrow expression$

Returns the product of the elements contained in *List*. *Start* and *End* are optional. They specify a range of elements.

 $product(Matrix 1[, Start[, End]]) \Rightarrow matrix$

Returns a row vector containing the products of the elements in the columns of *Matrix 1*. *Start* and *end* are optional. They specify a range of rows.

Empty (void) elements are ignored. For more information on empty elements, see page 248.

product({1,2,3,4})	24
$\operatorname{product}(\{2,x,y\})$	2·x·y
product({4,5,8,9},2,3)	40

$ \frac{1}{4} $	2 5 8	3 6 9	[28 80 162]
	2 5 8	$\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}, 1, 2 $	[4 10 18]

 $propFrac(Expr1[, Var]) \Rightarrow expression$

propFrac(rational number) returns rational number as the sum of an integer and a fraction having the same sign and a greater denominator magnitude than numerator magnitude.

propFrac(rational expression,Var) returns the sum of proper ratios and a polynomial with respect to Var. The degree of Var in the denominator exceeds the degree of Var in the numerator in each proper ratio. Similar powers of *Var* are collected. The terms and their factors are sorted with Var as the main variable.

If Var is omitted, a proper fraction expansion is done with respect to the most main variable. The coefficients of the polynomial part are then made proper with respect to their most main variable first and so on.

For rational expressions, propFrac() is a faster but less extreme alternative to expand().

You can use the propFrac() function to represent mixed fractions and demonstrate addition and subtraction of mixed fractions.

$\operatorname{propFrac}\left(\frac{4}{3}\right)$	$1 + \frac{1}{3}$
${\text{propFrac}\left(\frac{-4}{3}\right)}$	$-1-\frac{1}{3}$

$$\operatorname{propFrac}\left(\frac{x^{2} + x + 1}{x + 1} + \frac{y^{2} + y + 1}{y + 1}, x\right)$$

$$\frac{1}{x + 1} + x + \frac{y^{2} + y + 1}{y + 1}$$

$$\operatorname{propFrac}(Ans)$$

$$\frac{1}{x + 1} + x + \frac{1}{y + 1} + y$$

$\operatorname{propFrac}\left(\frac{11}{7}\right)$	$1 + \frac{4}{7}$
$propFrac\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8 + \frac{37}{44}$
$propFrac\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2 - \frac{29}{44}$

Q

QR

QR *Matrix*, *qMatrix*, *rMatrix*[, *Tol*]

Calculates the Householder QR factorization of a real or complex matrix. The resulting Q and R matrices are stored to the specified Matrix. The Q matrix is unitary. The R matrix is upper triangular.

Catalogue > 🗐

The floating-point number (9.) in m1 causes results to be calculated in floating-point form.

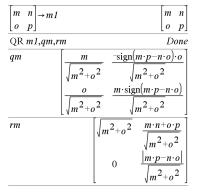
QR Catalogue > 🚉

Optionally, any matrix element is treated as zero if its absolute value is less than *Tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *Tol* is ignored.

- If you use ctrl enter or set the Auto or Approximate mode to Approximate, computations are done using floatingpoint arithmetic.
- If Tol is omitted or not used, the default tolerance is calculated as: 5E-14 ·max(dim(Matrix)) ·rowNorm (Matrix)

The QR factorization is computed numerically using Householder transformations. The symbolic solution is computed using Gram-Schmidt. The columns in *qMatName* are the orthonormal basis vectors that span the space defined by *matrix*.

1 2 3		1	2	3
4 5 6	$\rightarrow m1$	4	5	6
7 8 9.		[7	8	9.]
QR <i>m1,q</i>	n,rm		D	one
qm	0.123091 0.904534	0.40	082	48
	0.492366 0.301511	-0.8	164	197
	0.86164 -0.301511	0.40	082	48]
rm	8.12404 9.60114	. 11	.07	82
	0. 0.904534	4 1.	809	07
	0. 0.		0.	



QuadReg Catalogue > 2

QuadReg X,Y [, Freq] [, Category, Include]]

Computes the quadratic polynomial regressiony = $a \cdot x^2 + b \cdot x + con$ lists X and Y with frequency Freq. A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for *Include*.

 \boldsymbol{X} and \boldsymbol{Y} are lists of independent and dependent variables.

QuadReg

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.RegEqn	Regression equation: a ·x²+b ·x+c
stat.a, stat.b, stat.c	Regression coefficients
stat.R ²	Coefficient of determination
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

Catalogue > 23 QuartReg

QuartReg X,Y [, Freq] [, Category, Include]]

Computes the quartic polynomial regressiony = a $\cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + eon lists X and Y with$ frequency *Freq.* A summary of results is stored in the stat.results variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in Freq specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers \geq 0.

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

r	
Output variable	Description
stat.RegEqn	Regression equation: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Regression coefficients
stat.R ²	Coefficient of determination
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified <i>X List</i> actually used in the regression based on restrictions of <i>Freq</i> , <i>Category List</i> and <i>Include Categories</i>
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$ and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

R

R ▶ P θ()	Catalogue > 🗐

 $R \triangleright P\theta (xExpr, yExpr) \Rightarrow expression$

 $R \triangleright P\theta$ (xList, yList) \Rightarrow list $R \triangleright P\theta$ (xMatrix, yMatrix) \Rightarrow matrix

Returns the equivalent θ -coordinate of the (x,y) pair arguments.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

$$\mathbb{R} \triangleright \mathbb{P}\theta(x,y)$$
 $90 \cdot \operatorname{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$

In Gradian angle mode:

$$R \triangleright P\theta(x,y) \qquad 100 \cdot \operatorname{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$$

R▶**P**θ()

Catalogue > 23

Note: You can insert this function from the computer keyboard by typing R@>Ptheta (...).

In Radian angle mode:

R▶Pθ
$$\left[\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix} \right]$$
 $\left[\begin{bmatrix} 0 & \tan^{-1}\left(\frac{16}{\pi}\right) + \frac{\pi}{2} & 0.643501 \end{bmatrix} \right]$

R►Pr()

Catalogue > 🔯

 $R \triangleright Pr(xExpr, yExpr) \Rightarrow expression$

 $R \triangleright Pr(xList, yList) \Rightarrow list$ $R \triangleright Pr(xMatrix, yMatrix) \Rightarrow matrix$

Returns the equivalent r-coordinate of the (x,y) pair arguments.

Note: You can insert this function from the computer keyboard by typing R@>Pr (...).

In Radian angle mode:

$$\mathbb{R} \bullet \Pr \left[\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix} \right]$$

$$\left[3 & \frac{\sqrt{\pi^2 + 256}}{4} & 2.5 \end{bmatrix}$$

▶ Rad

Catalogue > 🕮

 $Expr1 \triangleright Rad \Rightarrow expression$

Converts the argument to radian angle measure.

Note: You can insert this operator from the computer keyboard by typing @>Rad.

In Degree angle mode:

In Gradian angle mode:

rand()

Catalogue > 🗐

 $rand() \Rightarrow expression$ $rand(\#Trials) \Rightarrow list$

rand() returns a random value between 0 and 1.

rand(#Trials) returns a list containing #Trials random values between 0 and 1. Set the random-number seed.

RandSeed 1147	Done
rand(2)	{0.158206,0.717917}

randBin()

Catalogue > 23

randBin(n, p) \Rightarrow expression randBin(n, p, #Trials) \Rightarrow list

randBin(80,0.5) 42 randBin(80,0.5,3) {41,32,39}

randBin(*n*, *p*) returns a random real number from a specified Binomial distribution.

randBin(n, p, #Trials) returns a list containing #Trials random real numbers from a specified Binomial distribution.

Catalogue > 🗐

randint

(lowBound,upBound)

⇒ expression

randint

(lowBound,upBound ,#Trials) ⇒ list

randInt

(lowBound,upBound) returns a random integer within the range specified by lowBound and upBound integer bounds.

randint

randint
(lowBound,upBound
,#Trials) returns a
list containing
#Trials random
integers within the
specified range.

randInt(3,10)	5
randInt(3,10,4)	{9,7,5,8}

randMat() Catalogue > [[3]

randMat(numRows, numColumns) \Rightarrow matrix

Returns a matrix of integers between -9 and 9 of the specified dimension.

 $Both \ arguments \ must \ simplify \ to \ integers.$

RandSeed 1147	Done			
randMat(3,3)	[8	8	-3	6
	-	2	3	-6
	[(C	4	-6]

Note: The values in this matrix will change each time you press [enter].

randNorm()

Catalogue > 23

 $randNorm(\mu, \sigma) \Rightarrow expression$ $randNorm(\mu, \sigma, \#Trials) \Rightarrow list$ RandSeed 1147 Done randNorm(0,1) 0.492541randNorm(3,4.5) -3.54356

randNorm(μ, σ) returns a decimal number from the specified normal distribution. It could be any real number but will be heavily concentrated in the interval $[\mu-3\bullet\sigma, \mu+3\bullet\sigma]$.

 $randNorm(\mu, \sigma, \#Trials)$ returns a list containing #Trials decimal numbers from the specified normal distribution.

randPoly()	Catalogue > 🗐

 $randPoly(Var, Order) \Rightarrow expression$

Returns a polynomial in Var of the specified *Order*. The coefficients are random integers in the range -9 through 9. The leading coefficient will not be zero.

Order must be 0-99.

RandSeed 1147 Done randPoly
$$(x,5)$$
 $-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$

randSamp()

 $randSamp(List, \#Trials[, noRepl]) \Rightarrow list$

Returns a list containing a random sample of #Trials trials from List with an option for sample replacement (noRepl=0), or no sample replacement (noRepl=1). The default is with sample replacement.

Define $list3 = \{1,2,3,4,5\}$ Done Define *list4*=randSamp(*list3*,6) Done

Catalogue > 🕮

{2,3,4,3,1,2}

RandSeed Catalogue > 🕮

list4

RandSeed Number

If Number = 0, sets the seeds to the factory defaults for the random-number generator. If $Number \neq 0$, it is used to generate two seeds, which are stored in system variables seed1 and seed2.

RandSeed 1147	Done
rand()	0.158206

real()

Catalogue > 📳

$real(Expr1) \Rightarrow expression$

Returns the real part of the argument.

Note: All undefined variables are treated as real variables. See also **imag()**, page 88.

$$real(List1) \Rightarrow list$$

Returns the real parts of all elements.

$$real(Matrix 1) \Rightarrow matrix$$

Returns the real parts of all elements.

$real(2+3\cdot i)$	2
real(z)	Z
$real(x+i\cdot y)$	x

$$real(\{a+i\cdot b,3,i\}) \qquad \qquad \{a,3,0\}$$

▶ Rect

Catalogue > 🗐

Vector ▶Rect

Note: You can insert this operator from the computer keyboard by typing @>Rect.

Displays *Vector* in rectangular form [x, y, z]. The vector must be of dimension 2 or 3 and can be a row or a column.

Note: Rect is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update *ans*.

Note: See also ▶ Polar, page 131.

complex Value ▶ Rect

Displays complexValue in rectangular form a+bi. The complexValue can have any complex form. However, an $re^{i\theta}$ entry causes an error in Degree angle mode.

Note: You must use parentheses for an $(r \angle \theta)$ polar entry.

$$\left[3 \ \angle \frac{\pi}{4} \ \angle \frac{\pi}{6} \right] \triangleright \text{Rect} \\
\left[\frac{3 \cdot \sqrt{2}}{4} \ \frac{3 \cdot \sqrt{2}}{4} \ \frac{3 \cdot \sqrt{3}}{2} \right]$$

 $[a \cdot \cos(b) \cdot \sin(c) \quad a \cdot \sin(b) \cdot \sin(c) \quad a \cdot \cos(c)]$

In Radian angle mode:

$$\frac{\pi}{4 \cdot e^{\frac{\pi}{3}}} \operatorname{Rect} \qquad \frac{\pi}{4 \cdot e^{\frac{\pi}{3}}}$$

$$\frac{\pi}{4 \cdot e^{\frac{\pi}{3}}} \operatorname{Rect} \qquad 2 + 2 \cdot \sqrt{3} \cdot \frac{\pi}{3}$$

In Gradian angle mode:

In Degree angle mode:

$$((4 \angle 60))$$
 Rect $2+2\cdot\sqrt{3}\cdot i$

Note: To type \angle , select it from the symbol list in the Catalogue.

 $ref(Matrix 1[, Tol]) \Rightarrow matrix$

Returns the row echelon form of *Matrix1*.

Optionally, any matrix element is treated as zero if its absolute value is less than *Tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *Tol* is ignored.

- If you use ctri enter or set the Auto or Approximate mode to Approximate, computations are done using floatingpoint arithmetic.
- If *Tol* is omitted or not used, the default tolerance is calculated as: $5E-14 \cdot max(dim(Matrix 1)) \cdot rowNorm$ (Matrix 1)

Avoid undefined elements in *Matrix 1*. They can lead to unexpected results.

For example, if *a* is undefined in the following expression, a warning message appears and the result is shown as:

The warning appears because the generalized element 1/a would not be valid for a=0.

You can avoid this by storing a value to a beforehand or by using the constraint ("|") operator to substitute a value, as shown in the following example.

$$ref \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} | a = 0 \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Note: See also rref(), page 154.

$$\operatorname{ref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$ref(m1) \qquad \begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$$

RefreshProbeVars

Allows you to access sensor data from all connected sensor probes in your TI-Basic program.

StatusVar Value	Status
statusVar =0	Normal (continue with the program)
	The Vernier DataQuest™ application is in data collection mode.
statusVar =1	Note: The Vernier DataQuest™ application must be in meter mode for this command to work.
=2	The Vernier DataQuest™ application is not launched.
statusVar =3	The Vernier DataQuest™ application is launched, but you have not connected any probes.

Example

Define temp()=

Prqm

© Check if system is ready

RefreshProbeVars status

If status=0 Then

Disp "ready"

For n, 1, 50

RefreshProbeVars status

temperature:=meter.temperature

Disp "Temperature:

", temperature

If temperature>30 Then

Disp "Too hot"

EndIf

© Wait for 1 second between samples

Wait 1

EndFor

Else

Disp "Not ready. Try again

later"

EndIf

EndPrqm

Note: This can also be used with TI-InnovatorTM Hub.

remain()

 $remain(Expr1, Expr2) \Rightarrow expression$

 $remain(List1, List2) \Rightarrow list$ $remain(Matrix 1. Matrix 2) \Rightarrow matrix$

Returns the remainder of the first argument with respect to the second argument as defined by the identities:

remain(x,0) x remain(x,y) x-y•iPart(x/y)

As a consequence, note that remain(-x,y) remain(x,y). The result is either zero or it has the same sign as the first argument.

Note: See also mod(), page 114.

remain(7,0)	7
remain(7,3)	1
remain(-7,3)	-1
remain(7,-3)	1
remain(-7,-3)	-1
remain({12,-14,16},{9,7,-5})	{3,0,1}

	-	_				-	
remain	9	-7	4	3	}	1	-1
\	6	4	4	-3	}	2	1

Request

Request promptString, var[, DispFlag [, status Var]]

Request promptString, func(arg1, ...argn) [, DispFlag [, statusVar]]

Programming command: Pauses the program and displays a dialog box containing the message *promptString* and an input box for the user's response.

When the user types a response and clicks OK, the contents of the input box are assigned to variable var.

If the user clicks Cancel, the program proceeds without accepting any input. The program uses the previous value of var if *var* was already defined.

The optional *DispFlag* argument can be any expression.

- If DispFlag is omitted or evaluates to **1**, the prompt message and user's response are displayed in the Calculator history.
- If DispFlag evaluates to $\mathbf{0}$, the prompt and response are not displayed in the history.

Catalogue >

Catalogue > 🕮

Define a program:

Define request_demo()=Prgm Request "Radius: ",r Disp "Area = ",pi*r2 EndPrgm

Run the program and type a response:

request demo()



Result after selecting OK:

Radius: 6/2 Area= 28.2743

Request

The optional *statusVar* argument gives the program a way to determine how the user dismissed the dialog box. Note that status Var requires the DispFlag argument.

- If the user clicked **OK** or pressed **Enter** or **Ctrl+Enter**. variable *statusVar* is set to a value of 1.
- Otherwise, variable status Var is set to a value of 0.

The func() argument allows a program to store the user's response as a function definition. This syntax operates as if the user executed the command:

Define func(arg1, ...argn) = user'sresponse

The program can then use the defined function func(). The promptString should guide the user to enter an appropriate user's response that completes the function definition.

Note: You can use the Request command within a user-defined program but not within a function.

To stop a program that contains a Request command inside an infinite loop:

- Handheld: Hold down the file on key and press enter repeatedly.
- Windows®: Hold down the F12 key and press Enter repeatedly.
- Macintosh®: Hold down the F5 key and press Enter repeatedly.
- iPad®: The app displays a prompt. You can continue waiting or cancel.

Note: See also RequestStr, page 148.

Define a program:

Define polynomial()=Prgm Request "Enter a polynomial in p(x)Disp "Real roots are:",polyRoots (p(x),x)EndPrgm

Run the program and type a response:

polynomial()



Result after entering x^3+3x+1 and selecting OK:

Real roots are: {-0.322185}

RequestStr

Catalogue > 🕮

RequestStr promptString, var[, DispFlag]

Define a program:

RequestStr



Programming command: Operates identically to the first syntax of the Request command, except that the user's response is always interpreted as a string. By contrast, the **Request** command interprets the response as an expression unless the user encloses it in quotation marks ("").

Note: You can use the RequestStr command within a user-defined program but not within a function.

To stop a program that contains a RequestStr command inside an infinite loop:

- Handheld: Hold down the Gion key and press enter repeatedly.
- Windows®: Hold down the F12 key and press **Enter** repeatedly.
- Macintosh®: Hold down the F5 key and press **Enter** repeatedly.
- iPad®: The app displays a prompt. You can continue waiting or cancel.

Note: See also Request, page 147.

Define requestStr demo()=Prgm RequestStr "Your name:", name, 0 Disp "Response has ",dim(name)," characters." EndPrgm

Run the program and type a response:

requestStr demo()



Result after selecting **OK** (Note that the DispFlag argument of **0** omits the prompt and response from the history):

requestStr demo()

Response has 5 characters.

Return

Return [Expr]

Returns Expr as the result of the function. Use within a Func...EndFunc block.

Note: Use Return without an argument within a Prgm...EndPrgm block to exit a program.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Catalogue > 23

Define factorial (nn)= Func Local answer,counter $1 \rightarrow answer$ For counter,1,nn answer · counter → answer EndFor Return answer EndFunc factorial (3) 6

Catalogue > 23 right() $right(List1[,Num]) \Rightarrow list$ right($\{1,3,-2,4\},3$) $\{3,-2,4\}$

Catalogue > 23

Returns the rightmost Num elements contained in List1.

If you omit *Num*, returns all of *List1*.

 $right(sourceString[, Num]) \Rightarrow string$

Returns the rightmost *Num* characters contained in character string *sourceString*.

If you omit *Num*, returns all of *sourceString*.

 $right(Comparison) \Rightarrow expression$

Returns the right side of an equation or inequality.

$$right(x<3)$$
 3

rk23 ()

rk23(Expr, Var, depVar, {Var0, VarMax}, depVar0, VarStep [, diftol]) \Rightarrow matrix

rk23(SystemOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, VarStep[, diftol]) \Rightarrow matrix

rk23(*ListOfExpr*, *Var*, *ListOfDepVars*, $\{Var0, VarMax\}$, *ListOfDepVars0*, VarStep[, diftol]) $\Rightarrow matrix$

Uses the Runge-Kutta method to solve the system

$$\frac{d depVar}{d Var} = Expr(Var, depVar)$$

with $depVar(Var\theta)=depVar\theta$ on the interval $[Var\theta,VarMax]$. Returns a matrix whose first row defines the Var output values as defined by VarStep. The second row defines the value of the first solution component at the corresponding Var values, and so on.

Expr is the right hand side that defines the ordinary differential equation (ODE).

SystemOfExpr is a system of right-hand sides that define the system of ODEs (corresponds to order of dependent variables in ListOfDepVars).

Differential equation:

y'=0.001*y*(100-y) and y(0)=10

rk23
$$(0.001 \cdot y \cdot (100 - y), t, y, \{0,100\}, 10, 1)$$
 $\begin{bmatrix} 0. & 1. & 2. & 3. & 4\\ 10. & 10.9367 & 11.9493 & 13.042 & 14.2 \end{bmatrix}$

To see the entire result, press ▲ and then use ◄ and ▶ to move the cursor.

Same equation with diftol set to 1.E-6

Compare above result with CAS exact solution obtained using deSolve() and seqGen():

deSolve(
$$y$$
'=0.001· y ·(100- y) and y (0)=10, t , y)
$$y = \frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9}.$$

seqGen
$$\left\{ \frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9}, t_y, \{0,100\} \right\}$$
 $\left\{ 10., 10.9367, 11.9494, 13.0423, 14.2189, 15.48 \right\}$

System of equations:

ListOfExpr is a list of right-hand sides that define the system of ODEs (corresponds to order of dependent variables in ListOfDepVars).

Var is the independent variable.

ListOfDepVars is a list of dependent variables.

{Var0, VarMax} is a two-element list that tells the function to integrate from Var0 to VarMax.

ListOfDepVars0 is a list of initial values for dependent variables.

If *VarStep* evaluates to a nonzero number: sign(VarStep) = sign(VarMax-Var0) and solutions are returned at Var0+i*VarStep for all i=0,1,2,... such that Var0+i*VarStepis in [var0, VarMax] (may not get a solution value at VarMax).

if VarStep evaluates to zero, solutions are returned at the "Runge-Kutta" Var values.

diftol is the error tolerance (defaults to 0.001).

$$\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}$$

with y1(0)=2 and y2(0)=5

rk23
$$\left\{\begin{bmatrix} -yI+0.1\cdot yI\cdot y2\\ 3\cdot y2-yI\cdot y2 \end{bmatrix}, t, \{yI,y2\}, \{0,5\}, \{2,5\}, 1 \right\}$$

 $\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \end{pmatrix}$

5. 16.8311 12.3133 3.51112 6.27245

root()		Catalogue > 🕡
$root(Expr) \Rightarrow root$ $root(Expr1, Expr2) \Rightarrow root$	3√8	2
root(<i>Expr</i>) returns the square root of <i>Expr</i> .	$\sqrt[3]{3}$	$\frac{1}{3}$
root(Expr1, Expr2) returns the Expr2 root of Expr1. Expr1 can be a real or complex	3 √3.	1.44225

Note: See also Nth root template, page 2.

floating point constant, an integer or complex rational constant, or a general

symbolic expression.

rotate() Catalogue > 🗐

 $rotate(Integer 1[, \#ofRotations]) \Rightarrow integer$ In Bin base mode:

rotate()

Catalogue > 23

Rotates the bits in a binary integer. You can enter *Integer I* in any number base; it is converted automatically to a signed, 64-bit binary form. If the magnitude of *Integer I* is too large for this form, a symmetric modulo operation brings it within the range. For more information, see **Base2**, page 17.

If #ofRotations is positive, the rotation is to the left. If #ofRotations is negative, the rotation is to the right. The default is -1 (rotate right one bit).

For example, in a right rotation:

Each bit rotates right.

0b0000000000001111010110000110101

Rightmost bit rotates to leftmost.

produces:

0b10000000000000111101011000011010

The result is displayed according to the Base mode.

 $rotate(List1[,\#ofRotations]) \Rightarrow list$

Returns a copy of *List1* rotated right or left by #of Rotations elements. Does not alter *List1*.

If #ofRotations is positive, the rotation is to the left. If #of Rotations is negative, the rotation is to the right. The default is -1 (rotate right one element).

 $rotate(String1[,\#ofRotations]) \Rightarrow string$

Returns a copy of *String1* rotated right or left by #ofRotations characters. Does not alter *String1*.

If #ofRotations is positive, the rotation is to the left. If #ofRotations is negative, the rotation is to the right. The default is -1 (rotate right one character).

rotate(0b11111111111	111111111111111111111111111111111111111
0b100000000000000000000000000000000000	000000000000000000011
rotate(256,1)	0b1000000000

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

In Hex base mode:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h800000000000001E3
rotate(0h78E,2)	0h1E38

Important: To enter a binary or hexadecimal number, always use the 0b or 0h prefix (zero, not the letter O).

In Dec base mode:

rotate({1,2,3,4})	$\{4,1,2,3\}$
rotate({1,2,3,4},-2)	{3,4,1,2}
rotate({1,2,3,4},1)	{2,3,4,1}

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

round()

Catalogue > 23

 $round(Expr1[, digits]) \Rightarrow expression$

round(1.234567,3)

Returns the argument rounded to the specified number of digits after the decimal point.

digits must be an integer in the range 0-12. If *digits* is not included, returns the argument rounded to 12 significant digits.

Note: Display digits mode may affect how this is displayed.

 $round(List 1[. digits]) \Rightarrow list$

Returns a list of the elements rounded to the specified number of digits.

 $round(Matrix 1[, digits]) \Rightarrow matrix$

Returns a matrix of the elements rounded to the specified number of digits.

round(
$$\{\pi,\sqrt{2},\ln(2)\},4$$
)
 $\{3.1416,1.4142,0.6931\}$

round	(5) ln(3)] 1	1.6	1.1
Tound .	π е	1 1	3.1	2.7

rowAdd()

 $rowAdd(Matrix1, rIndex1, rIndex2) \Rightarrow$ matrix

Returns a copy of *Matrix1* with row rIndex2 replaced by the sum of rows rIndex 1 and rIndex 2.

Catalogue > 🗐

$rowAdd \begin{bmatrix} 3 & 4 \\ -3 & -2 \end{bmatrix}, 1, 2 $		$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$
$rowAdd \begin{pmatrix} a & b \\ c & d \end{pmatrix}, 1, 2 \end{pmatrix}$	$\begin{bmatrix} a \\ a+c \end{bmatrix}$	$\begin{bmatrix} b \\ b+d \end{bmatrix}$

rowDim()

 $rowDim(Matrix) \Rightarrow expression$

Returns the number of rows in Matrix.

Note: See also colDim(), page 26.

Catalogue > 🗐

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	1 2 3 4 5 6
[2 0]	[3 0]
rowDim(m1)	3

ro

rowNorm()

 $rowNorm(Matrix) \Rightarrow expression$

Returns the maximum of the sums of the absolute values of the elements in the rows in Matrix.

Note: All matrix elements must simplify to numbers. See also colNorm(), page 26.

Catalogue > 🕮

rowNorm
$$\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}$$
 25

rowSwap()

rowSwap(Matrix 1, rIndex 1, rIndex 2) \Rightarrow matrix

Returns *Matrix1* with rows *rIndex1* and *rIndex2* exchanged.

1 2	1	2
$\begin{vmatrix} 3 & 4 \end{vmatrix} \rightarrow mat$	3	4
[5 6]	5	6
rowSwap(mat,1,3)	5	6
	3	4
	1	2

rref()

 $rref(Matrix 1[, Tol]) \Rightarrow matrix$

Returns the reduced row echelon form of *Matrix I*.

Optionally, any matrix element is treated as zero if its absolute value is less than *Tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *Tol* is ignored.

- If you use <u>ctrl</u> <u>enter</u> or set the **Auto or Approximate** mode to Approximate, computations are done using floating-point arithmetic.
- If Tol is omitted or not used, the default tolerance is calculated as: 5E-14 •max(dim(Matrix I)) •rowNorm (Matrix I)

Note: See also ref(), page 145.

Catalogue > 😰

Catalogue > 🕮

$$\operatorname{rref} \left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \right) \qquad \begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

S

sec() trig key

 $sec(Expr1) \Rightarrow expression$

 $sec(List1) \Rightarrow list$

Returns the secant of *Expr1* or returns a list containing the secants of all elements in *List1*.

In Degree angle mode:

$$\frac{\sec(45)}{\sec(\{1,2.3,4\})} \qquad \frac{\sqrt{2}}{\left\{\frac{1}{\cos(1)},1.00081,\frac{1}{\cos(4)}\right\}}$$

sec()

trig kev

Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode setting. You can use °, G, or r to override the angle mode temporarily.

sec-1()

trig key

 $sec-1(Expr1) \Rightarrow expression$

$$sec-1(List1) \Rightarrow list$$

Returns the angle whose secant is *Expr1* or returns a list containing the inverse secants of each element of *List1*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arcsec (...).

In Degree angle mode:

In Gradian angle mode:

$$\sec^{-1}(\sqrt{2})$$
 50

In Radian angle mode:

$$\left\{0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right)\right\}$$

sech()

Catalogue > 🗐

 $sech(Expr1) \Rightarrow expression$

$$sech(List1) \Rightarrow list$$

Returns the hyperbolic secant of Expr1 or returns a list containing the hyperbolic secants of the List1 elements.

sech(3) cosh(3 sech({1,2.3,4}) $\frac{1}{\cosh(1)}$, 0.198522,

sech-1()

Catalogue > 🕮

 $sech-1(Expr1) \Rightarrow expression$

$$sech-1(List1) \Rightarrow list$$

Returns the inverse hyperbolic secant of Expr1 or returns a list containing the inverse hyperbolic secants of each element of List1.

Note: You can insert this function from the keyboard by typing arcsech (...).

In Radian angle and Rectangular complex mode:

sech³(1) 0
sech³(
$$\{1, -2, 2.1\}$$
) $\{0, \frac{2 \cdot \pi}{3} \cdot i, 8. \epsilon^{-1} + 1.07448 \cdot i\}$

Send Hub Menu

Send exprOrString1[, exprOrString2] ...

Programming command: Sends one or more TI-Innovator™ Hub commands to a connected hub.

exprOrString must be a valid
TI-Innovator™ Hub Command. Typically,
exprOrString contains a "SET ..." command
to control a device or a "READ ..." command
to request data.

The arguments are sent to the hub in succession.

Note: You can use the **Send** command within a user-defined programme but not within a function.

Note: See also Get (page 77), GetStr (page 83), and eval() (page 61).

Example: Turn on the blue element of the built-in RGB LED for 0.5 seconds.

Example: Request the current value of the hub's built-in light-level sensor. A **Get** command retrieves the value and assigns it to variable *lightval*.

Send "READ BRIGHTNESS"	Done
Get lightval	Done
lightval	0.347922

Example: Send a calculated frequency to the hub's built-in speaker. Use special variable *iostr.SendAns* to show the hub command with the expression evaluated.

n:=50		50
m:=4		4
Send "SET SOUND eval(m	n)"	Done
iostr.SendAns	"SET SOUNI	200"

seq()

 $seq(Expr, Var, Low, High[, Step]) \Rightarrow list$

Increments Var from Low through High by an increment of Step, evaluates Expr, and returns the results as a list. The original contents of Var are still there after seq() is completed.

The default value for Step = 1.

Catalogue > 🕮

$\overline{\operatorname{seq}(n^2,n,1,6)}$	{1,4,9,16,25,36}
$\frac{1}{\operatorname{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)}$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\operatorname{sum}\left\{\operatorname{seq}\left(\frac{1}{2},n,1,10,1\right)\right\}$	1968329
$\operatorname{sum}\left\{\operatorname{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right\}$	1270080

Note: To force an approximate result,

Handheld: Press ctrl enter.

Windows®: Press Ctrl+Enter.

Macintosh®: Press #Enter.

iPad®: Hold enter, and select ≈ ..

$$\overline{\operatorname{sum}\left(\operatorname{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)}$$
 1.54977

 $seqGen(Expr, Var, depVar, \{Var0,$ VarMax}[, ListOfInitTerms [, VarStep[, $CeilingValue]]]) <math>\Rightarrow list$

Generates a list of terms for sequence depVar(Var)=Expr as follows: Increments independent variable *Var* from *Var0* through VarMax by VarStep, evaluates depVar(Var) for corresponding values of Var using the Expr formula and ListOfInitTerms, and returns the results as a list.

seqGen(ListOrSystemOfExpr, Var, *ListOfDepVars*, {*Var0*, *VarMax*} [, MatrixOfInitTerms[, VarStep[, CeilingValue]]]) $\Rightarrow matrix$

Generates a matrix of terms for a system (or list) of sequences *ListOfDepVars* (*Var*)=*ListOrSystemOfExpr* as follows: Increments independent variable Var from *Var0* through *VarMax* by *VarStep*, evaluates ListOfDepVars(Var) for corresponding values of Var using ListOrSystemOfExpr formula and MatrixOfInitTerms, and returns the results as a matrix.

The original contents of *Var* are unchanged after segGen() is completed.

The default value for VarStep = 1.

Generate the first 5 terms of the sequence u $(n) = u(n-1)^2/2$, with u(1)=2 and VarStep=1.

$$\frac{1}{\text{seqGen}\left(\frac{(u(n-1))^{2}}{n}, n, u, \{1,5\}, \{2\}\right)} \left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

Example in which Var0=2:

seqGen
$$\left(\frac{u(n-1)+1}{n}, n, u, \{2,5\}, \{3\}\right)$$
 $\left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$

Example in which initial term is symbolic:

$$\frac{\operatorname{seqGen}(u(n-1)+2,n,u,\{1,5\},\{a\})}{\{a,a+2,a+4,a+6,a+8\}}$$

System of two sequences:

$$\begin{split} \operatorname{seqGen} \! \left\{ & \frac{1}{n}, \frac{u \not (n-1)}{2} \! + \! u I(n-1) \right\}, \! \left\{ u I, \! u 2 \right\}, \! \left\{ 1, \! 5 \right\} \begin{bmatrix} -1 \\ 2 \end{bmatrix} \\ & \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix} \end{split}$$

Note: The Void () in the initial term matrix above is used to indicate that the initial term for u1(n) is calculated using the explicit sequence formula u1(n)=1/n.

segn()

seqn(Expr(u, n[, ListOfInitTerms[, nMax[, CeilingValue]]]) $\Rightarrow list$

Catalogue > 23

Generate the first 6 terms of the sequence u(n) = u(n-1)/2, with u(1)=2.

$$\frac{\operatorname{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right)}{\left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}}$$

seqn()

Catalogue > 23

Generates a list of terms for a sequence u (n)=Expr(u,n) as follows: Increments n from 1 through nMax by 1, evaluates u(n) for corresponding values of n using the Expr(u,n) formula and ListOfInitTerms, and returns the results as a list.

$$\overline{\operatorname{seqn}\left(\frac{1}{n^2}, 6\right)} \qquad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

 $seqn(Expr(n[, nMax[, CeilingValue]]) \Rightarrow list$

Generates a list of terms for a non-recursive sequence u(n)=Expr(n) as follows: Increments n from 1 through nMax by 1, evaluates u(n) for corresponding values of n using the Expr(n) formula, and returns the results as a list.

If *nMax* is missing, *nMax* is set to 2500

If nMax=0, nMax is set to 2500

Note: seqn() calls seqGen() with n0=1 and nstep =1

series()

series(Expr1, Var, Order[, Point]**)** \Rightarrow expression

series(Expr1, Var, Order[, Point]) | Var>Point ⇒ expression

series(Expr1, Var, Order[, Point]) | Var<Point ⇒ expression

Returns a generalized truncated power series representation of ExprI expanded about Point through degree Order. Order can be any rational number. The resulting powers of (Var-Point) can include negative and/or fractional exponents. The coefficients of these powers can include logarithms of (Var-Point) and other functions of Var that are dominated by all powers of (Var-Point) having the same exponent sign.

Catalogue > 🗐

$$\begin{aligned} & \text{series} \left(\frac{1 - \cos(x - 1)}{(x - 1)^2}, x, 4, 1 \right) & & \frac{1}{2} - \frac{(x - 1)^2}{24} + \frac{(x - 1)^4}{720} \\ & \text{series} \left(\frac{-1}{e^z}, z_{-1} \right) & & z_{-} - 1 \\ & & \text{series} \left(\left(1 + \frac{1}{n} \right)^n, n, 2, \infty \right) & & \mathbf{e} - \frac{\mathbf{e}}{2 \cdot n} + \frac{11 \cdot \mathbf{e}}{24 \cdot n^2} \end{aligned}$$

series
$$\left(\tan^{3}\left(\frac{1}{x}\right), x, 5\right), x > 0$$
 $\frac{\pi}{2} - x + \frac{x^{3}}{3} - \frac{x^{5}}{5}$
series $\left(\int \frac{\sin(x)}{x} dx, x, 6\right)$ $x - \frac{x^{3}}{18} + \frac{x^{5}}{600}$
series $\left(\int_{0}^{x} \sin(x \cdot \sin(t)) dt, x, 7\right)$ $\frac{x^{3}}{2} - \frac{x^{5}}{24} - \frac{29 \cdot x^{7}}{720}$

series
$$((1+\mathbf{e}^x)^2, x, 2, 1)$$

 $(\mathbf{e}+1)^2+2 \cdot \mathbf{e} \cdot (\mathbf{e}+1) \cdot (x-1)+\mathbf{e} \cdot (2 \cdot \mathbf{e}+1) \cdot (x-1)^2$

series()

Point defaults to 0. *Point* can be ∞ or $-\infty$, in which cases the expansion is through degree Order in 1/(Var - Point).

series(...) returns "series(...)" if it is unable to determine such a representation, such as for essential singularities such as sin(1/z) at z=0. $e^{-1/z}$ at z=0. or e^z at z = ∞ or $-\infty$.

If the series or one of its derivatives has a iump discontinuity at *Point*, the result is likely to contain sub-expressions of the form sign(...) or abs(...) for a real expansion variable or (-1)floor(...angle(...)...) for a complex expansion variable, which is one ending with "_". If you intend to use the series only for values on one side of *Point*, then append the appropriate one of "|Var>" Point", "| Var < Point", "| " $Var \ge Point$ ", or "Var < Point" to obtain a simpler result.

series() can provide symbolic approximations to indefinite integrals and definite integrals for which symbolic solutions otherwise can't be obtained.

series() distributes over 1st-argument lists and matrices.

series() is a generalized version of taylor().

As illustrated by the last example to the right, the display routines downstream of the result produced by series(...) might rearrange terms so that the dominant term is not the leftmost one.

Note: See also dominantTerm(), page 55.

setMode()

Catalogue > 🕮

setMode(modeNameInteger, settingInteger) $\Rightarrow integer$ $setMode(list) \Rightarrow integer\ list$

Valid only within a function or program.

Display approximate value of π using the default setting for Display Digits, and then display π with a setting of Fix 2. Check to see that the default is restored after the program executes.

setMode(modeNameInteger, settingInteger) temporarily sets mode modeNameInteger to the new setting settingInteger, and returns an integer corresponding to the original setting of that mode. The change is limited to the duration of the program/function's execution.

modeNameInteger specifies which mode you want to set. It must be one of the mode integers from the table below.

settingInteger specifies the new setting for the mode. It must be one of the setting integers listed below for the specific mode you are setting.

setMode(*list*) lets you change multiple settings. *list* contains pairs of mode integers and setting integers. **setMode**(*list*) returns a similar list whose integer pairs represent the original modes and settings.

If you have saved all mode settings with $getMode(0) \rightarrow var$, you can use setMode(var) to restore those settings until the function or program exits. See getMode(), page 82.

Note: The current mode settings are passed to called subroutines. If any subroutine changes a mode setting, the mode change will be lost when control returns to the calling routine.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define prog1()=Prgm	Done
Disp approx	
setMode(1,1	16)
Disp approx	$\chi(\pi)$
EndPrgm	
prog1()	
	3.14159
	3.14
	Done

Mode Name	Mode Integer	Setting Integers
Display Digits	1	1=Float, 2=Float1, 3=Float2, 4=Float3, 5=Float4, 6=Float5, 7=Float6, 8=Float7, 9=Float8, 10=Float9, 11=Float10, 12=Float11, 13=Float12, 14=Fix0, 15=Fix1, 16=Fix2, 17=Fix3, 18=Fix4, 19=Fix5, 20=Fix6, 21=Fix7, 22=Fix8, 23=Fix9, 24=Fix10, 25=Fix11, 26=Fix12

Mode Name	Mode Integer	Setting Integers
Angle	2	1=Radian, 2=Degree, 3=Gradian
Exponential Format	3	1=Normal, 2=Scientific, 3=Engineering
Real or Complex	4	1=Real, 2=Rectangular, 3=Polar
Auto or Approx.	5	1=Auto, 2=Approximate, 3=Exact
Vector Format	6	1=Rectangular, 2=Cylindrical, 3=Spherical
Base	7	1=Decimal, 2=Hex, 3=Binary
Unit system	8	1=SI, 2=Eng/US

shift() Catalogue > 🗐

 $shift(Integer1[,\#ofShifts]) \Rightarrow integer$

Shifts the bits in a binary integer. You can enter *Integer 1* in any number base; it is converted automatically to a signed, 64-bit binary form. If the magnitude of *Integer1* is too large for this form, a symmetric modulo operation brings it within the range. For more information, see ▶Base2, page 17.

If #ofShifts is positive, the shift is to the left. If #ofShifts is negative, the shift is to the right. The default is -1 (shift right one bit).

In a right shift, the rightmost bit is dropped and 0 or 1 is inserted to match the leftmost bit. In a left shift, the leftmost bit is dropped and 0 is inserted as the rightmost hit.

For example, in a right shift:

Each bit shifts right.

0b0000000000000111101011000011010

Inserts 0 if leftmost bit is 0, or 1 if leftmost bit is 1.

produces:

In Bin base mode:

shift(0b1111010110000110101)		
	0b111101011000011010	
shift(256,1)	0b1000000000	

In Hex base mode:

shift(0h78E)	0h3C7
shift(0h78E,-2)	0h1E3
shift(0h78E,2)	0h1E38

Important: To enter a binary or hexadecimal number, always use the 0b or Oh prefix (zero, not the letter O).

0b00000000000000111101011000011010

The result is displayed according to the Base mode. Leading zeros are not shown.

$$shift(List1[,\#ofShifts]) \Rightarrow list$$

Returns a copy of List1 shifted right or left by #ofShifts elements. Does not alter List1.

If #ofShifts is positive, the shift is to the left. If #ofShifts is negative, the shift is to the right. The default is -1 (shift right one element).

Elements introduced at the beginning or end of *list* by the shift are set to the symbol "undef".

$$shift(String1[,\#ofShifts]) \Rightarrow string$$

Returns a copy of *String1* shifted right or left by *#ofShifts* characters. Does not alter *String1*.

If #ofShifts is positive, the shift is to the left. If #ofShifts is negative, the shift is to the right. The default is -1 (shift right one character).

Characters introduced at the beginning or end of *string* by the shift are set to a space.

In Dec base mode:

shift({1,2,3,4})	{undef,1,2,3}
shift({1,2,3,4},-2)	$\{undef,undef,1,2\}$
shift({1,2,3,4},2)	${3,4,undef,undef}$

shift("abcd")	" abc"
shift("abcd",-2)	" ab"
shift("abcd",1)	"bcd "

sign() Catalogue > [3]

 $sign(Expr1) \Rightarrow expression$

 $sign(List1) \Rightarrow list$ $sign(Matrix 1) \Rightarrow matrix$

For real and complex Expr1, returns Expr1/abs(Expr1) when $Expr1 \neq 0$.

Returns 1 if ExprI is positive. Returns -1 if ExprI is negative.

sign(0) represents the unit circle in the complex domain.

For a list or matrix, returns the signs of all the elements.

sign(-3.2)	-1
sign({2,3,4,-5})	$\{1,1,1,-1\}$
sign(1+ x)	1

If complex format mode is Real:

sign([-3	0	3])	[-	-1	± 1	1]

simult()

Catalogue > 🕮

 $simult(coeffMatrix, constVector[, Tol]) \Rightarrow$ matrix

Returns a column vector that contains the solutions to a system of linear equations.

Note: See also linSolve(), page 101.

coeffMatrix must be a square matrix that contains the coefficients of the equations.

constVector must have the same number of rows (same dimension) as coeffMatrix and contain the constants.

Optionally, any matrix element is treated as zero if its absolute value is less than Tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, Tol is ignored.

- If you set the **Auto or Approximate** mode to Approximate, computations are done using floating-point arithmetic.
- If *Tol* is omitted or not used, the default tolerance is calculated as: 5E-14 •max(dim(coeffMatrix)) •rowNorm(*coeffMatrix*)

 $simult(coeffMatrix, constMatrix[, Tol]) \Rightarrow$ matrix

Solves multiple systems of linear equations, where each system has the same equation coefficients but different constants.

Each column in constMatrix must contain the constants for a system of equations. Each column in the resulting matrix contains the solution for the corresponding system.

Solve for x and y:

$$x + 2y = 1$$

$$3x + 4y = -1$$

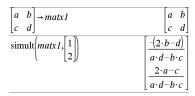
$$\begin{array}{c|c} simult \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} & \begin{bmatrix} -3 \\ 2 \end{bmatrix} \end{array}$$

The solution is x=-3 and y=2.

Solve:

ax + bv = 1

cx + dy = 2



Solve:

x + 2y = 1

3x + 4y = -1

x + 2y = 2

3x + 4y = -3

$$simult \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix} \qquad \begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$$

For the first system, x=-3 and y=2. For the second system, x=-7 and y=9/2.

▶ sin Catalogue > 🕮

Expr \triangleright sin

Note: You can insert this operator from the computer keyboard by typing @>sin.

$$(\cos(x))^2 \triangleright \sin \qquad 1 - (\sin(x))^2$$

Represents *Expr* in terms of sine. This is a display conversion operator. It can be used only at the end of the entry line.

► sin reduces all powers of cos(...) modulo 1—sin(...)^2 so that any remaining powers of sin(...) have exponents in the range (0, 2). Thus, the result will be free of cos(...) if and only if cos(...) occurs in the given expression only to even powers.

Note: This conversion operator is not supported in Degree or Gradian Angle modes. Before using it, make sure that the Angle mode is set to Radians and that *Expr* does not contain explicit references to degree or gradian angles.

sin()	trig key
-------	----------

 $sin(Expr1) \Rightarrow expression$

 $sin(List1) \Rightarrow list$

sin(*Expr1*) returns the sine of the argument as an expression.

sin(List1) returns a list of the sines of all elements in List1.

Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode. You can use °, g, or r to override the angle mode setting temporarily.

In Degree angle mode:

$\frac{1}{\sin\left(\frac{\pi}{r}\right)}$	$\sqrt{2}$
4 }	2
sin(45)	$\sqrt{2}$
	2
sin({0.60.90})	

$$\sin(\{0,60,90\})$$
 $\left\{0,\frac{\sqrt{3}}{2},1\right\}$

In Gradian angle mode:

sin(50)	$\sqrt{2}$
	2

In Radian angle mode:

$\sin\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
sin(45°)	$\frac{\sqrt{2}}{2}$

 $sin(squareMatrix 1) \Rightarrow squareMatrix$

In Radian angle mode:

sin()



Returns the matrix sine of *squareMatrix1*. This is not the same as calculating the sine of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

$\sin\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	5 2	$\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$		
Æθ	-2 [1]]	-0.04542	-0.031999
		-0.045492	0.04342	-0.020274
	Į	-0.048739	-0.00523	0.961051

sin-1()

trig key

 $sin-1(Expr1) \Rightarrow expression$

 $sin-1(List1) \Rightarrow list$

sin-1(*Expr1*) returns the angle whose sine is *Expr1* as an expression.

sin-1(List1) returns a list of the inverse sines of each element of List1.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arcsin (...).

 $sin-1(squareMatrix 1) \Rightarrow squareMatrix$

Returns the matrix inverse sine of squareMatrix1. This is not the same as calculating the inverse sine of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Degree angle mode:

sin-1(1)

90

In Gradian angle mode:

sin-1(1) 100

In Radian angle mode:

 $\sin^{-1}(\{0,0.2,0.5\})$ {0,0.201358,0.523599}

In Radian angle mode and Rectangular complex format mode:

$$\begin{array}{l} \sin^4 \begin{pmatrix} 1 & 5 \\ 4 & 2 \end{pmatrix} \\ \begin{bmatrix} -0.174533 - 0.12198 \cdot \boldsymbol{i} & 1.74533 - 2.35591 \cdot \boldsymbol{i} \\ 1.39626 - 1.88473 \cdot \boldsymbol{i} & 0.174533 - 0.593162 \cdot \boldsymbol{i} \end{bmatrix} \end{array}$$

sinh()

Catalogue > 🗐

 $sinh(Expr1) \Rightarrow expression$

 $sinh(List1) \Rightarrow list$

sinh (Expr1) returns the hyperbolic sine of the argument as an expression.

sinh(1.2)1.50946 $sinh(\{0,1.2,3.\})$ {0,1.50946,10.0179}

sinh (List I) returns a list of the hyperbolic sines of each element of List I.

 $sinh(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix hyperbolic sine of *squareMatrix1*. This is not the same as calculating the hyperbolic sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$$sinh \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \\
= \begin{bmatrix} 360.954 & 305.708 & 239.604 \\ 352.912 & 233.495 & 193.564 \\ 298.632 & 154.599 & 140.251 \end{bmatrix}$$

sinh-1() Catalogue > [3]

 $sinh-1(Expr1) \Rightarrow expression$

 $sinh-1(List1) \Rightarrow list$

sinh-1(Expr1) returns the inverse hyperbolic sine of the argument as an expression.

sinh-1(*List1*) returns a list of the inverse hyperbolic sines of each element of *List1*.

Note: You can insert this function from the keyboard by typing arcsinh (...).

 $sinh-1(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix inverse hyperbolic sine of *squareMatrix1*. This is not the same as calculating the inverse hyperbolic sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

$\begin{array}{ccc} \sinh^{3}(0) & 0 \\ \sinh^{3}(\left\{0,2.1,3\right\}) & \left\{0,1.48748,\sinh^{3}(3)\right\} \end{array}$

In Radian angle mode:

SinReg

Catalogue > 🗐

SinReg X, Y[, [Iterations],[Period][, Category, Include]]

Computes the sinusoidal regression on lists X and Y. A summary of results is stored in the *stat.results* variable. (See page 174.)

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Iterations is a value that specifies the maximum number of times (1 through 16) a solution will be attempted. If omitted, 8 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

Period specifies an estimated period. If omitted, the difference between values in X should be equal and in sequential order. If you specify *Period*, the differences between x values can be unequal.

Category is a list of category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

The output of **SinReg** is always in radians, regardless of the angle mode setting.

For information on the effect of empty elements in a list, see "Empty (Void) Elements," page 248.

Output variable	Description
stat.RegEqn	Regression Equation: a•sin(bx+c)+d
stat.a, stat.b, stat.c, stat.d	Regression coefficients
stat.Resid	Residuals from the regression
stat.XReg	List of data points in the modified X $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$, and $Include$ $Categories$
stat.YReg	List of data points in the modified Y $List$ actually used in the regression based on restrictions of $Freq$, $Category$ $List$, and $Include$ $Categories$
stat.FreqReg	List of frequencies corresponding to stat. XReg and stat. YReg

 $solve(Equation, Var) \Rightarrow Boolean$ expression

solve(Equation, Var=Guess) ⇒ Boolean expression

solve(Inequality, Var**)** \Rightarrow Boolean expression

Returns candidate real solutions of an equation or an inequality for *Var*. The goal is to return candidates for all solutions. However, there might be equations or inequalities for which the number of solutions is infinite.

Solution candidates might not be real finite solutions for some combinations of values for undefined variables.

For the Auto setting of the Auto or Approximate mode, the goal is to produce exact solutions when they are concise, and supplemented by iterative searches with approximate arithmetic when exact solutions are impractical.

Due to default cancellation of the greatest common divisor from the numerator and denominator of ratios, solutions might be solutions only in the limit from one or both sides.

For inequalities of types \geq , \leq , <, or >, explicit solutions are unlikely unless the inequality is linear and contains only Var.

For the Exact mode, portions that cannot be solved are returned as an implicit equation or inequality.

Use the constraint ("|") operator to restrict the solution interval and/or other variables that occur in the equation or inequality. When you find a solution in one interval, you can use the inequality operators to exclude that interval from subsequent searches.

$$\frac{\text{solve}(a \cdot x^2 + b \cdot x + c = 0, x)}{x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a} \text{ or } x = \frac{-(\sqrt{b^2 - 4 \cdot a \cdot c} + b)}{2 \cdot a}$$

Ans|a=1 and b=1 and c=1

$$x = \frac{-1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ or } x = \frac{-1}{2} - \frac{\sqrt{3}}{2} \cdot i$$

solve
$$((x-a) \cdot \mathbf{e}^x = x \cdot (x-a), x)$$

 $x=a \text{ or } x=-0.567143$

$$(x+1)\cdot\frac{x-1}{x-1}+x-3$$

solve
$$(5 \cdot x - 2 \ge 2 \cdot x, x)$$
 $x \ge \frac{2}{3}$

exact(solve(
$$(x-a) \cdot e^x = x \cdot (x-a), x$$
))
 $e^x + x = 0 \text{ or } x = a$

In Radian angle mode:

solve
$$\left(\tan(x) = \frac{1}{x}, x\right) | x > 0 \text{ and } x < 1$$

 $x = 0.860334$

solve()

Catalogue > 23

false is returned when no real solutions are found, true is returned if solve() can determine that any finite real value of *Var* satisfies the equation or inequality.

Since solve() always returns a Boolean result, you can use "and," "or," and "not" to combine results from solve() with each other or with other Boolean expressions.

Solutions might contain a unique new undefined constant of the form **n***j* with *j* being an integer in the interval 1–255. Such variables designate an arbitrary integer.

In Real mode, fractional powers having odd denominators denote only the real branch. Otherwise, multiple branched expressions such as fractional powers, logarithms, and inverse trigonometric functions denote only the principal branch. Consequently, solve() produces only solutions corresponding to that one real or principal branch.

Note: See also cSolve(), cZeros(), nSolve(), and zeros().

solve(Eqn1 and Eqn2[and ...], VarOrGuess1, VarOrGuess2[, ...]) ⇒ Boolean expression

solve(SystemOfEgns, VarOrGuess1, VarOrGuess2[, ...]) ⇒ Boolean expression

solve({*Eqn1*, *Eqn2* [,...]} {VarOrGuess1, VarOrGuess2 [, ...]}) ⇒ Boolean expression

Returns candidate real solutions to the simultaneous algebraic equations, where each VarOrGuess specifies a variable that vou want to solve for.

solve(x=x+1,x)	false
solve(x=x,x)	true

$$2 \cdot x - 1 \le 1$$
 and solve $\left(x^2 \ne 9, x\right)$ $x \ne -3$ and $x \le 1$

In Radian angle mode:

$$solve(sin(x)=0,x) x=n1\cdot\pi$$

$$solve \begin{cases} \frac{1}{x^3} = -1, x \end{cases}$$

$$solve (\sqrt{x} = -2, x)$$

$$solve (-\sqrt{x} = -2, x)$$

$$r = 4$$

solve
$$\left(y=x^2-2 \text{ and } x+2 \cdot y=-1, \left\{x,y\right\}\right)$$

 $x=\frac{-3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$

You can separate the equations with the and operator, or you can enter a *SystemOfEqns* using a template from the Catalogue. The number of *VarOrGuess* arguments must match the number of equations. Optionally, you can specify an initial guess for a variable. Each *VarOrGuess* must have the form:

variable

– or –

variable = real or non-real number

For example, x is valid and so is x=3.

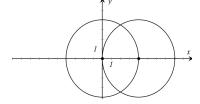
If all of the equations are polynomials and if you do NOT specify any initial guesses, solve() uses the lexical Gröbner/Buchberger elimination method to attempt to determine all real solutions.

For example, suppose you have a circle of radius r at the origin and another circle of radius r centred where the first circle crosses the positive x-axis. Use **solve()** to find the intersections.

As illustrated by r in the example to the right, simultaneous polynomial equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

You can also (or instead) include solution variables that do not appear in the equations. For example, you can include z as a solution variable to extend the previous example to two parallel intersecting cylinders of radius r.

The cylinder solutions illustrate how families of solutions might contain arbitrary constants of the form ck, where k is an integer suffix from 1 through 255.



solve
$$\left\{x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \left\{x,y\right\}\right\}$$

 $x=\frac{r}{2}$ and $y=\frac{\sqrt{3} \cdot r}{2}$ or $x=\frac{r}{2}$ and $y=\frac{-\sqrt{3} \cdot r}{2}$

solve
$$\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \left\{x,y,z\right\}\right)$$

 $x=\frac{r}{2}$ and $y=\frac{\sqrt{3} \cdot r}{2}$ and $z=c1$ or $x=\frac{r}{2}$ and $y=\frac{r}{2}$

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or varOrGuess list.

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in the solution variables, solve() uses Gaussian elimination to attempt to determine all real solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables. solve() determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

Each solution variable starts at its guessed value if there is one; otherwise, it starts at 0.0.

Use guesses to seek additional solutions one by one. For convergence, a guess may have to be rather close to a solution.

solve
$$\left(x + e^z \cdot y = 1 \text{ and } x - y = \sin(z), \left\{x, y\right\}\right)$$

$$x = \frac{e^z \cdot \sin(z) + 1}{e^z + 1} \text{ and } y = \frac{-\left(\sin(z) - 1\right)}{e^z + 1}$$

solve
$$\left(\mathbf{e}^{z} \cdot y=1 \text{ and } -y=\sin(z), \{y,z\}\right)$$

y=2.812\vec{\vec{e}}^{10} \text{ and } z=21.9911 \text{ or } y=0.001871\vec{\vec{e}}

To see the entire result, press \triangle and then use ◀ and ▶ to move the cursor.

solve
$$\left(e^{z} \cdot y = 1 \text{ and } -y = \sin(z), \left\{y, z = 2 \cdot \pi\right\}\right)$$

 $y = 0.001871 \text{ and } z = 6.28131$

SortA **SortA** *List1*[, *List2*] [, *List3*]... **SortA** Vector1[, Vector2] [, Vector3]...

Sorts the elements of the first argument in ascending order.

If you include additional arguments, sorts the elements of each so that their new positions match the new positions of the elements in the first argument.

All arguments must be names of lists or vectors. All arguments must have equal dimensions.

	•	•
$\{2,1,4,3\} \rightarrow list1$	{2,1	,4,3}
SortA list1		Done
list1	{1,2	2,3,4}
$\{4,3,2,1\} \rightarrow list2$	{4,3	5,2,1}
SortA list2,list1		Done
list2	{1,2	2,3,4}
list1	{4,3	5,2,1}

Catalogue > 🕮

Empty (void) elements within the first argument move to the bottom. For more information on empty elements, see page 248.

SortD Catalogue > 23

SortD *List1*[, *List2*][, *List3*]... **SortD** Vector1[,Vector2][,Vector3]...

Identical to SortA, except SortD sorts the elements in descending order.

Empty (void) elements within the first argument move to the bottom. For more information on empty elements, see page 248.

$\{2,1,4,3\} \rightarrow list1$	{2,1,4,3}
$\{1,2,3,4\} \rightarrow list2$	{1,2,3,4}
SortD list1,list2	Done
list1	{4,3,2,1}
list2	{3,4,1,2}

➤ Sphere

Vector ▶ Sphere

Note: You can insert this operator from the computer keyboard by typing @>Sphere.

Displays the row or column vector in spherical form $[\rho \angle \theta \angle \phi]$.

Vector must be of dimension 3 and can be either a row or a column vector.

Note: ▶ **Sphere** is a display-format instruction, not a conversion function. You can use it only at the end of an entry line.

Catalogue > 🕮

Note: To force an approximate result,

Handheld: Press ctrl enter. Windows®: Press Ctrl+Enter. Macintosh®: Press #+Enter. iPad®: Hold enter, and select ≈ .

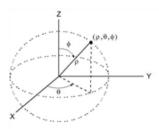
Note: To force an approximate result,

Handheld: Press ctrl enter. Windows®: Press Ctrl+Enter. Macintosh®: Press #+Enter. iPad®: Hold enter, and select ≈ .

$$\left[2 \ \angle \frac{\pi}{4} \ 3\right] \triangleright \text{Sphere}$$
 $\left[3.60555 \ \angle 0.785398 \ \angle 0.588003\right]$

Press enter





sqrt() $sqrt(Expr1) \Rightarrow expression$

 $sqrt(List1) \Rightarrow list$

Returns the square root of the argument.

For a list, returns the square roots of all the elements in List1.

Note: See also Square root template, page 1.

Catalogue > 😰

 $\sqrt{4}$ $\sqrt{9,a,4}$ $\{3, \sqrt{a}, 2\}$

stat.results

Displays results from a statistics calculation.

The results are displayed as a set of namevalue pairs. The specific names shown are dependent on the most recently evaluated statistics function or command.

You can copy a name or value and paste it into other locations.

Note: Avoid defining variables that use the same names as those used for statistical analysis. In some cases, an error condition could occur. Variable names used for statistical analysis are listed in the table below.

$xlist:=\{1,2,3,4,5\}$	{1,2,3,4,5}	
ylist:={4,8,11,14,17}	{4,8,11,14,17}	

LinRegMx xlist,ylist,1: stat.results

	"Linear Regression (mx+b)"	
	"RegEqn"	"m*x+b"
	"m"	3.2
	"b"	1.2
	"r²"	0.996109
	"r"	0.998053
	"Resid"	"{}"

stat.values	["Linear Regression (mx+b)"]
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4,0.4,0.2,0.,-0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat. ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat. Upper Pred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.F	stat.n	stat. Σ x	stat.X
stat.b9	stat.FBlock	Stat. $\hat{\pmb{p}}$	$stat.\Sigmax^2$	stat.X1
stat.b10	stat.Fcol	stat. \hat{p} 1	stat. Σ xy	stat.x2
stat.bList	stat.FInteract	stat. p̂ 2	stat. Σ y	stat.XDiff
$\text{stat.} \chi^2$	stat.FreqReg	stat. $\hat{\pmb{p}}$ Diff	$stat.\Sigmay^2$	stat.XList
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. y
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. ŷ
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat. ŷ List
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
				stat. I NEg

stat.CUpperList stat.ME stat.O1Y stat.SS

stat.d stat.MedianX

Note: Each time the Lists & Spreadsheet application calculates statistical results, it copies the "stat." group variables to a "stat#." group, where # is a number that is incremented automatically. This lets you maintain previous results while performing multiple calculations.

stat.values Catalogue > 🕮

stat.values

See the stat.results example.

Displays a matrix of the values calculated for the most recently evaluated statistics function or command.

Unlike stat.results, stat.values omits the names associated with the values.

You can copy a value and paste it into other locations.

stDevPop()

Catalogue > 23

 $stDevPop(List [, freqList]) \Rightarrow expression$

Returns the population standard deviation of the elements in List.

Each *freaList* element counts the number of consecutive occurrences of the corresponding element in *List*.

Note:List must have at least two elements. Empty (void) elements are ignored. For more information on empty elements, see page 248.

 $stDevPop(Matrix 1[, freqMatrix]) \Rightarrow$ matrix

Returns a row vector of the population standard deviations of the columns in Matrix 1.

Each freaMatrix element counts the number of consecutive occurrences of the corresponding element in *Matrix 1*.

In Radian angle and auto modes:

$$\frac{\text{stDevPop}(\left\{a,b,c\right\})}{\frac{\sqrt{2\cdot\left(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2\right)}}{3}}$$

$$\frac{1}{\text{stDevPop}(\left\{1,2,5,-6,3,-2\right\})} \qquad \frac{\sqrt{465}}{6}$$

$$\frac{1}{\text{stDevPop}(\left\{1,3,2,5,-6,4\right\},\left\{3,2,5\right\})}{3} \qquad 4.11107$$



Note: *Matrix I* must have at least two rows. Empty (void) elements are ignored. For more information on empty elements, see page 248.

stDevSamp()

Catalogue > 23

 $stDevSamp(List[, freqList]) \Rightarrow expression$

Returns the sample standard deviation of the elements in *List*.

Each *freqList* element counts the number of consecutive occurrences of the corresponding element in *List*.

Note: *List* must have at least two elements. Empty (void) elements are ignored. For more information on empty elements, see page 248.

 $stDevSamp(Matrix 1[, freqMatrix]) \Rightarrow matrix$

Returns a row vector of the sample standard deviations of the columns in *Matrix I*.

Each *freqMatrix* element counts the number of consecutive occurrences of the corresponding element in *Matrix1*.

Note: Matrix I must have at least two rows. Empty (void) elements are ignored. For more information on empty elements, see page 248.

$$\frac{\text{stDevSamp}(\{a,b,c\})}{\sqrt{3 \cdot \left(a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2\right)}} \frac{3}{3}$$

$$\frac{3}{\text{stDevSamp}(\{1,2,5,-6,3,-2\})} \frac{\sqrt{62}}{2}$$

$$\frac{3}{\text{stDevSamp}(\{1,3,2,5,-6,4\},\{3,2,5\})}$$

$$\frac{4.33345}{4.33345}$$

$$stDevSamp \begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \begin{bmatrix} 4 & \sqrt{13} & 2 \end{bmatrix} \\
stDevSamp \begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix} \\
[2.7005 & 5.44695]$$

Catalogue > 🕮 Stop Stop i = 00 Define *prog1*()=Prgm Programming command: Terminates the Done For i, 1, 10, 1program. If i=5Stop is not allowed in functions. Stop EndFor EndPrgm prog1() Done 5

Catalogue > 📳

Stop

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Store

See \rightarrow (store), page 230.

string()		Catalogue > 🕎
$string(Expr) \Rightarrow string$	string(1.2345)	"1.2345"
Simplifies $Expr$ and returns the result as a	string(1+2)	"3"
character string.	string($\cos(x) + \sqrt{3}$)	$"\cos(x) + \sqrt{3}$

subMat()		Catalogue > 🕎
subMat($Matrix 1[$, $startRow][$, $startCol][$, $endRow][$, $endCol]$) $\Rightarrow matrix$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow m1$	1 2 3 4 5 6
Returns the specified submatrix of ${\it Matrix 1}$.	$\frac{[7 \ 8 \ 9]}{\text{subMat}(m1,2,1,3,2)}$	[7 8 9]
Defaults: startRow=1, startCol=1,	subMat(<i>m1</i> ,2,1,3,2)	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
endRow=last row, endCol=last column.	$\operatorname{subMat}(m1,2,2)$	[5 6] [8 9]

See Σ (), page 221. Sum (Sigma)

sum()	Ca	talogue > 🏥
$sum(List[, Start[, End]]) \Rightarrow expression$	sum({1,2,3,4,5})	15
Returns the sum of all elements in $List$.	$\operatorname{sum}(\{a,2\cdot a,3\cdot a\})$	6· <i>a</i>
Start and End are optional. They specify a	$\operatorname{sum}(\operatorname{seq}(n,n,1,10))$	55
range of elements.	$sum({1,3,5,7,9},3)$	21
Any void argument produces a void result. Empty (void) elements in $List$ are ignored. For more information on empty elements, see page 248.		

sum()

 $sum(Matrix 1[, Start[, End]]) \Rightarrow matrix$

Returns a row vector containing the sums of all elements in the columns in *Matrix 1*.

Start and *End* are optional. They specify a range of rows.

Any void argument produces a void result. Empty (void) elements in *Matrix I* are ignored. For more information on empty elements, see page 248.

sum	1	2	3 6	[5 7 9]
J	$\lfloor 4 \rfloor$	5	6]/	
	1	2	3 6 9	[12 15 18]
sum	4	5	6	
/	7	8	9∬	
	1	2	3 6 9,2,3	[11 13 15]
sum	4	5	6 ,2,3	
/	7	8	9]	

sumIf({1,2,3,4},2<?<5,{10,20,30,40})

sumIf()

 $sumIf(List,Criteria[,SumList]) \Rightarrow value$

Returns the accumulated sum of all elements in *List* that meet the specified *Criteria*. Optionally, you can specify an alternate list, *sumList*, to supply the elements to accumulate.

List can be an expression, list, or matrix. SumList, if specified, must have the same dimension(s) as List.

Criteria can be:

- A value, expression, or string. For example, 34 accumulates only those elements in *List* that simplify to the value 34.
- A Boolean expression containing the symbol ? as a place holder for each element. For example, ?<10 accumulates only those elements in List that are less than 10.

When a *List* element meets the *Criteria*, the element is added to the accumulating sum. If you include *sumList*, the corresponding element from *sumList* is added to the sum instead.

Within the Lists & Spreadsheet application, you can use a range of cells in place of *List* and *sumList*.

sumIf($\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5\}$) $e^{+\pi+7}$

Catalogue > 🕮

Catalogue > 🕮

70

sumIf()

Catalogue > [13]

Empty (void) elements are ignored. For more information on empty elements, see page 248.

Note: See also countif(), page 35.

sumSeq()

See Σ (), page 221.

system()

system(Eqn1[, Eqn2[, Eqn3[, ...]]])

system(*Expr1*[, *Expr2*[, *Expr3*[, ...]]])

Returns a system of equations, formatted as a list. You can also create a system by using a template.

Note: See also System of equations, page 3.

T

Catalogue > 23

Catalogue > 🕮

solve $\begin{cases} x+y=0\\ x-y=8 \end{cases}$, x,y	x=4 and $y=-4$
---	----------------

Matrix |**T**⇒matrix

T (transpose)

Returns the complex conjugate transpose of Matrix 1.

Note: You can insert this operator from the computer keyboard by typing @t.

$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}_{T}$	[1 4 7]
4 5 6 T	2 5 8
[7 8 9]	3 6 9
$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{T}$	$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$
$\begin{bmatrix} c & d \end{bmatrix}$	$\begin{bmatrix} b & d \end{bmatrix}$
$\begin{bmatrix} 1+i & 2+i \end{bmatrix}$ ^T	$\begin{bmatrix} 1-i & 3-i \end{bmatrix}$
$\begin{bmatrix} 3+i & 4+i \end{bmatrix}$	$\begin{bmatrix} 2-i & 4-i \end{bmatrix}$

tan()

trig kev

 $tan(Expr1) \Rightarrow expression$

 $tan(List1) \Rightarrow list$

tan(*Expr1*) returns the tangent of the argument as an expression.

tan(List1) returns a list of the tangents of all elements in *List1*.

In Degree angle mode:

$\tan\left(\frac{\pi}{4}r\right)$	1
tan(45)	1
$\tan(\{0,60,90\})$	$\{0,\sqrt{3},\text{undef}\}$

tan()



Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode. You can use °, G or ^r to override the angle mode setting temporarily.

In Gradian angle mode:

$\tan\left(\frac{\pi}{4}^{r}\right)$	1
tan(50)	1
$\tan(\{0,50,100\})$	$\{0,1,$ undef $\}$

In Radian angle mode:

$\tan\left(\frac{\pi}{4}\right)$	1
tan(45°)	1
$\tan\left\{\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right\}$	$\{0,\sqrt{3},0,1\}$

$tan(squareMatrix I) \Rightarrow squareMatrix$ In Ra

Returns the matrix tangent of squareMatrix I. This is not the same as calculating the tangent of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalisable. The result always contains floating-point numbers.

In Radian angle mode:

$$\tan\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan-1()

^{trig} key

 $tan^{-1}(Expr1) \Rightarrow expression$

 $tan^{-1}(List 1) \Rightarrow list$

 $tan^{-1}(Expr1)$ returns the angle whose tangent is Expr1 as an expression.

 $tan^{-1}(List 1)$ returns a list of the inverse tangents of each element of List 1.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

Note: You can insert this function from the keyboard by typing arctan (...).

 $tan^{-1}(squareMatrix 1) \Rightarrow squareMatrix$

In Degree angle mode:

In Gradian angle mode:

In Radian angle mode:

$$tan^{-1}(\{0,0.2,0.5\}) = \{0,0.197396,0.463648\}$$

In Radian angle mode:

tan-1()



Returns the matrix inverse tangent of squareMatrix1. This is not the same as calculating the inverse tangent of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalisable. The result always contains floating-point numbers.

$tan^{-1}\begin{bmatrix} 1\\4\\6 \end{bmatrix}$	$\begin{bmatrix} 5 & 3 \\ 2 & 1 \\ -2 & 1 \end{bmatrix}$		
	-0.083658	1.26629	0.62263
	0.748539	0.630015	-0.070012
	1.68608	$^{-}1.18244$	0.455126

tangentLine()

Catalogue > 🕮

tangentLine

(Expr1, Var, Point) ⇒ expression

tangentLine

 $(Expr1, Var=Point) \Rightarrow expression$

Returns the tangent line to the curve represented by Expr1 at the point specified in Var=Point.

Make sure that the independent variable is not defined. For example, If f1(x):=5 and x:=3, then tangentLine(f1(x),x,2) returns "false."

$tangentLine(x^2,x,1)$	2· <i>x</i> -1
$\frac{1}{\text{tangentLine}((x-3)^2-4, x=3)}$	-4
$\frac{1}{\text{tangentLine}\left(x^{\frac{1}{3}}, x=0\right)}$	<i>x</i> =0
$\frac{1}{\text{tangentLine}(\sqrt{x^2-4}, x=2)}$	undef
$x:=3: tangentLine(x^2,x,1)$	5

tanh()

Catalogue > 🕮

 $tanh(Expr1) \Rightarrow expression$

 $tanh(List1) \Rightarrow list$

tanh(Expr1) returns the hyperbolic tangent of the argument as an expression.

tanh(List1) returns a list of the hyperbolic tangents of each element of List1.

 $tanh(squareMatrix1) \Rightarrow squareMatrix$

Returns the matrix hyperbolic tangent of squareMatrix1. This is not the same as calculating the hyperbolic tangent of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalisable. The result always contains floating-point numbers.

tanh(1.2) 0.833655 tanh({0,1}) (0,tanh(1)

In Radian angle mode:

$$\tanh \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

 $tanh^{-1}(Expr1) \Rightarrow expression$

 $tanh^{-1}(List1) \Rightarrow list$

tanh⁻¹(Expr1) returns the inverse hyperbolic tangent of the argument as an expression.

tanh⁻¹(List1) returns a list of the inverse hyperbolic tangents of each element of List1.

Note: You can insert this function from the keyboard by typing arctanh (...).

 $tanh^{-1}(squareMatrix 1) \Rightarrow squareMatrix$

Returns the matrix inverse hyperbolic tangent of *squareMatrix1*. This is not the same as calculating the inverse hyperbolic tangent of each element. For information about the calculation method, refer to cos ().

squareMatrix1 must be diagonalisable. The result always contains floating-point numbers.

In Rectangular complex format:

$$\frac{\tanh^{-1}(0)}{\tanh^{-1}(\{1,2.1,3\})}$$

$$\left\{ undef,0.518046-1.5708 \cdot i, \frac{\ln(2)}{2} - \frac{\pi}{2} \cdot i \right\}$$

In Radian angle mode and Rectangular complex format:

$$tanh^{-1} \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \\
\begin{bmatrix} -0.099353+0.164058 \cdot \mathbf{i} & 0.267834-1.4908 \\ -0.087596-0.725533 \cdot \mathbf{i} & 0.479679-0.94736 \\ 0.511463-2.08316 \cdot \mathbf{i} & -0.878563+1.7901 \end{bmatrix}$$

To see the entire result, press ▲ and then use ◀ and ▶ to move the cursor.

taylor()

taylor(Expr1, Var, Order[, Point])⇒expression

Returns the requested Taylor polynomial. The polynomial includes non-zero terms of integer degrees from zero through Order in (Var minus Point), taylor() returns itself if there is no truncated power series of this order, or if it would require negative or fractional exponents. Use substitution and/or temporary multiplication by a power of (Var minus Point) to determine more general power series.

Point defaults to zero and is the expansion point.

Catalogue > 🕮

$$\frac{\operatorname{taylor}(e^{\sqrt{x}}, x, 2)}{\operatorname{taylor}(e^{\sqrt{x}}, x, 2, 0)} = \frac{\operatorname{taylor}(e^{\sqrt{x}}, x, 2, 0)}{\operatorname{taylor}(e^{\sqrt{x}}, x, 2, 0)} = \frac{\frac{3}{2}}{24} + \frac{\frac{3}{2}}{6} + \frac{x}{2} + \sqrt{x} + 1$$

$$\frac{\operatorname{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3\right)}{\operatorname{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3, 0\right)} = \exp \operatorname{and}\left(\frac{\operatorname{taylor}\left(\frac{x}{x \cdot (x-1)}, x, 4\right)}{x}, x\right) - \frac{x^{3} - x^{2} - x - \frac{1}{x} - 1}{x}$$

tCdf()

Catalogue > 🕮

 $tCdf(lowBound.upBound.df) \Rightarrow number if lowBound$

and upBound are numbers, list if lowBound and upBound are lists

Computes the Student-t distribution probability between lowBound and upBound for the specified degrees of freedom df.

For $P(X \le upBound)$, set $lowBound = -\infty$.

tCollect() Catalogue > 🕮

$tCollect(Expr1) \Rightarrow expression$

Returns an expression in which products and integer powers of sines and cosines are converted to a linear combination of sines and cosines of multiple angles, angle sums and angle differences. The transformation converts trigonometric polynomials into a linear combination of their harmonics.

Sometimes tCollect() will accomplish your goals when the default trigonometric simplification does not. tCollect() tends to reverse transformations done by tExpand(). Sometimes applying tExpand() to a result from tCollect(), or vice versa, in two separate steps simplifies an expression.

t Collect $((\cos(\alpha))^2)$	$\frac{\cos(2\cdot\alpha)+1}{2}$
$tCollect(sin(\alpha) \cdot cos(\beta))$	$\frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{\sin(\alpha-\beta)}$
	2

tExpand()

$tExpand(Expr1) \Rightarrow expression$

Returns an expression in which sines and cosines of integer-multiple angles, angle sums and angle differences are expanded. Because of the identity $(\sin(x))2+(\cos$ (x))2=1, there are many possible equivalent results. Consequently, a result might differ from a result shown in other publications.

Sometimes tExpand() will accomplish your goals when the default trigonometric simplification does not. tExpand() tends to reverse transformations done by tCollect(). Sometimes applying tCollect() to a result from tExpand(), or vice versa, in two separate steps simplifies an expression.

Catalogue > 🕮

tExpand(
$$\sin(3 \cdot \varphi)$$
) $4 \cdot \sin(\varphi) \cdot (\cos(\varphi))^2 - \sin(\varphi)$
tExpand($\cos(\alpha - \beta)$) $\cos(\alpha) \cdot \cos(\beta) + \sin(\alpha) \cdot \sin(\beta)$

Note: Degree-mode scaling by $\pi/180$ interferes with the ability of tExpand() to recognise expandable forms. For best results, tExpand() should be used in Radian mode.

Text

Catalogue > 23

TextpromptString[, DispFlag]

Programming command: Pauses the programme and displays the character string promptString in a dialogue box.

When the user selects **OK**, programme execution continues.

The optional flag argument can be any expression.

- If DispFlag is omitted or evaluates to **1**, the text message is added to the Calculator history.
- If DispFlag evaluates to **0**, the text message is not added to the history.

If the programme needs a typed response from the user, refer to Request, page 147, or RequestStr, page 148.

Note: You can use this command within a userdefined programme but not within a function.

Define a programme that pauses to display each of five random numbers in a dialogue box.

Within the Prgm...EndPrgm template, complete each line by pressing [4] instead of [enter]. On the computer keyboard, hold down Alt and press Enter.

Define text demo()=Prgm For i,1,5

strinfo:="Random number " & string(rand(i)) Text strinfo

EndFor

EndPrgm

Run the programme:

text demo()

Sample of one dialogue box:



Then

See If, page 86.

tInterval List[,Freq[,CLevel]]

(Data list input)

tinterval \bar{x} , sx,n[,CLevel]

(Summary stats input)

Computes a t confidence interval. A summary of results is stored in the *stat.results* variable (page 174).

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.CLower, stat.CUpper	Confidence interval for an unknown population mean
stat. $\overline{\mathbf{x}}$	Sample mean of the data sequence from the normal random distribution
stat.ME	Margin of error
stat.df	Degrees of freedom
stat.σx	Sample standard deviation
stat.n	Length of the data sequence with sample mean

tInterval_2Samp

Catalogue > 🗐

tInterval_2Samp List1,List2[,Freq1[,Freq2[,CLevel [,Pooled]]]]

(Data list input)

tinterval 2Samp $\bar{x}1$,sx1,n1, $\bar{x}2$,sx2,n2[,CLevel[Pooled]

(Summary stats input)

Computes a two-sample t confidence interval. A summary of results is stored in the *stat.results* variable (page 174).

Pooled=1 pools variances; *Pooled*=0 does not pool variances.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description	
stat.CLower, stat.CUpper	Confidence interval containing confidence level probability of distribution	
stat. \overline{x} 1- \overline{x} 2	Sample means of the data sequences from the normal random distribution	
stat.ME	Margin of error	
stat.df	Degrees of freedom	
stat. \overline{x} 1, stat. \overline{x} 2	Sample means of the data sequences from the normal random distribution	
stat.σx1, stat.σx2	Sample standard deviations for List 1 and List 2	
stat.n1, stat.n2	Number of samples in data sequences	
stat.sp	The pooled standard deviation. Calculated when $Pooled$ = YES	

tmpCnv() Catalogue > [3]

tmpCnv(*Expr*_°*tempUnit*, _°*tempUnit2***)** ⇒ *expression* _°*tempUnit2*

Converts a temperature value specified by Expr from one unit to another. Valid temperature units are:

°C Celsius

°F Fahrenheit

_°K Kelvin

_°R Rankine

To type $^{\circ},$ select it from the Catalogue symbols.

to type _ , press ctrl _ .

For example, 100_°C converts to 212_°F.

To convert a temperature range, use Δ tmpCnv() instead.

tmpCnv(100·_°C,_°F)	212.·_°F
tmpCnv(32·_°F,_°C)	0.·_°C
tmpCnv(0·_°C,_°K)	273.15·_°K
tmpCnv(0·_°F,_°R)	459.67·_°R

Note: You can use the Catalogue to select temperature units.

$\Delta tmpCnv()$

Catalogue > [13]

 $\Delta tmpCnv(Expr_^\circ tempUnit,_^\circ tempUnit2)$ $\Rightarrow expression_^\circ tempUnit2$

Note: You can insert this function from the keyboard by typing deltaTmpCnv (...).

Converts a temperature range (the difference between two temperature values) specified by Expr from one unit to another. Valid temperature units are:

°C	Cel	lsiı	us

_°F Fahrenheit

To enter °, select it from the Symbol Palette or type @d.

1 °C and 1 °K have the same magnitude, as do 1 °F and 1 °R. However, 1 °C is 9/5 as large as 1 °F.

For example, a 100_°C range (from 0_°C to 100 °C) is equivalent to a 180 °F range.

To convert a particular temperature value instead of a range, use tmpCnv().

ΔtmpCnv(100·_°C,_°F)	180.⋅_°F
ΔtmpCnv(180·_°F,_°C)	100.∙_°C
∆tmpCnv(100·_°C,_°K)	100.⋅_°K
ΔtmpCnv(100·_°F,_°R)	100.·_°R
$\Delta tmpCnv(1\cdot_^{\circ}C,_^{\circ}F)$	1.8·_°F

Note: You can use the Catalogue to select temperature units.

tPdf() Catalogue > 🕮

 $tPdf(XVal,df) \Rightarrow number \text{ if } XVal \text{ is a number, } list \text{ if }$ XVal is a list

Computes the probability density function (pdf) for the Student-t distribution at a specified x value with specified degrees of freedom df.

_°K Kelvin

_°R Rankine

trace()

trace(squareMatrix)⇒expression

Returns the trace (sum of all the elements on the main diagonal) of *squareMatrix*.

*	2 3 5 6 8 9	15
trace $\begin{bmatrix} a \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ a \end{bmatrix}$	2·a

Try

Catalogue > 📳

Catalogue > 🕮

Try

block1

Else

block2

EndTry

Executes block1 unless an error occurs. programme execution transfers to block2 if an error occurs in block1. System variable errCode contains the error code to allow the programme to perform error recovery. For a list of error codes, see "Error codes and messages," page 258.

block1 and block2 can be either a single statement or a series of statements separated with the ":" character.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Example 2

To see the commands **Try**, **CIrErr** and **PassErr** in operation, enter the eigenvals() programme shown at the right. Run the programme by executing each of the following expressions.

eigenvals
$$\begin{bmatrix} -3\\ -41\\ 5 \end{bmatrix}$$
 $\begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}$

Define prog I()=Prgm

Try z:=z+1Disp "z incremented."

Else

Disp "Sorry, z undefined."

EndTry

EndPrgm

z:=1:progI()
z incremented.

Done

DelVar z:prog1()

Sorry, z undefined.

Done

Define eigenvals(a,b)=Prgm

© programme eigenvals(A,B) displays eigenvalues of A·B

Try

Disp "A= ",a

Disp "B= ",b

Disp " "

Disp "Eigenvalues of A·B are:",eigVl(a*b)

Try

Catalogue > [13]

eigenvals $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

Note: See also CIrErr, page 25, and PassErr, page 129.

Flse

If errCode=230 Then

Disp "Error: Product of A·B must be a square matrix"

ClrErr

Flse

PassErr

EndIf

EndTrv

EndPrgm

Catalogue > [3] **tTest**

 $\mathsf{tTest} \ \mu 0 \mathcal{L}ist[\mathcal{F}reg[\mathcal{H}ypoth]]$

(Data list input)

tTest $\mu \theta$, \overline{x} ,sx,n,[Hypoth]

(Summary stats input)

Performs a hypothesis test for a single unknown population mean μ when the population standard deviation σ is unknown. A summary of results is stored in the stat.results variable (page 174).

Test H_0 : $\mu = \mu 0$, against one of the following:

For H_a : $\mu < \mu 0$, set Hypoth < 0

For H_a : $\mu \neq \mu 0$ (default), set Hypoth=0

For H_a : $\mu > \mu 0$, set Hypoth > 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.t	$(\overline{\mathbf{x}} - \mu 0) / (\text{stdev} / \text{sqrt(n)})$
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected

Output variable	Description
stat.df	Degrees of freedom
$stat.\overline{\mathbf{x}}$	Sample mean of the data sequence in $List$
stat.sx	Sample standard deviation of the data sequence
stat.n	Size of the sample

tTest_2Samp

Catalogue > 🕄

tTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth* [,*Pooled*]]]]

(Data list input)

 $\mathsf{tTest_2Samp}\ \overline{\mathsf{x}}\ l,\!sx\ l,\!n\ l,\!\overline{\mathsf{x}}\ 2,\!sx\ 2,\!n\ 2[,\!Hypoth[,\!Pooled]]$

(Summary stats input)

Computes a two-sample *t* test. A summary of results is stored in the *stat.results* variable (page 174).

Test H_0 : $\mu 1 = \mu 2$, against one of the following:

For H_a : μ 1< μ 2, set Hypoth<0

For H_a : $\mu 1 \neq \mu 2$ (default), set Hypoth=0

For H_a : μ 1> μ 2, set Hypoth>0

Pooled=1 pools variances

Pooled=0 does not pool variances

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.t	Standard normal value computed for the difference of means
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat.df	Degrees of freedom for the t-statistic
stat. \overline{x} 1, stat. \overline{x} 2	Sample means of the data sequences in $List\ 1$ and $List\ 2$
stat.sx1, stat.sx2	Sample standard deviations of the data sequences in $List\ 1$ and $List\ 2$
stat.n1, stat.n2	Size of the samples
stat.sp	The pooled standard deviation. Calculated when $Pooled$ =1.

Catalogue > [3] tvmFV()

tvmFV(N,I,PV,Pmt,[PpY],[CpY],tvmFV(120,5,0,-500,12,12) [PmtAt]) \Rightarrow value

77641.1

Financial function that calculates the future value of money.

Note: Arguments used in the TVM functions are described in the table of TVM arguments, page 192. See also amortTbl(). page 8.

tvmI() Catalogue > 🕮

tvml(N,PV,Pmt,FV,[PpY],[CpY],[PmtAt]) \Rightarrow value

tvmI(240,100000,-1000,0,12,12) 10.5241

Financial function that calculates the interest rate per year.

Note: Arguments used in the TVM functions are described in the table of TVM arguments, page 192. See also amortTbl(), page 8.

tvmN() Catalogue > 🗐

tvmN(I,PV,Pmt,FV,[PpY],[CpY],tvmN(5,0,-500,77641,12,12) [PmtAt]) \Rightarrow value

120.

Financial function that calculates the number of payment periods.

Note: Arguments used in the TVM functions are described in the table of TVM arguments, page 192. See also amortTbl(), page 8.

tvmPmt() Catalogue > 🗐

tvmPmt(N,I,PV,FV,[PpY],[CpY],tvmPmt(60,4,30000,0,12,12) -552.496 [PmtAt] \Rightarrow value

Financial function that calculates the amount of each payment.

Note: Arguments used in the TVM functions are described in the table of TVM arguments, page 192. See also amortTbl(),

page 8.

Catalogue > 🕄 tvmPV()

tvmPV(N,I,Pmt,FV,[PpY],[CpY],[PmtAt]) \Rightarrow value

tvmPV(48,4,-500,30000,12,12)

-3426.7

Financial function that calculates the present value.

Note: Arguments used in the TVM functions are described in the table of TVM arguments, page 192. See also amortTbl(). page 8.

TVM argument*	Description	Data type
N	Number of payment periods	real number
I	Annual interest rate	real number
PV	Present value	real number
Pmt	Payment amount	real number
FV	Future value	real number
PpY	Payments per year, default=1	integer > 0
СрҮ	Compounding periods per year, default=1	integer > 0
PmtAt	Payment due at the end or beginning of each period, default=end	integer (0=end, 1=beginning)

^{*} These time-value-of-money argument names are similar to the TVM variable names (such as tvm.pv and tvm.pmt) that are used by the Calculator application's finance solver. Financial functions, however, do not store their argument values or results to the TVM variables.

TwoVar Catalogue > 23

TwoVar X, Y[, [Freq] [, Category, Include]]

Calculates the TwoVar statistics. A summary of results is stored in the *stat.results* variable (page 174).

All the lists must have equal dimension except for Include.

X and Y are lists of independent and dependent variables.

Freq is an optional list of frequency values. Each element in *Freq* specifies the frequency of occurrence for each corresponding X and Y data point. The default value is 1. All elements must be integers ≥ 0 .

Category is a list of numeric category codes for the corresponding X and Y data.

Include is a list of one or more of the category codes. Only those data items whose category code is included in this list are included in the calculation.

An empty (void) element in any of the lists X, Freq, or *Category* results in a void for the corresponding element of all those lists. An empty element in any of the lists XI through X20 results in a void for the corresponding element of all those lists. For more information on empty elements, see page 248.

Output variable	Description
stat.x	Mean of x values
stat. x	Sum of x values
stat. x2	Sum of x2 values
stat.sx	Sample standard deviation of x
stat. x	Population standard deviation of x
stat.n	Number of data points
stat. <u>y</u>	Mean of y values
stat. y	Sum of y values
stat. y ²	Sum of y2 values
stat.sy	Sample standard deviation of y
stat. y	Population standard deviation of y
stat. xy	Sum of x · y values
stat.r	Correlation coefficient
stat.MinX	Minimum of x values

Output variable	Description
stat.Q ₁ X	1st Quartile of x
stat.MedianX	Median of x
stat.Q ₃ X	3rd Quartile of x
stat.MaxX	Maximum of x values
stat.MinY	Minimum of y values
stat.Q ₁ Y	1st Quartile of y
stat.MedY	Median of y
stat.Q ₃ Y	3rd Quartile of y
stat.MaxY	Maximum of y values
stat. (x-) ²	Sum of squares of deviations from the mean of x
stat. (y-) ²	Sum of squares of deviations from the mean of y

U

unitV() Catalogue > 📳

unitV(Vector1)⇒vector

Returns either a row- or column-unit vector, depending on the form of Vector 1.

Vector 1 must be either a single-row matrix or a single-column matrix.

$$\frac{\operatorname{unitV}(\begin{bmatrix} a & b & c \end{bmatrix})}{\begin{bmatrix} a \\ \sqrt{a^2+b^2+c^2} \end{bmatrix}} \frac{b}{\sqrt{a^2+b^2+c^2}} \frac{c}{\sqrt{a^2+b^2+c^2}}$$

$$\frac{\operatorname{unitV}(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix})}{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}} \frac{\sqrt{6}}{\sqrt{6}} \frac{\sqrt{6}}{3} \frac{\sqrt{6}}{\sqrt{6}}$$

$$\frac{\sqrt{14}}{\sqrt{14}} \frac{\sqrt{14}}{\sqrt{14}}$$

To see the entire result, press ▲ and then use \triangleleft and \triangleright to move the cursor.

unLock

unLock Var1[, Var2] [, Var3] ...

unLock Var.

Unlocks the specified variables or variable group. Locked variables cannot be modified or deleted.

See Lock, page 105, and getLockinfo(), page 82.

a:=65	65
Lock a	Done
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
a:=75	75
DelVar a	Done

Catalogue > 13

Catalogue > 🕮

Catalogue > 🗐

varPop()

 $varPop(List[, freqList]) \Rightarrow expression$

Returns the population variance of *List*.

Each *freqList* element counts the number of consecutive occurrences of the corresponding element in List.

Note: List must contain at least two elements.

If an element in either list is empty (void), that element is ignored, and the corresponding element in the other list is also ignored. For more information on empty elements, see page 248.

varPop({5,10,15,20,25,30})	875
	12
Ans·1.	72.9167

varSamp()

 $varSamp(List[, freqList]) \Rightarrow expression$

Returns the sample variance of *List*.

Each *freqList* element counts the number of consecutive occurrences of the corresponding element in List.

Note: *List* must contain at least two elements.

$\operatorname{varSamp}(\{a,b,c\})$	
$a^2-a\cdot(b+c)+b^2$	$2-b\cdot c+c^2$
3	
$varSamp({1,2,5,-6,3,-2})$	$\frac{31}{2}$
varSamp({1,3,5},{4,6,2})	68
, mounty((1,5,5),(1,5,2))	33

If an element in either list is empty (void), that element is ignored, and the corresponding element in the other list is also ignored. For more information on empty elements, see page 248.

 $varSamp(Matrix 1[, freqMatrix]) \Rightarrow matrix$

Returns a row vector containing the sample variance of each column in *Matrix 1*.

Each *freqMatrix* element counts the number of consecutive occurrences of the corresponding element in *Matrix1*.

If an element in either matrix is empty (void), that element is ignored, and the corresponding element in the other matrix is also ignored. For more information on empty elements, see page 248.

Note: *Matrix1* must contain at least two rows.

varSamp	1 -3	2	5			[4.75	1.03	4]
					7.			
/	-1.1	2	2.2	6	3			
varSamp	3.4	5	5.1	, 2	4			
varSamp	-2.3	4	1.3	[[5	1]			
					[3.9	1731	2.084	11]

W

Wait

Wait timeInSeconds

Suspends execution for a period of *timeInSeconds* seconds.

Wait is particularly useful in a programme that needs a brief delay to allow requested data to become available.

The argument *timeInSeconds* must be an expression that simplifies to a decimal value in the range 0 through 100. The command rounds this value up to the nearest 0.1 seconds.

To cancel a Wait that is in progress,

- Handheld: Hold down the figure on key and press enter repeatedly.
- Windows®: Hold down the F12 key and press Enter repeatedly.
- Macintosh®: Hold down the F5 key and

Catalogue > 📳

Wait 4

To wait 1/2 second:

To wait 4 seconds:

Wait 0.5

To wait 1.3 seconds using the variable seccount:

seccount:=1.3
Wait seccount

This example switches a green LED on for 0.5 seconds and then switches it off.

Send "SET GREEN 1 ON" Wait 0.5 Send "SET GREEN 1 OFF" press Enter repeatedly.

iPad®: The app displays a prompt. You can continue waiting or cancel.

Note: You can use the Wait command within a user-defined programme but not within a function.

warnCodes ()

 $warnCodes(Expr1, StatusVar) \Rightarrow expression$

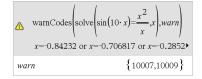
Evaluates expression *Expr1*, returns the result and stores the codes of any generated warnings in the *StatusVar* list variable. If no warnings are generated, this function assigns Status Var an empty list.

Expr1 can be any valid TI-Nspire™ or TI-Nspire™ CAS maths expression. You cannot use a command or assignment as Expr1.

Status Var must be a valid variable name.

For a list of warning codes and associated messages, see page 266.

Catalogue > 🗐



To see the entire result, press \triangle and then use ◀ and ▶ to move the cursor.

when()

when(Condition, trueResult [, falseResult] [, unknownResult]) $\Rightarrow expression$

Returns trueResult, falseResult, or unknownResult, depending on whether Condition is true, false, or unknown. Returns the input if there are too few arguments to specify the appropriate result.

Omit both falseResult and unknownResult to make an expression defined only in the region where Condition is true.

Use an **undef** falseResult to define an expression that graphs only on an interval.

Catalogue > 🗐

when
$$(x<0,x+3)|x=5$$
 undef

when() when() is helpful for defining recursive

Catalogue > 23

functions.

when $(n>0, n \cdot factoral(n-1), 1) \rightarrow factoral$	
	Done
factoral(3)	6
3!	6

While Catalogue > 🗐

While Condition

Block

EndWhile

Executes the statements in *Block* as long as Condition is true.

Block can be either a single statement or a sequence of statements separated with the ":" character.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define sum_of_recip(r	=Func		
	Local i,tempsum		
	$1 \rightarrow i$		
	$0 \rightarrow tempsum$		
	While $i \le n$		
	$tempsum + \frac{1}{i} \rightarrow tempsum$		
	$i+1 \rightarrow i$		
	EndWhile		
	Return tempsum		
	EndFunc		
	Done		
sum_of_recip(3)	11		
	6		

X

Catalogue > 🕮 xor

BooleanExpr1xorBooleanExpr2 returns Boolean expression

BooleanList1xorBooleanList2 returns Boolean list

BooleanMatrix1xorBooleanMatrix2 returns *Boolean matrix*

Returns true if BooleanExpr1 is true and BooleanExpr2 is false, or vice versa.

Returns false if both arguments are true or if both are false. Returns a simplified Boolean expression if either of the arguments cannot be resolved to true or false.

true xor true	false
5>3 xor 3>5	true

Note: See or, page 127.

Integer 1 xor Integer $2 \Rightarrow integer$

Compares two real integers bit-by-bit using an xor operation. Internally, both integers are converted to signed, 64-bit binary numbers. When corresponding bits are compared, the result is 1 if either bit (but not both) is 1; the result is 0 if both bits are 0 or both bits are 1. The returned value represents the bit results and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 64-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range. For more information, see >Base2, page 17.

Note: See or, page 127.

In Hex base mode:

Important: Zero, not the letter O.

0h7AC36 xor 0h3D5F 0h79169

In Bin base mode:

0b100101 xor 0b100 0b100001

Note: A binary entry can have up to 64 digits (not counting the 0b prefix). A hexadecimal entry can have up to 16 digits.

Z

zeroes()

 $zeroes(Expr, Var) \Rightarrow list$

 $zeroes(Expr, Var=Guess) \Rightarrow list$

Returns a list of candidate real values of Var that make Expr=0. zeroes() does this by computing explist(solve (Expr=0, Var), Var).

For some purposes, the result form for zeroes() is more convenient than that of solve(). However, the result form of zeroes () cannot express implicit solutions. solutions that require inequalities, or solutions that do not involve Var.

Note: See also cSolve(), cZeroes() and solve ().

Catalogue > 23

$$\frac{\left\{\frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}, \frac{-\left(\sqrt{b^2 - 4 \cdot a \cdot c} + b\right)}{2 \cdot a}\right\}}{a \cdot x^2 + b \cdot x + c|x = Ans[2]}$$

$$\frac{\operatorname{exact}\left(\operatorname{zeros}\left(a\cdot\left(e^{x}+x\right)\cdot\left(\operatorname{sign}(x)-1\right),x\right)\right) \quad \left\{ \square \right\}}{\operatorname{exact}\left(\operatorname{solve}\left(a\cdot\left(e^{x}+x\right)\cdot\left(\operatorname{sign}(x)-1\right)=0,x\right)\right)}$$

$$e^{x}+x=0 \text{ or } x>0 \text{ or } a=0$$

zeroes({Expr1, Expr2}, {VarOrGuess1, VarOrGuess2 [, ...]}) $\Rightarrow matrix$

Returns candidate real zeroes of the simultaneous algebraic expressions, where each VarOrGuess specifies an unknown whose value you seek.

Optionally, you can specify an initial guess for a variable. Each VarOrGuess must have the form:

variable

- or -

variable = real or non-real number

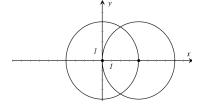
For example, x is valid and so is x=3.

If all of the expressions are polynomials and if you do NOT specify any initial guesses, zeroes() uses the lexical Gröbner/Buchberger elimination method to attempt to determine all real zeroes.

For example, suppose you have a circle of radius r at the origin and another circle of radius r centred where the first circle crosses the positive x-axis. Use zeroes() to find the intersections.

As illustrated by r in the example to the right, simultaneous polynomial expressions can have extra variables that have no values, but represent given numeric values that could be substituted later.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the *varOrGuess* list. To extract a row, index the matrix by [row].



zeros
$$\left\{ x^2 + y^2 - r^2, (x-r)^2 + y^2 - r^2 \right\}, \left\{ x, y \right\}$$

$$\left[\frac{r}{2} \quad \frac{-\sqrt{3} \cdot r}{2} \right]$$

$$\left[\frac{r}{2} \quad \frac{\sqrt{3} \cdot r}{2} \right]$$

Extract row 2:

Ans[2]	<u>r</u>	$\sqrt{3} \cdot r$
	2	2

You can also (or instead) include unknowns that do not appear in the expressions. For example, you can include z as an unknown to extend the previous example to two parallel intersecting cylinders of radius r. The cylinder zeroes illustrate how families of zeroes might contain arbitrary constants in the form ck, where k is an integer suffix from 1 through 255.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or varOrGuess list.

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in the unknowns, zeroes() uses Gaussian elimination to attempt to determine all real zeroes.

If a system is neither polynomial in all of its variables nor linear in its unknowns, zeroes () determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

Each unknown starts at its guessed value if there is one: otherwise, it starts at 0.0.

Use guesses to seek additional zeroes one by one. For convergence, a guess may have to be rather close to a zero.

$$\frac{\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\left\{x,y,z\right\}\right)}{\left[\frac{r}{2} \frac{-\sqrt{3}\cdot r}{2} c1\right]}$$

$$\frac{r}{2} \frac{\sqrt{3}\cdot r}{2} c1$$

zeros
$$\left\{x+e^z \cdot y-1, x-y-\sin(z)\right\}, \left\{x,y\right\}$$

$$\left[\frac{e^z \cdot \sin(z)+1}{e^z+1} \quad \frac{-\left(\sin(z)-1\right)}{e^z+1}\right]$$

zeros(
$$\{e^z \cdot y - 1, \neg y - \sin(z)\}, \{y, z\}$$
)
$$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76\mathbf{e} - 11 & 1796.99 \\ 2.\mathbf{e} - 13 & 254.469 \end{bmatrix}$$

$$\frac{1}{\operatorname{zeros}(\left\{\mathbf{e}^{z} \cdot y - 1, -y - \sin(z)\right\}, \left\{y, z = 2 \cdot \pi\right\})} \begin{bmatrix} 0.001871 & 6.28131 \end{bmatrix}$$

zInterval

Catalogue > 🕮

zInterval σ_{l} List[,Freq[,CLevel]]

(Data list input)

zinterval σ, \overline{x}, n [, CLevel]

(Summary stats input)



Computes a z confidence interval. A summary of results is stored in the *stat.results* variable (page 174).

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.CLower, stat.CUpper	Confidence interval for an unknown population mean
$\operatorname{stat}.\overline{\mathbf{x}}$	Sample mean of the data sequence from the normal random distribution
stat.ME	Margin of error
stat.sx	Sample standard deviation
stat.n	Length of the data sequence with sample mean
stat.σ	Known population standard deviation for data sequence List

zInterval_1Prop

Catalogue > 23

zInterval 1Prop x,n [,CLevel]

Computes a one-proportion z confidence interval. A summary of results is stored in the stat.results variable (page 174).

x is a non-negative integer.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.CLower, stat.CUpper	Confidence interval containing confidence level probability of distribution
stat. \hat{p}	The calculated proportion of successes
stat.ME	Margin of error
stat.n	Number of samples in data sequence

zInterval_2Prop

Catalogue > 23

zInterval 2Prop x1,n1,x2,n2[,CLevel]

Computes a two-proportion z confidence interval. A summary of results is stored in the stat.results variable (page 174).



x1 and x2 are non-negative integers.

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.CLower, stat.CUpper	Confidence interval containing confidence level probability of distribution
stat. \hat{p} Diff	The calculated difference between proportions
stat.ME	Margin of error
stat. p 1	First sample proportion estimate
stat. p̂ 2	Second sample proportion estimate
stat.n1	Sample size in data sequence one
stat.n2	Sample size in data sequence two

zInterval_2Samp

Catalogue > 23

zInterval_2Samp σ_1, σ_2 , List1, List2[, Freq1[, Freq2, [*CLevel*]]]

(Data list input)

zinterval_2Samp $\sigma_1, \sigma_2, \overline{x} l, nl, \overline{x} 2, n2[, CLevel]$

(Summary stats input)

Computes a two-sample z confidence interval. A summary of results is stored in the *stat.results* variable (page 174).

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.CLower, stat.CUpper	Confidence interval containing confidence level probability of distribution
stat. \overline{x} 1- \overline{x} 2	Sample means of the data sequences from the normal random distribution
stat.ME	Margin of error
$\operatorname{stat}.\overline{x}1$, $\operatorname{stat}.\overline{x}2$	Sample means of the data sequences from the normal random distribution

Output variable	Description
stat.σx1, stat.σx2	Sample standard deviations for $List\ 1$ and $List\ 2$
stat.n1, stat.n2	Number of samples in data sequences
stat.r1, stat.r2	Known population standard deviations for data sequence $List\ 1$ and $List\ 2$

zTest Catalogue > 👰

zTest μ *θ*, σ ,List,[Freq[Hypoth]]

(Data list input)

zTest μ *θ*, σ , \overline{x} ,n[,Hypoth]

(Summary stats input)

Performs a z test with frequency *freqlist*. A summary of results is stored in the *stat.results* variable (page 174).

Test H_0 : $\mu = \mu 0$, against one of the following:

For H_a : $\mu < \mu 0$, set Hypoth < 0

For H_a: $\mu \neq \mu 0$ (default), set Hypoth=0

For H_a : $\mu > \mu 0$, set Hypoth > 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.z	$(\overline{\mathbf{x}} - \mu 0) / (\sigma / \text{sqrt(n)})$
stat.P Value	Least probability at which the null hypothesis can be rejected
$stat.\overline{\mathbf{x}}$	Sample mean of the data sequence in List
stat.sx	Sample standard deviation of the data sequence. Only returned for ${\it Data}$ input.
stat.n	Size of the sample

zTest_1Prop

Catalogue > 🔯

 $\mathbf{zTest_1Prop}\ p0,\!x,\!n[,\!Hypoth]$

zTest 1Prop

Computes a one-proportion z test. A summary of results is stored in the stat.results variable (page 174).

x is a non-negative integer.

Test H_0 : $p = p\theta$ against one of the following:

For H_a : p > p0, set Hypoth > 0

For H_a : $p \neq p0$ (default), set Hypoth=0

For H_a : p < p0, set Hypoth < 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.p0	Hypothesized population proportion
stat.z	Standard normal value computed for the proportion
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat. \hat{p}	Estimated sample proportion
stat.n	Size of the sample

zTest 2Prop

Catalogue > 🗐

 $zTest_2Prop x1,n1,x2,n2[,Hypoth]$

Computes a two-proportion z test. A summary of results is stored in the stat.results variable (page 174).

x1 and x2 are non-negative integers.

Test H_0 : p1 = p2, against one of the following:

For H_a : p1 > p2, set Hypoth > 0

For H_a : $p1 \neq p2$ (default), set Hypoth=0

For H_a : p < p0, set Hypoth < 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.z	Standard normal value computed for the difference of proportions
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat. p ̂ 1	First sample proportion estimate
stat. p 2	Second sample proportion estimate
stat. $\hat{\pmb{p}}$	Pooled sample proportion estimate
stat.n1, stat.n2	Number of samples taken in trials 1 and 2

zTest_2Samp

Catalogue > 23

zTest_2Samp σ_1, σ_2 ,*List1*,*List2*[,*Freq1*[,*Freq2* [,*Hypoth*]]]

(Data list input)

zTest_2Samp $\sigma_1, \sigma_2, \overline{x}1, n1, \overline{x}2, n2[, Hypoth]$

(Summary stats input)

Computes a two-sample *z* test. A summary of results is stored in the *stat.results* variable (page 174).

Test H_0 : $\mu 1 = \mu 2$, against one of the following:

For H_a : μ 1 < μ 2, set Hypoth<0

For H_a : $\mu 1 \neq \mu 2$ (default), set Hypoth=0

For H_a : $\mu 1 > \mu 2$, Hypoth > 0

For information on the effect of empty elements in a list, see "Empty (Void) Elements", page 248.

Output variable	Description
stat.z	Standard normal value computed for the difference of means
stat.PVal	Smallest level of significance at which the null hypothesis can be rejected
stat. \overline{x} 1, stat. \overline{x} 2	Sample means of the data sequences in List1 and List2
stat.sx1, stat.sx2	Sample standard deviations of the data sequences in List1 and List2
stat.n1, stat.n2	Size of the samples

Symbols

+ (add)		+ key
$Expr1 + Expr2 \Rightarrow expression$	56	56
Returns the sum of the two arguments.	56+4	60
	60+4	64
	64+4	68
	68+4	72
$List1 + List2 \Rightarrow list$	$\left\{22,\pi,\frac{\pi}{2}\right\}\to l1$	$\left\{22,\pi,\frac{\pi}{2}\right\}$
$Matrix1 + Matrix2 \Rightarrow matrix$		(-)
Returns a list (or matrix) containing the	$\left\{10,5,\frac{\pi}{2}\right\} \to l2$	$\left\{10,5,\frac{\pi}{2}\right\}$
sums of corresponding elements in <i>List1</i> and <i>List2</i> (or <i>Matrix1</i> and <i>Matrix2</i>).	$\frac{11+12}{Ans+\{\pi, -5, -\pi\}}$	$\frac{\left\{32,\pi+5,\pi\right\}}{\left\{\pi+32,\pi,0\right\}}$
Dimensions of the arguments must be equal.	$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$
·	[0 4] [0 1]	[c u · 1]
$Expr + Listl \Rightarrow list$	15+{10,15,20}	{25,30,35}
$List1 + Expr \Rightarrow list$	{10,15,20}+15	{25,30,35}
Returns a list containing the sums of $Expr$ and each element in $List1$.		
$Expr + Matrix 1 \Rightarrow matrix$	20+ 1 2	21 2
$Matrix l + Fxnr \Rightarrow matrix$	[3 4]	[3 24]

 $Matrix1 + Expr \Rightarrow matrix$

Returns a matrix with Expr added to each element on the diagonal of Matrix 1. *Matrix1* must be square.

Note: Use .+ (dot plus) to add an expression to each element.

- (subtract)		- key
$Expr1 - Expr2 \Rightarrow expression$	6-2	4
Returns Expr1 minus Expr2.	$\pi - \frac{\pi}{6}$	$\frac{5 \cdot \pi}{6}$
List1 −List2⇒ list	$\left\{22,\pi,\frac{\pi}{2}\right\} - \left\{10,5,\frac{\pi}{2}\right\}$	$\{12,\pi-5,0\}$
$Matrix1 - Matrix2 \Rightarrow matrix$	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	[2 2]

– (subtract)



Subtracts each element in List2 (or *Matrix2*) from the corresponding element in *List1* (or *Matrix1*), and returns the results.

Dimensions of the arguments must be equal.

$$Expr - List1 \Rightarrow list$$

$$List1 - Expr \Rightarrow list$$

Subtracts each *List1* element from *Expr* or

 $Expr - Matrix 1 \Rightarrow matrix$

and returns a list of the results.

$$Matrix 1 - Expr \Rightarrow matrix$$

Expr - Matrix 1 returns a matrix of Exprtimes the identity matrix minus Matrix1. Matrix1 must be square.

subtracts *Expr* from each *List1* element,

Matrix 1 - Expr returns a matrix of Exprtimes the identity matrix subtracted from Matrix1. Matrix1 must be square.

Note: Use .- (dot minus) to subtract an expression from each element.

15-{10,15,20}	{ 5,0,-5 }
{10,15,20}-15	{-5,0,5}

20-1	2		19	-2
[3	4	.]	-3	16

(multiply)

$Expr1 \cdot Expr2 \Rightarrow expression$

Returns the product of the two arguments.

$$List1$$
• $List2 \Rightarrow list$

Returns a list containing the products of the corresponding elements in *List1* and *List2*.

Dimensions of the lists must be equal.

 $Matrix 1 \cdot Matrix 2 \Rightarrow matrix$

Returns the matrix product of *Matrix1* and Matrix2.

The number of columns in *Matrix I* must equal the number of rows in *Matrix2*.

$$\begin{array}{ccc}
2 \cdot 3.45 & 6.9 \\
x \cdot y \cdot x & x^2 \cdot y
\end{array}$$

× kev

$$\begin{cases}
1.,2,3 \cdot \{4,5,6\} & \{4,10,18\} \\
\frac{2}{a},\frac{3}{2} \cdot \left\{a^2,\frac{b}{3}\right\} & \left\{2 \cdot a,\frac{b}{2}\right\}
\end{cases}$$

$$\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix} \begin{bmatrix}
a & d \\
b & e \\
c & f
\end{bmatrix}$$

$$\begin{bmatrix}
a+2 \cdot b+3 \cdot c & d+2 \cdot e+3 \cdot f \\
4 \cdot a+5 \cdot b+6 \cdot c & 4 \cdot d+5 \cdot e+6 \cdot f
\end{bmatrix}$$

•(multiply)

× kev

$$Expr \cdot Listl \Rightarrow list$$

 $\pi \cdot \{4,5,6\} \qquad \qquad \{4 \cdot \pi, 5 \cdot \pi, 6 \cdot \pi\}$

 $List1 \cdot Expr \Rightarrow list$

Returns a list containing the products of Expr and each element in List 1.

 $Expr \cdot Matrix 1 \Rightarrow matrix$

 $Matrix1 \cdot Expr \Rightarrow matrix$

Returns a matrix containing the products of Expr and each element in Matrix 1.

$[1 \ 2].0.01$	$\begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$
[3 4]	0.03 0.04
λ·identity(3)	[\lambda 0 0]
	0 1 0
	[0 0 λ]

Note: Use .•(dot multiply) to multiply an expression by each element.

/	(divide)	
/	uiviue,	

÷ key

 $Expr1/Expr2 \Rightarrow expression$

Returns the quotient of Expr1 divided by Expr2.

Note: See also Fraction template, page 1.

 $List1/List2 \Rightarrow list$

Returns a list containing the quotients of *List1* divided by *List2*.

Dimensions of the lists must be equal.

 $Expr/List1 \Rightarrow list$

 $List1/Expr \Rightarrow list$

Returns a list containing the quotients of Expr divided by List1 or List1 divided by Expr.

 $Matrix1/Expr \Rightarrow matrix$

Returns a matrix containing the quotients of Matrix 1/Expr.

 $Matrix1/Value \Rightarrow matrix$

$\frac{2}{3.45}$ 0.57971 $\frac{x^3}{x^2}$

$\frac{a}{\left\{3,a,\sqrt{a}\right\}}$	$\left\{\frac{a}{3},1,\sqrt{a}\right\}$
$\frac{\{a,b,c\}}{a \cdot b \cdot c}$	$\left\{\frac{1}{b \cdot c}, \frac{1}{a \cdot c}, \frac{1}{a \cdot b}\right\}$

$\begin{bmatrix} a & b & c \end{bmatrix}$	1_1_	_1_	_1_
$a \cdot b \cdot c$	$b \cdot c$	$a \cdot c$	$a \cdot b$

/(divide)



Note: Use ./ (dot divide) to divide an expression by each element.

^ (power)

^ key

Expr1	^ <i>Expr2</i> ⇒	expression
-------	------------------	------------

$$\{a,2,c\}$$
 $\{1,b,3\}$ $\{a,2^b,c^3\}$

Returns the first argument raised to the power of the second argument.

Note: See also Exponent template, page 1.

For a list, returns the elements in *List1* raised to the power of the corresponding elements in *List2*.

In the real domain, fractional powers that have reduced exponents with odd denominators use the real branch versus the principal branch for complex mode.

$$Expr \land List1 \Rightarrow list$$

Returns Expr raised to the power of the elements in List1.

$$List1 \land Expr \Rightarrow list$$

Returns the elements in List1 raised to the power of Expr.

 $squareMatrix1 \land integer \Rightarrow matrix$

Returns *squareMatrix1* raised to the *integer* power.

square Matrix 1 must be a square matrix.

If integer = -1, computes the inverse matrix.

If *integer* < -1, computes the inverse matrix to an appropriate positive power.

$p^{\{a,2,-3\}}$	$\int_{\mathbb{R}^{n}} a_{n}^{2} \underline{1}$
] <i>p</i> , <i>p</i> ,—

$$\{1,2,3,4\}^{-2}$$
 $\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\}$

1	2]2	7	10
3	$\begin{bmatrix} 2 \\ 4 \end{bmatrix}$	15	22

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \qquad \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \\ 11 & -5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \qquad \qquad \begin{bmatrix} \frac{11}{2} & \frac{-5}{2} \\ \frac{-15}{4} & \frac{7}{4} \end{bmatrix}$$

x² (square)

x² key

. + kevs

- kevs

Expr12⇒ expression

Returns the square of the argument.

List $12 \Rightarrow list$

Returns a list containing the squares of the elements in List1.

 $squareMatrix 12 \Rightarrow matrix$

Returns the matrix square of squareMatrix I. This is not the same as calculating the square of each element. Use .^2 to calculate the square of each element.

42	16
$\{2,4,6\}^2$	{4,16,36}
$ \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^{2} $	40 64 88 49 79 109 58 94 130
$ \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} $ $^{\circ}$ 2	4 16 36 9 25 49 16 36 64

.+ (dot add)

 $Matrix1 + Matrix2 \Rightarrow matrix$

Expr .+ Matrix 1 ⇒ matrix

Matrix1.+Matrix2 returns a matrix that is the sum of each pair of corresponding elements in Matrix1 and Matrix2.

Expr.+Matrix 1 returns a matrix that is the sum of Expr and each element in Matrix 1.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} + \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a+c \\ b+5 \end{bmatrix}$	6
$\begin{bmatrix} b & 3 \end{bmatrix} \begin{bmatrix} 5 & d \end{bmatrix}$	b+5	d+3
$x + \begin{bmatrix} c & 4 \end{bmatrix}$	x+c	$\begin{bmatrix} x+4 \\ x+d \end{bmatrix}$
[5 d]	x+5	x+d

.- (dot subt.)

Matrix1 .- Matrix2⇒ matrix

Expr - $Matrix 1 \Rightarrow matrix$

Matrix1.— Matrix2 returns a matrix that is the difference between each pair of corresponding elements in Matrix1 and Matrix2.

 $Expr.-Matrix \ I$ returns a matrix that is the difference of Expr and each element in $Matrix \ I$.

	Keys
$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \cdot \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix}$	$\begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$
$[b \ 3] \ [d \ 5]$	$\lfloor b-d$ -2 \rfloor
$\begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix}$	$\begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$
$\begin{bmatrix} d & 5 \end{bmatrix}$	$\begin{bmatrix} x-d & x-5 \end{bmatrix}$

Symbols 211

.•(dot mult.)

Matrix1 .• Matrix2⇒ matrix

 $Expr \cdot Matrix 1 \Rightarrow matrix$

Matrix1.• Matrix2 returns a matrix that is the product of each pair of corresponding elements in Matrix1 and Matrix2.

Expr. • Matrix I returns a matrix containing the products of Expr and each element in Matrix I.

[a 2].[c 4]	[a·c 8
$\begin{bmatrix} b & 3 \end{bmatrix}$ $\begin{bmatrix} 5 & d \end{bmatrix}$	$\begin{bmatrix} a \cdot c & 8 \\ 5 \cdot b & 3 \cdot d \end{bmatrix}$
$x \cdot \begin{bmatrix} a & b \end{bmatrix}$	$a \cdot x b \cdot x$
$\begin{vmatrix} c & d \end{vmatrix}$	$c \cdot x = d \cdot x$

./(dot divide)

 $Matrix 1./Matrix 2 \Rightarrow matrix$

 $Expr./Matrix1 \Rightarrow matrix$

Matrix1./Matrix2 returns a matrix that is the quotient of each pair of corresponding elements in Matrix1 and Matrix2.

Expr./Matrix1 returns a matrix that is the quotient of Expr and each element in Matrix1

	÷_ keys
$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} . / \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$
	$\begin{bmatrix} \frac{b}{5} & \frac{3}{d} \end{bmatrix}$
$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$
	$\left[\frac{x}{5} \frac{x}{d}\right]$

.^ (dot power)

 $Matrix1 \land Matrix2 \Rightarrow matrix$

 $Expr. \land Matrix 1 \Rightarrow matrix$

Matrix1. Matrix2 returns a matrix where each element in Matrix2 is the exponent for the corresponding element in Matrix1.

Expr. \land Matrix I returns a matrix where each element in Matrix I is the exponent for Expr.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} . $	$\begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$
$x \land \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$

– (negate)

(–) key

 $-Expr1 \Rightarrow expression$

 $-List1 \Rightarrow list$

 $-Matrix1 \Rightarrow matrix$

-2.43	-2.43
$-\{-1,0.4,1.2$ E19 $\}$	$\{1,-0.4,-1.2$ E19 $\}$
-a·-b	$a \cdot b$

- (negate)



Returns the negation of the argument.

For a list or matrix, returns all the elements negated.

If the argument is a binary or hexadecimal integer, the negation gives the two's complement.

In Bin base mode:

Important: Zero, not the letter O.

To see the entire result, press \triangle and then use \triangleleft and \triangleright to move the cursor.

% (percent)

ctrl 🕮 keys

 $Exprl\% \Rightarrow expression$

List1% ⇒ list

 $Matrix 1\% \Rightarrow matrix$

Note: To force an approximate result,

Handheld: Press ctrl enter.
Windows®: Press Ctrl+Enter.
Macintosh®: Press #Enter.
iPad®: Hold enter, and select = .

13%

0.13

({1,10,100})%

{0.01,0.1,1.

<u>argument</u>

For a list or matrix, returns a list or matrix with each element divided by 100.

= (equal)

Returns



 $Expr1=Expr2 \Rightarrow Boolean \ expression$

List1= $List2 \Rightarrow Boolean\ list$

 $Matrix 1 = Matrix 2 \Rightarrow Boolean matrix$

Returns true if Expr1 is determined to be equal to Expr2.

Returns false if Expr1 is determined to not be equal to Expr2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Example function that uses maths test symbols: =, \neq , <, \leq , >, \geq

Define g(x)=Func

If $x \le -5$ Then

Return 5

ElseIf x > -5 and x < 0 Then

Return -x

ElseIf $x \ge 0$ and $x \ne 10$ Then

Return x

ElseIf x=10 Then

Return 3

EndIf

EndFunc

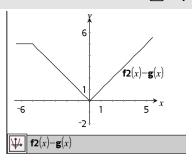
Done

Result of graphing g(x)

= (equal)

= kev

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.



\neq (not equal)

ctrl = kevs

 $Expr1 \neq Expr2 \Rightarrow Boolean expression$

See "=" (equal) example.

 $List1 \neq List2 \Rightarrow Boolean\ list$

 $Matrix 1 \neq Matrix 2 \Rightarrow Boolean matrix$

Returns true if *Expr1* is determined to be not equal to Expr2.

Returns false if *Expr1* is determined to be equal to Expr2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Note: You can insert this operator from the keyboard by typing /=

< (less than)

ctri = kevs

 $Expr1 < Expr2 \Rightarrow Boolean expression$

See "=" (equal) example.

 $List1 < List2 \Rightarrow Boolean list$

Matrix1<*Matrix2* ⇒ *Boolean matrix*

Returns true if *Expr1* is determined to be less than Expr2.

Returns false if Expr1 is determined to be greater than or equal to $\hat{Expr2}$.

< (less than)

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

\leq (less or equal)

ctrl = kevs

 $Expr1 \leq Expr2 \Rightarrow Boolean expression$

See "=" (equal) example.

 $List1 \le List2 \Rightarrow Boolean\ list$

 $Matrix1 < Matrix2 \Rightarrow Boolean matrix$

Returns true if *Expr1* is determined to be less than or equal to Expr2.

Returns false if Expr1 is determined to be greater than Expr2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Note: You can insert this operator from the keyboard by typing <=

> (greater than)

ctrl = kevs

 $Expr1>Expr2 \Rightarrow Boolean expression$

See "=" (equal) example.

 $List1>List2 \Rightarrow Boolean\ list$

 $Matrix 1 > Matrix 2 \Rightarrow Boolean matrix$

Returns true if Expr1 is determined to be greater than Expr2.

Returns false if *Expr1* is determined to be less than or equal to Expr2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

≥ (greater or equal)

ctrl = keys

 $Expr1 \ge Expr2 \Rightarrow Boolean expression$

See "=" (equal) example.

List1>List2 ⇒ Boolean list

 $Matrix1 \ge Matrix2 \Rightarrow Boolean \ matrix$

Returns true if Expr1 is determined to be greater than or equal to Expr2.

Returns false if ExprI is determined to be less than Expr2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Note: You can insert this operator from the keyboard by typing >=

⇒ (logical implication)

ctrl = keys

BooleanExpr1 ⇒ BooleanExpr2 returns Boolean expression

BooleanList1 ⇒ BooleanList2 returns
Boolean list

BooleanMatrix1 ⇒ BooleanMatrix2 returns Boolean matrix

 $Integer1 \Rightarrow Integer2$ returns Integer

Evaluates the expression **not** <argument1> **or** <argument2> and returns true, false, or a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Note: You can insert this operator from the keyboard by typing =>

5>3 or 3>5	true
5>3 ⇒ 3>5	false
3 or 4	7
3 ⇒ 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
$\boxed{\{1,2,3\} \Rightarrow \{3,2,1\}}$	{-1,-1,-3}

⇔ (logical double implication, XNOR)

 $BooleanExpr1 \Leftrightarrow BooleanExpr2$ returns Boolean expression

 $BooleanList1 \Leftrightarrow BooleanList2$ returns Boolean list

BooleanMatrix1

⇔ BooleanMatrix2 returns Boolean matrix

Integer1 ⇔ *Integer2* returns *Integer*

Returns the negation of an **XOR** Boolean operation on the two arguments. Returns true, false, or a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Note: You can insert this operator from the keyboard by typing <=>

5>3 xor 3>5	true
5>3 ⇔ 3>5	false
3 xor 4	7
3 ⇔ 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
$\{1,2,3\} \leftrightarrow \{3,2,1\}$	{-3,-1,-3}

! (factorial)

 $Matrix 1! \Rightarrow matrix$

Returns the factorial of the argument.

For a list or matrix, returns a list or matrix of factorials of the elements.

?!► kev $Expr1! \Rightarrow expression$ 120 ({5,4,3})! {120,24,6} $List1! \Rightarrow list$ 2 1 2 24

ctrl 🕮 kevs & (append) String1 & String2 ⇒ string "Hello "&"Nick" "Hello Nick"

Returns a text string that is *String2* appended to String1.

d() (derivative)

Catalogue > 23

 $d(Expr1, Var[, Order]) \Rightarrow expression$

 $d(List1, Var[, Order]) \Rightarrow list$

 $d(Matrix1, Var[, Order]) \Rightarrow matrix$

Returns the first derivative of the first argument with respect to variable Var.

Order, if included, must be an integer. If the order is less than zero, the result will be an anti-derivative.

Note: You can insert this function from the keyboard by typing **derivative** (...).

d() does not follow the normal evaluation mechanism of fully simplifying its arguments and then applying the function definition to these fully simplified arguments. Instead, d() performs the following steps:

- Simplify the second argument only to the extent that it does not lead to a non-variable.
- Simplify the first argument only to the extent that it does recall any stored value for the variable determined by step 1.
- 3. Determine the symbolic derivative of the result of step 2 with respect to the variable from step 1.

If the variable from step 1 has a stored value or a value specified by the constraint ("|") operator, substitute that value into the result from step 3.

Note: See also First derivative, page 5; Second derivative, page 6; or Nth derivative, page 6.

$\frac{d}{dx}(f(x)\cdot g(x))$	$\frac{d}{dx}(f(x))\cdot g(x) + \frac{d}{dx}(g(x))\cdot f(x)$
$\frac{d}{dy} \left(\frac{d}{dx} \left(x^2 \cdot y^3 \right) \right)$	$6\cdot y^2\cdot x$
$\frac{d}{dx} \left(\left\{ x^2, x^3, x^4 \right\} \right)$	$\left\{2\cdot x, 3\cdot x^2, 4\cdot x^3\right\}$

∫() (integral)		Catalogue > 🕎
$ (Expr1, Var[,Lower,Upper]) \Rightarrow $ expression	$\int_{x^2 dx}^{b}$	$\frac{b^3}{3} - \frac{a^3}{3}$
$\int (Exprl,Var[,Constant]) \Rightarrow expression$	$\int a$	

Returns the integral of Expr1 with respect to the variable Var from Lower to Upper.

Note: See also Definite or Indefinite integral template, page 6.

Note: You can insert this function from the keyboard by typing integral (...).

If Lower and Upper are omitted, returns an anti-derivative. A symbolic constant of integration is omitted unless you provide the Constant argument.

Equally valid anti-derivatives might differ by a numeric constant. Such a constant might be disguised—particularly when an anti-derivative contains logarithms or inverse trigonometric functions. Moreover, piecewise constant expressions are sometimes added to make an anti-derivative valid over a larger interval than the usual formula.

() returns itself for pieces of *Expr1* that it cannot determine as an explicit finite combination of its built-in functions and operators.

When you provide *Lower* and *Upper*, an attempt is made to locate any discontinuities or discontinuous derivatives in the interval *Lower < Var < Upper* and to subdivide the interval at those places.

For the Auto setting of the Auto or Approximate mode, numerical integration is used where applicable when an antiderivative or a limit cannot be determined.

For the Approximate setting, numerical integration is tried first, if applicable. Antiderivatives are sought only where such numerical integration is inapplicable or fails.

$$\frac{\int x^2 dx}{\int (a \cdot x^2, x, c)} \frac{\frac{x^3}{3}}{\frac{a \cdot x^3}{3} + c}$$

$$\int b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} dx \quad b \cdot \int e^{-x^2} dx + \tan^{-1} \left(\frac{x}{a}\right)$$

Note: To force an approximate result,

Handheld: Press ctrl enter.
Windows®: Press Ctrl+Enter.
Macintosh®: Press #Enter.
iPad®: Hold enter, and select = ...

Catalogue > [13]

() can be nested to do multiple integrals. Integration limits can depend on integration variables outside them.

Note: See also nint(), page 120.

$$\int_{0}^{a} \int_{0}^{x} \ln(x+y) \, dy \, dx$$

$$\frac{a^{2} \cdot \ln(a)}{2} + \frac{a^{2} \cdot (4 \cdot \ln(2) - 3)}{4}$$

$\sqrt{\text{() (square root)}}$ $\sqrt{\text{ctrl } x^2 \text{ keys}}$ $\sqrt{(Exprl)} \Rightarrow expression$ $\sqrt{4}$ 2 $\sqrt{(Listl)} \Rightarrow list$ $\sqrt{9,a,4}$ $3,\sqrt{a,2}$

Returns the square root of the argument.

For a list, returns the square roots of all the elements in *List1*.

Note: You can insert this function from the keyboard by typing **sqrt(...)**

Note: See also Square root template, page 1.

11() (p. 04004)		catalogue > sas
$\Pi(Expr1, Var, Low, High) \Rightarrow expression$	5	_1
Note: You can insert this function from the keyboard by typing prodSeq() .	$\left \begin{array}{c} \left \begin{array}{c} \left(\frac{1}{n}\right) \end{array}\right $ $n=1$	120
Evaluates $Expr1$ for each value of Var from Low to $High$, and returns the product of the results.	$\frac{n}{\left \begin{array}{c} k \\ k \end{array}\right } \left(k^2\right)$	(n!) ²
Note: See also Product template (Π), page 5.	$\frac{5}{\prod_{n=1}^{5}} \left\{ \left\{ \frac{1}{n}, n, 2 \right\} \right\}$	$\left\{\frac{1}{120},120,32\right\}$

 Π () (prodSea)

Π () (prodSeq)

Catalogue > 🕎

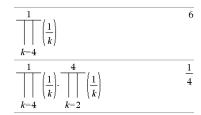
 $\Pi(Expr1, Var, Low, Low-1) \Rightarrow 1$

 $\Pi(Expr1, Var, Low, High) \Rightarrow 1/\Pi(Expr1, Var, High+1, Low-1)$ if High < Low-1

 $\frac{3}{\prod_{k=4}}(k)$

The product formulas used are derived from the following reference:

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.



Σ () (sumSeq)

 $\Sigma(Expr1, Var, Low, High) \Rightarrow expression$

Note: You can insert this function from the keyboard by typing sumSeq(...).

Evaluates ExprI for each value of Var from Low to High, and returns the sum of the results.

Note: See also Sum template, page 5.

 $\Sigma(Expr1, Var, Low, Low-1) \Rightarrow 0$

 $\Sigma(Expr1, Var, Low, High) \Rightarrow \mu$

 Σ (Expr1, Var, High+1, Low-1) if High < Low-1

The summation formulas used are derived from the following reference:

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

Catalogue > 🕡

137
60

$$\sum_{k=1}^{n} \binom{n \cdot (n+1) \cdot (2 \cdot n+1)}{6}$$

$$\sum_{n=1}^{\infty} \left(\frac{1}{n^2}\right) \qquad \frac{\pi^2}{6}$$

$$\sum_{k=4}^{3} (k)$$

$$\sum_{k=4}^{1} \langle k \rangle$$

$$\sum_{k=4}^{1} (k) + \sum_{k=2}^{4} (k)$$

 Σ Int(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [roundValue]) ⇒ value

 $\Sigma Int(NPmt1.NPmt2.amortTable) \Rightarrow value$

Amortization function that calculates the sum of the interest during a specified range of payments.

NPmt1 and NPmt2 define the start and end boundaries of the payment range.

N, I, PV, Pmt, FV, PpY, CpY, and PmtAtare described in the table of TVM arguments, page 192.

- If you omit Pmt, it defaults to Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt).
- If you omit FV, it defaults to FV=0.
- The defaults for PpY, CpY, and PmtAtare the same as for the TVM functions.

roundValue specifies the number of decimal places for rounding. Default=2.

 Σ Int(NPmt1,NPmt2,amortTable) calculates the sum of the interest based on amortization table amort Table. The amort Table argument must be a matrix in the form described under amortTbl(), page 8.

Note: See also $\Sigma Prn()$, below, and **Bal()**, page 17.

 Σ Int(1,3,12,4.75,20000,,12,12) -213.48

tbl:=amortTbl(12,12,4.75,20000,,12,12)				
	0	0.	0.	20000.
	1	-77.49	-1632.43	18367.6
	2	-71.17	-1638.75	16728.8
	3	-64.82	$^{-}1645.1$	15083.7
	4	-58.44	-1651.48	13432.2
	5	-52.05	-1657.87	11774.4
	6	-45.62	-1664.3	10110.1
	7	-39.17	-1670.75	8439.32
	8	-32.7	-1677.22	6762.1
	9	-26.2	-1683.72	5078.38
	10	-19.68	-1690.24	3388.14
	11	-13.13	-1696.79	1691.35
[12	-6.55	-1703.37	-12.02
$\Sigma Int(1,3,tbl)$				-213.48

$\Sigma Prn()$ Catalogue > 🕮

 $\Sigma Prn(NPmt1, NPmt2, N, I, PV, [Pmt],$ [FV], [PpY], [CpY], [PmtAt], [roundValue]) \Rightarrow value

 $\Sigma Prn(NPmt1, NPmt2, amortTable) \Rightarrow$ value

Amortization function that calculates the sum of the principal during a specified range of payments.

 Σ Prn(1,3,12,4.75,20000,,12,12) -4916.28 NPmt1 and NPmt2 define the start and end boundaries of the payment range.

N, I, PV, Pmt, FV, PpY, CpY, and PmtAtare described in the table of TVM arguments, page 192.

- If you omit Pmt, it defaults to Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt).
- If you omit FV, it defaults to FV=0.
- The defaults for PpY, CpY, and PmtAtare the same as for the TVM functions.

roundValue specifies the number of decimal places for rounding. Default=2.

 Σ Prn(NPmt1,NPmt2,amortTable) calculates the sum of the principal paid based on amortization table amort Table. The *amortTable* argument must be a matrix in the form described under amortTbl(), page 8.

Note: See also Σ Int(), above, and Bal(), page 17.

tbl:=amortTbl(12,12,4.75,20000,,12,12)			
0	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	$^{-}58.44$	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	$^{-45.62}$	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14

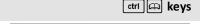
_12	-6.55	-1703.37	-12.02
$\Sigma Prn(1,3,tbl)$			-4916.28

11 -13.13 -1696.79 1691.35

(indirection)

varNameString

Refers to the variable whose name is varNameString. This lets you use strings to create variable names from within a function.



#("x"&"v"&"z") xyz

Creates or refers to the variable xyz.

$10 \rightarrow r$	10
"r" → s1	"r"
#s1	10

Returns the value of the variable (r) whose name is stored in variable s1.

E (scientific notation)

℡ key

mantissaEexponent

Enters a number in scientific notation. The number is interpreted as $mantissa \times 10^{exponent}$.

Hint: If you want to enter a power of 10 without causing a decimal value result, use 10^integer.

Note: You can insert this operator from the computer keyboard by typing @E. for example, type 2.3@E4 to enter 2.3E4.

23000.	23000.
2300000000.+4.1E15	4.1E15
3·10 ⁴	30000

g (gradian)

1 kev

 $Exprlg \Rightarrow expression$

 $Listlg \Rightarrow list$

 $Matrix lg \Rightarrow matrix$

In Degree, Gradian or Radian mode:

 $\frac{\cos(50^{9})}{\cos(\{0,100^{9},200^{9}\})} \frac{\sqrt{2}}{\{1,0,1\}}$

This function gives you a way to specify a gradian angle while in the Degree or Radian mode.

In Radian angle mode, multiplies Expr1 by $\pi/200$.

In Degree angle mode, multiplies Expr1 by g/100.

In Gradian mode, returns $\mathit{Expr1}$ unchanged.

Note: You can insert this symbol from the computer keyboard by typing @g.

r(radian)

1 kev

 $Exprlr \Rightarrow expression$

 $List Ir \Rightarrow list$

 $Matrix lr \Rightarrow matrix$

In Degree, Gradian or Radian angle mode:

$$\frac{\cos\left(\frac{\pi}{4^r}\right)}{\cos\left(\frac{\pi}{4^r} - \frac{\pi}{4^r}\right)} \frac{\sqrt{2}}{2}$$

This function gives you a way to specify a radian angle while in Degree or Gradian mode.

In Degree angle mode, multiplies the argument by $180/\pi$.

In Radian angle mode, returns the argument unchanged.

In Gradian mode, multiplies the argument by $200/\pi$.

Hint: Use r if you want to force radians in a function definition regardless of the mode that prevails when the function is used.

Note: You can insert this symbol from the computer keyboard by typing @r.

° (degree)

1 kev

 $Exprl^{\circ} \Rightarrow expression$

 $List1^{\circ} \Rightarrow list$

 $Matrix 1^{\circ} \Rightarrow matrix$

This function gives you a way to specify a degree angle while in Gradian or Radian mode.

In Radian angle mode, multiplies the argument by $\pi/180$.

In Degree angle mode, returns the argument unchanged.

In Gradian angle mode, multiplies the argument by 10/9.

Note: You can insert this symbol from the computer keyboard by typing @d.

In Degree, Gradian or Radian angle mode:

$$\cos(45^\circ)$$
 $\frac{\sqrt{2}}{2}$

In Radian angle mode:

Note: To force an approximate result,

Handheld: Press ctrl enter. Windows®: Press Ctrl+Enter Macintosh®: Press #+Fnter iPad®: Hold enter, and select ≈ .

$$\overline{\cos\left\{\left\{0, \frac{\pi}{4}, 90^{\circ}, 30.12^{\circ}\right\}\right\}} \\
\left\{1., 0.707107, 0., 0.864976\right\}$$

°, ', " (degree/minute/second)



ctri 🕮 kevs

 $dd^{\circ}mm'ss.ss'' \Rightarrow expression$

In Degree angle mode:

°, ', " (degree/minute/second)

ctrl 🕮 keys

dd A positive or negative number mm A non-negative number ss.ss A non-negative number

25°13'17.5"	25.2215
25°30'	51
	2

Returns dd+(mm/60)+(ss.ss/3600).

This base-60 entry format lets you:

- Enter an angle in degrees/minutes/seconds without regard to the current angle mode.
- Enter time as hours/minutes/seconds.

Note: Follow ss. with two apostrophes ("), not a quote symbol (").

∠ (angle)

ctrl 🕮 kevs

 $[Radius, \angle \theta_Angle] \Rightarrow vector$ (polar input)

 $[Radius, \angle \theta_Angle, Z_Coordinate] \Rightarrow vector$ (cylindrical input)

 $[Radius, \angle \theta_Angle, \angle \theta_Angle]$ ⇒ vector (spherical input)

Returns coordinates as a vector depending on the Vector Format mode setting: rectangular, cylindrical, or spherical.

Note: You can insert this symbol from the computer keyboard by typing @<.

 $(Magnitude \angle Angle) \Rightarrow complex Value$ (polar input)

Enters a complex value in $(r \angle \theta)$ polar form. The Angle is interpreted according to the current Angle mode setting.

In Radian mode and vector format set to: rectangular

cylindrical

$$\begin{bmatrix}
5 & \angle 60^{\circ} & \angle 45^{\circ}
\end{bmatrix} \qquad \boxed{\frac{5 \cdot \sqrt{2}}{2}} \quad \angle \frac{\pi}{3} \quad \frac{5 \cdot \sqrt{2}}{2}$$

spherical

$$\begin{bmatrix} 5 & \angle 60^{\circ} & \angle 45^{\circ} \end{bmatrix} \qquad \begin{bmatrix} 5 & \angle \frac{\pi}{3} & \angle \frac{\pi}{4} \end{bmatrix}$$

In Radian angle mode and Rectangular complex format:

$$\frac{1}{5+3\cdot i - \left(10 \angle \frac{\pi}{4}\right)} \qquad 5-5\cdot \sqrt{2} + \left(3-5\cdot \sqrt{2}\right)\cdot i$$

Note: To force an approximate result,

Handheld: Press ctrl enter.
Windows®: Press Ctrl+Enter.
Macintosh®: Press #Enter.
iPad®: Hold enter, and select = .

' (prime)

?!▶ key

variable ' variable ''

Enters a prime symbol in a differential equation. A single prime symbol denotes a 1st-order differential equation, two prime symbols denote a 2nd-order, and so on.

deSolve
$$v''=y^{-\frac{1}{2}}$$
 and $y(0)=0$ and $y'(0)=0,t,y$

$$\frac{3}{2 \cdot y^{\frac{4}{3}}}=t$$

_ (underscore as an empty element)

See "Empty (Void) Elements," page 248.

_ (underscore as unit designator)

tri L keys

9.84252·_ft

Expr_Unit

Designates the units for an *Expr*. All unit names must begin with an underscore.

You can use pre-defined units or create your own units. For a list of pre-defined units, open the Catalogue and display the Unit Conversions tab. You can select unit names from the Catalogue or type the unit names directly.

Variable

When Variable has no value, it is treated as though it represents a complex number. By default, without the $_$, the variable is treated as real.

If Variable has a value, the _ is ignored and Variable retains its original data type.

Note: You can store a complex number to a variable without using _ . However, for best results in calculations such as cSolve() and cZeros(), the is recommended.

Note: You can find the conversion symbol,

 \blacktriangleright , in the Catalogue. Click $\left| \int \Sigma \right|$, and then click **Maths Operators**.

Assuming z is undefined:

3·_m▶_ft

real(z)	z
real(z_)	$real(z_{-})$
imag(z)	0
$imag(z_{_})$	$imag(z_{_})$

► (convert)		ctrl 🖾 keys
$Expr_Unit1 \triangleright _Unit2 \Rightarrow Expr_Unit2$	3·_m▶_ft	9.84252·_ft

► (convert)



Converts an expression from one unit to another.

The _ underscore character designates the units. The units must be in the same category, such as Length or Area.

For a list of pre-defined units, open the Catalogue and display the Unit Conversions tab:

- You can select a unit name from the list.
- You can select the conversion operator,
 from the top of the list.

You can also type unit names manually. To type "_" when typing unit names on the handheld, press ctrl ___.

Note: To convert temperature units, use tmpCnv() and $\Delta tmpCnv()$. The \blacktriangleright conversion operator does not handle temperature units.

10^()

Catalogue > 🗐

10^ (Expr1) \Rightarrow expression

10^ (Listl) $\Rightarrow list$

Returns 10 raised to the power of the argument.

For a list, returns 10 raised to the power of the elements in *List I*.

10^(squareMatrix 1**)** \Rightarrow squareMatrix

Returns 10 raised to the power of squareMatrix I. This is not the same as calculating 10 raised to the power of each element. For information about the calculation method, refer to cos().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

10 ^{1.5}	31.6228
$10^{\{0,-2,2,a\}}$	$\left\{1, \frac{1}{100}, 100, 10^{a}\right\}$

^-1 (reciprocal)

ctri 🕮 kevs

 $Expr1 \land -1 \Rightarrow expression$

 $List1 \land -1 \Rightarrow list$

Returns the reciprocal of the argument.

For a list, returns the reciprocals of the elements in List 1.

 $squareMatrix1 \land -1 \Rightarrow squareMatrix$

Returns the inverse of *squareMatrix1*.

squareMatrix1 must be a non-singular square matrix.

$(3.1)^{-1}$	0.322581	
$\{a,4,-0.1,x,-2\}^{-1}$	$\left\{\frac{1}{a}, \frac{1}{4}, -10., \frac{1}{x}, \frac{-1}{2}\right\}$	

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \qquad \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix}$$

$$\begin{bmatrix}
1 & 2 \\
a & 4
\end{bmatrix}^{-1} & \frac{-2}{a-2} & \frac{1}{a-2} \\
\frac{a}{2 \cdot (a-2)} & \frac{-1}{2 \cdot (a-2)}$$

| (constraint operator)

Expr | BooleanExpr1[and BooleanExpr2]...

Expr | BooleanExpr1[orBooleanExpr2]...

The constraint ("|") symbol serves as a binary operator. The operand to the left of | is an expression. The operand to the right of | specifies one or more relations that are intended to affect the simplification of the expression. Multiple relations after | must be joined by logical "and" or "or" operators.

The constraint operator provides three basic types of functionality:

- Substitutions
- · Interval constraints
- Exclusions

Substitutions are in the form of an equality, such as x=3 or y=sin(x). To be most effective, the left side should be a simple variable. $Expr \mid Variable = value$ will substitute value for every occurrence of Variable in Expr.

x+1 x=3	4
$x+y x=\sin(y)$	$\sin(y)+y$
$x+y \sin(y)=x$	x+y

$$x^{3}-2 \cdot x+7 \rightarrow f(x) \qquad Done$$

$$f(x)|x=\sqrt{3} \qquad \sqrt{3+7}$$

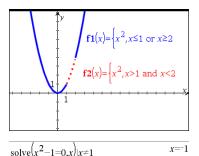
$$(\sin(x))^{2}+2 \cdot \sin(x)-6|\sin(x)=d \qquad d^{2}+2 \cdot d-6$$

| (constraint operator)

ctrl 🕮 keys

Interval constraints take the form of one or more inequalities joined by logical "and" or "or" operators. Interval constraints also permit simplification that otherwise might be invalid or not computable.

$solve(x^2-1=0,x) x>0 \text{ and } x<2$	<i>x</i> =1
$\sqrt{x} \cdot \sqrt{\frac{1}{x}} x>0$	1
$\sqrt{x} \cdot \sqrt{\frac{1}{x}}$	$\sqrt{\frac{1}{x}} \cdot \sqrt{x}$



Exclusions use the "not equals" (/= or \(\neq \)) relational operator to exclude a specific value from consideration. They are used primarily to exclude an exact solution when using cSolve(), cZeros(), fMax(), fMin(), solve(), zeros(), and so on.

ightarrow (store)		ctrl var key
$Expr \rightarrow Var$	$\frac{\pi}{4} \rightarrow myvar$	<u>π</u>
$List \rightarrow Var$	$\frac{4}{2 \cdot \cos(x) \to y I(x)}$	Done
$Matrix \rightarrow Var$	$\frac{2 \cdot \cos(x) \rightarrow y f(x)}{\left\{1,2,3,4\right\} \rightarrow lst5}$	{1,2,3,4}
$Expr \rightarrow Function(Param1,)$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
$List \rightarrow Function(Param1,)$	"Hello" $\rightarrow str1$	"Hello"

 $Matrix \rightarrow Function(Param1,...)$

If the variable Var does not exist, creates it and initializes it to Expr, List, or Matrix.

If the variable Var already exists and is not locked or protected, replaces its contents with Expr, List, or Matrix.

→ (store) ctrl var key

Hint: If you plan to do symbolic computations using undefined variables, avoid storing anything into commonly used, one-letter variables such as a, b, c, x, y, z, and so on.

Note: You can insert this operator from the keyboard by typing =: as a shortcut. For example, type pi/4 =: myvar.

:= (assign)		ctrl [III] keys
Var := Expr	$myvar := \frac{\pi}{4}$	<u>π</u>
Var := List	$\frac{4}{yI(x):=2\cdot\cos(x)}$	Done
Var := Matrix	$\frac{y_{1(x),-2\cdot\cos(x)}}{lst5:=\{1,2,3,4\}}$	{1,2,3,4}
Function(Param1,) := Expr	$natg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
Function(Param1,) := List	str1:="Hello"	"Hello"

Function(Param1,...) := Matrix

If variable Var does not exist, creates Var and initializes it to Expr, List, or Matrix.

If Var already exists and is not locked or protected, replaces its contents with Expr, List, or Matrix.

Hint: If you plan to do symbolic computations using undefined variables, avoid storing anything into commonly used, one-letter variables such as a, b, c, x, y, z, and so on.

© (comment)

© [*text*]

© processes text as a comment line, allowing you to annotate functions and programs that you create.

© can be at the beginning or anywhere in the line. Everything to the right of ©, to the end of the line, is the comment.

Note for entering the example: For instructions on entering multi-line programme and function definitions, refer to the Calculator section of your product guidebook.

Define g(n)=Func

© Declare variables

Local i.result

result:=0

For i,1,n,1 ©Loop n times

result:=result+i²

EndFor

Return result

EndFunc

Done

g(3)14

0 B keys, 0 H keys 0b, 0h

0b binaryNumber Oh hexadecimal Number

Denotes a binary or hexadecimal number, respectively. To enter a binary or hex number, you must enter the 0b or 0h prefix regardless of the Base mode. Without a prefix, a number is treated as decimal (base 10).

Results are displayed according to the Base mode.

In Dec base mode:

0b10+0hF+10 27

In Bin base mode:

0b10+0hF+100b11011

In Hex base mode:

0b10+0hF+10 0h1B

TI-Nspire[™] CX II - Draw Commands

This is a supplemental document for the TI-Nspire™ Reference Guide and the TI-Nspire™ CAS Reference Guide, All TI-Nspire™ CX II commands will be incorporated and published in version 5.1 of the TI-Nspire™ Reference Guide and the TI-Nspire™ CAS Reference Guide.

Graphics Programming

New commands have been added on TI-Nspire™ CX II Handhelds and TI-Nspire™ desktop applications for graphics programming.

The TI-Nspire™ CX II Handhelds will switch into this graphics mode while executing graphics commands and switch back to the context in which the program was executed after completion of the program.

The screen will display "Running..." in the top bar while the program is being executed. It will show "Finished" when the program completes. Any key-press will transition the system out of the graphics mode.

- The transition to graphics mode is triggered automatically when one of the Draw (graphics) commands is encountered during execution of the TI-Basic program.
- This transition will only happen when executing a program from calculator; in a document or calculator in scratchpad.
- The transition out of graphics mode happens upon termination of the program.
- The graphics mode is only available on the TI-Nspire™ CX II Handhelds and the desktop TI-Nspire™ CX II Handhelds view. This means it is not available in the computer document view or PublishView (.tnsp) on the desktop nor on iOS.
 - If a graphics command is encountered while executing a TI-Basic program from the incorrect context, an error message is displayed and the TI-Basic program is terminated.

Graphics Screen

The graphics screen will contain a header at the top of the screen that cannot be written to by graphics commands.

The graphics screen drawing area will be cleared (colour = 255,255,255) when the graphics screen is initialized.

Graphics Screen	Default
Height	212
Width	318
Colour	white: 255,255,255

Default View and Settings

- The status icons in the top bar (battery status, press-to-test status, network indicator etc.) will not be visible while a graphics program is running.
- Default drawing colour: Black (0.0.0)
- Default pen style normal, smooth
 - Thickness: 1 (thin), 2 (normal), 3 (thickest)
 - Style: 1 (smooth), 2 (dotted), 3 (dashed)
- All drawing commands will use the current colour and pen settings; either default values or those which were set via TI-Basic commands.
- Text font is fixed and cannot be changed.
- Any output to the graphics screen will be drawn within a clipping window which is the size of the graphics screen drawing area. Any drawn output that extends outside of this clipped graphics screen drawing area will not be drawn. No error message will be displayed.
- All x,y coordinates specified for drawing commands are defined such that 0,0 is at the top left corner of the graphics screen drawing area.
 - Exceptions:
 - **DrawText** uses the coordinates as the bottom left corner of the bounding box for the text.
 - **SetWindow** uses the bottom left corner of the screen
- All parameters for the commands can be provided as expressions that evaluate to a number which is then rounded to the nearest integer.

Graphics Screen Errors Messages

If the validation fails, an error message will display.

Error Message	Description	View
Error Syntax	If the syntax checker finds any syntax errors, it displays an error message and tries to position the cursor near the first error so you can correct it.	Error Syntax
Error Too few arguments	The function or command is missing one or more arguments	Error Too few arguments The function or command is missing one or more arguments. OK
Error Too many arguments	The function or command contains and excessive number of arguments and cannot be evaluated.	Error Too many arguments The function or command contains an excessive number of arguments and cannot be evaluated. OK
Error Invalid data type	An argument is of the wrong data type.	Error Invalid data type An argument is of the wrong data type. OK

Invalid Commands While in Graphics Mode

Some commands are not allowed once the program switches to graphics mode. If these commands are encountered while in graphics mode an error will be displayed and the program will be terminated.

Disallowed Command	Error Message
Request	Request cannot be executed in graphics mode
RequestStr	RequestStr cannot be executed in graphics mode
Text	Text cannot be executed in graphics mode

The commands that print text to the calculator - disp and dispAt - will be supported commands in the graphics context. The text from these commands will be sent to the Calculator screen (not on Graphics) and will be visible after the program exits and the system switches back to the Calculator app

Clear	Catalogue > [[]] CXII
Clear x, y, width, height	Clear
Clears entire screen if no parameters are specified.	Clears entire screen
If x , y , $width$ and $height$ are specified, the rectangle defined by the parameters will be cleared.	Clear 10,10,100,50 Clears a rectangle area with top left corner on (10, 10) and with width 100, height 50

DrawArc

Catalogue > 🔯

DrawArc x, y, width, height, startAngle, arcAngle

Draw an arc within the defined bounding rectangle with the provided start and arc angles.

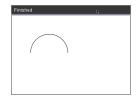
x, y: upper left coordinate of bounding rectangle width, height: dimensions of bounding rectangle The "arc angle" defines the sweep of the arc.

These parameters can be provided as expressions that evaluate to a number which is then rounded to the nearest integer.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



See Also: FillArc

DrawCircle

Catalogue > 🗐 CXII

DrawCircle x, y, radius

x, y: coordinate of centre radius: radius of the circle DrawCircle 150,150,40



See Also: FillCircle

DrawLine

Catalogue > 🗐

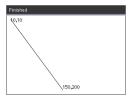
DrawLine x1, y1, x2, y2

Draw a line from x1, y1, x2, y2.

Expressions that evaluate to a number which is then rounded to the nearest integer.

Screen bounds: If the specified coordinates causes any part of the line to be drawn outside of the graphics screen, that part of the line will be clipped and no error message will be displayed.

DrawLine 10,10,150,200



DrawPoly

Catalogue > 🕮

The commands have two variants:

DrawPoly xlist, ylist

or

DrawPoly x1, y1, x2, y2, x3, y3...xn, yn

Note: DrawPoly *xlist*, *ylist*

Shape will connect x1, y1 to x2, y2, x2, y2 to x3, y3

and so on.

Note: DrawPoly *x1*, *y1*, *x2*, *y2*, *x3*, *y3*...*xn*, *yn* xn, yn will **NOT** be automatically connected to x1, y1.

Expressions that evaluate to a list of real floats xlist, ylist

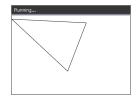
Expressions that evaluate to a single real float x1, y1...xn, yn = coordinates for vertices of polygon

Note: DrawPoly: Input size dimensions (width/height) relative to drawn lines.

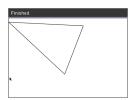
The lines are drawn in a bounding box around the specified coordinate and dimensions such that the actual size of the drawn polygon will be larger than the width and height.

See Also: FillPoly

xlist:={0,200,150,0} ylist:={10,20,150,10} DrawPoly xlist, ylist



DrawPoly 0,10,200,20,150,150,0,10





Catalogue > 🗐

CXII

DrawRect x, y, width, height

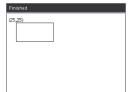
x, y: upper left coordinate of rectangle

width, height: width and height of rectangle (rectangle drawn down and right from starting coordinate).

Note: The lines are drawn in a bounding box around the specified coordinate and dimensions such that the actual size of the drawn rectangle will be larger than the width and height indicated.

See Also: FillRect

DrawRect 25,25,100,50



DrawText

DrawText x, y, exprOrString1 [,exprOrString2]...

x, y: coordinate of text output

Draws the text in *exprOrString* at the specified x, y coordinate location.

The rules for exprOrString are the same as for Disp DrawText can take multiple arguments.

DrawText 50,50,"Hello World"



FillArc

Catalogue > 🕮 FillArc 50,50,100,100,0,180

FillArc x, y, width, height startAngle, arcAngle

x, y: upper left coordinate of bounding rectangle

Draw and fill an arc within the defined bounding rectangle with the provided start and arc angles.

Default fill colour is black. The fill colour can be set by the SetColor command

The "arc angle" defines the sweep of the arc



FillCircle

Catalogue > 🕮

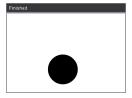
FillCircle x, y, radius

x, y: coordinate of centre

Draw and fill a circle at the specified centre with the specified radius.

Default fill colour is black. The fill colour can be set by the SetColor command.

FillCircle 150,150,40



Here!

FillPoly

Catalogue > 23 CXII

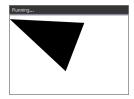
FillPoly xlist, ylist

or

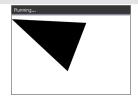
FillPoly x1, y1, x2, y2, x3, y3...xn, yn

Note: The line and colour are specified by SetColor and SetPen

xlist:={0,200,150,0} ylist:={10,20,150,10} FillPoly xlist, ylist



FillPolv 0,10,200,20,150,150,0,10



FillRect

Catalogue > 🗐 CXII

FillRect x, y, width, height

x, y: upper left coordinate of rectangle

width, height: width and height of rectangle

Draw and fill a rectangle with the top left corner at the coordinate specified by (x,y)

Default fill colour is black. The fill colour can be set by the SetColor command

Note: The line and colour are specified by SetColor and SetPen



getPlatform() Catalogue > 📳 getPlatform() getPlatform() "dt" Returns:

"dt" on desktop software applications

[&]quot;hh" on TI-Nspire™ CX handhelds

[&]quot;ios" on TI-Nspire™ CX iPad® app

PaintBuffer Catalogue > 23 **PaintBuffer** UseBuffer For n,1,10 Paint graphics buffer to screen x:=randInt(0,300) This command is used in conjunction with UseBuffer y:=randInt(0,200) to increase the speed of display on the screen when radius:=randInt(10,50) the program generates multiple graphical objects. Wait 0.5 DrawCircle x,y,radius EndFor PaintBuffer This program will display all the 10 circles at once. If the "UseBuffer" command

See Also: UseBuffer

is removed, each circle will be displayed as it is

drawn.

PlotXY x, y, shape

x, y: coordinate to plot shape

shape: a number between 1 and 13 specifying the shape

- 1 Filled circle
- 2 Empty circle
- 3 Filled square
- 4 Empty square
- 5 Cross
- 6 Plus
- 7 Thin
- 8 medium point, solid
- 9 medium point, empty
- 10 larger point, solid
- 11 larger point, empty
- 12 largest point, solid
- 13 largest point, empty

PlotXY 100,100,1

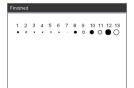


For n,1,13

DrawText 1+22*n,40,n

PlotXY 5+22*n,50,n

EndFor



SetColor

Catalogue > 🕮

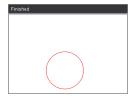
SetColor

Red-value, Green-value, Blue-value

Valid values for red, green and blue are between 0 and 255

Sets the colour for subsequent Draw commands

SetColor 255,0,0 DrawCircle 150,150,100



SetPen

Catalogue > 🗐 CXII

SetPen

thickness, style

thickness: 1 <= thickness <= 3 | 1 is thinnest. 3 is thickest

style: 1 = Smooth, 2 = Dotted, 3 = Dashed

Sets the pen style for subsequent Draw commands

SetPen 3,3

DrawCircle 150,150,50



SetWindow



SetWindow

xMin, xMax, yMin, yMax

Establishes a logical window that maps to the graphics drawing area. All parameters are required.

If the part of drawn object is outside the window, the output will be clipped (not shown) and no error message is displayed.

SetWindow 0,160,0,120

will set the output window to have 0,0 in the bottom left corner with a width of 160 and a height of 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

DrawLine 0,0,100,100

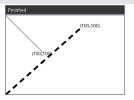


If xmin is greater than or equal to xmax or ymin is greater than or equal to ymax, an error message is shown.

Any objects drawn before a SetWindow command will not be re-drawn in the new configuration.

To reset the window parameters to the default, use:

SetWindow 0,0,0,0



UseBuffer Catalogue > 23

UseBuffer

Draw to an off screen graphics buffer instead of screen (to increase performance)

This command is used in conjunction with PaintBuffer to increase the speed of display on the screen when the program generates multiple graphical objects.

With UseBuffer, all the graphics are displayed only after the next PaintBuffer command is executed.

UseBuffer only needs to be called once in the program i.e. every use of PaintBuffer does not need a corresponding UseBuffer

UseBuffer

For n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

radius:=randInt(10,50)

Wait 0.5

DrawCircle x,y,radius

EndFor

PaintBuffer

This program will display all the 10 circles at once.

If the "UseBuffer" command is removed, each circle will be displayed as it is drawn.

See Also: PaintBuffer

Empty (Void) Elements

When analyzing real-world data, you might not always have a complete data set. TI-Nspire™ CAS Software allows empty, or void, data elements so you can proceed with the nearly complete data rather than having to start over or discard the incomplete cases.

You can find an example of data involving empty elements in the Lists & Spreadsheet chapter, under "Graphing spreadsheet data."

The delVoid() function lets you remove empty elements from a list. The isVoid() function lets you test for an empty element. For details, see delVoid(), page 48, and isVoid(), page 93.

Note: To enter an empty element manually in a maths expression, type "" or the keyword void. The keyword void is automatically converted to a "_" symbol when

Calculations involving void elements

The majority of calculations involving a void input will produce a void result. See special cases below.

gcd(100,_)	_
3+_	_
{5,_,10}-{3,6,9}	{2,_,1}

List arguments containing void elements

The following functions and commands ignore (skip) void elements found in list arguments.

count, countlf, cumulativeSum, freqTable list, frequency, max, mean, median, product, stDevPop, stDevSamp, sum, sumif, varPop and varSamp, as well as regression calculations, OneVar, TwoVar and FiveNumSummary statistics, confidence intervals and stat tests

sum({2,_,3,5,6.6})	16.6
median({1,2,_,_,3})	2
cumulativeSum($\{1,2,4,5\}$)	{1,3,_,7,12}
$ \begin{array}{c} \text{cumulativeSum} \begin{bmatrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{bmatrix} \end{array} $	$\begin{bmatrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{bmatrix}$

SortA and SortD move all void elements within the first argument to the bottom.

{5,4,3,_,1} → list1	{5,4,3,_,1}
$\{5,4,3,2,1\} \rightarrow list2$	{5,4,3,2,1}
SortA list1,list2	Done
list1	{1,3,4,5,_}
list2	{1,3,4,5,2}

List arguments containing void elements

$\{1,2,3,_,5\} \rightarrow list1$	{1,2,3,_,5}
$\{1,2,3,4,5\} \rightarrow list2$	{1,2,3,4,5}
SortD list1,list2	Done
list1	{5,3,2,1,_}
list2	{5,3,2,1,4}

In regressions, a void in an X or Y list introduces a void for the corresponding element of the residual.

11:={1,2,3,4,5}: 12:={2,_,3,5,6.6}	
	{2,_,3,5,6.6}
LinRegMx 11,12	Done
stat.Resid	
{0.434286,_,-0.862857,	-0.011429,0.44}
stat.XReg	{1.,_,3.,4.,5.}
stat.YReg	{2.,_,3.,5.,6.6}
stat.FreqReg	{1., ,1.,1.,1.}

An omitted category in regressions introduces a void for the corresponding element of the residual.

$l1:=\{1,3,4,5\}: l2:=\{2,3,5,6.6\}$	{2,3,5,6.6}
cat:={ "M", "M", "F", "F" }: incl:={	"F"}
	{ "F" }
LinRegMx 11,12,1,cat,incl	Done
stat.Resid	{_,_,0.,0.}
stat.XReg	{_,_,4.,5.}
stat.YReg	{_,_,5.,6.6}
stat.FreqReg	{_,_,1.,1.}

A frequency of 0 in regressions introduces a void for the corresponding element of the residual.

<i>11</i> :={1,3,4,5}: <i>12</i> :={2,3,5,6.6}	{2,3,5,6.6}
LinRegMx 11,12,{1,0,1,1}	Done
stat.Resid { 0.069231,_,-0.27	6923,0.207692}
stat.XReg	{1.,_,4.,5.}
stat.YReg	{2.,_,5.,6.6}
stat.FreqReg	{1.,_,1.,1.}

Shortcuts for Entering Maths Expressions

Shortcuts let you enter elements of maths expressions by typing instead of using the Catalogue or Symbol Palette. For example, to enter the expression $\sqrt{6}$, you can type sqrt(6) on the entry line. When you press [enter], the expression sqrt(6) is changed to $\sqrt{6}$. Some shortrcuts are useful from both the handheld and the computer keyboard. Others are useful primarily from the computer keyboard.

From the Handheld or Computer Keyboard

To enter this:	Type this shortcut:
π	pi
θ	theta
∞	infinity
<u>≤</u>	<=
2	>=
≠	/=
⇒ (logical implication)	=>
dd⇔ (logical double implication, XNOR)	<=>
→ (store operator)	=:
(absolute value)	abs ()
√()	sqrt()
d()	derivative()
<u> </u>	integral()
Σ () (Sum template)	sumSeq()
Π() (Product template)	prodSeq()
sin ⁻¹ (), cos ⁻¹ (),	arcsin(), arccos(),
ΔList()	deltaList()
ΔtmpCnv()	deltaTmpCnv()

From the Computer Keyboard

To enter this:	Type this shortcut:
c1, c2, (constants)	@c1, @c2,
n1, n2, (integer constants)	@n1, @n2,
i (imaginary constant)	@i

To enter this:	Type this shortcut:
e (natural log base e)	@ e
E (scientific notation)	@E
T (transpose)	@t
r (radians)	@r
° (degrees)	@d
g (gradians)	@g
∠ (angle)	@<
▶ (conversion)	@>
Decimal, ▶approxFraction () and so on.	<pre>@>Decimal, @>approxFraction() and so on.</pre>

EOS™ (Equation Operating System) Hierarchy

This section describes the Equation Operating System (EOS™) that is used by the TI-Nspire™ CAS maths and science learning technology. Numbers, variables and functions are entered in a simple, straightforward sequence. EOS™ software evaluates expressions and equations using parenthetical grouping and according to the priorities described below.

Order of Evaluation

Level	Operator
	•
1	Parentheses (), brackets [], braces { }
2	Indirection (#)
3	Function calls
4	Post operators: degrees-minutes-seconds ($^{\circ}$,',"), factorial (!), percentage (%), radian (r), subscript ([]), transpose (T)
5	Exponentiation, power operator (^)
6	Negation (-)
7	String concatenation (&)
8	Multiplication (*), division (/)
9	Addition (+), subtraction (-)
10	Equality relations: equal (=), not equal (\neq or /=), less than (<), less than or equal (\leq or <=), greater than (>), greater than or equal (\geq or >=)
11	Logical not
12	Logical and
13	Logical or
14	xor, nor, nand
15	Logical implication (⇒)
16	Logical double implication, XNOR (\Leftrightarrow)
17	Constraint operator (" ")
18	Store (→)

Parentheses, Brackets and Braces

All calculations inside a pair of parentheses, brackets, or braces are evaluated first. For example, in the expression 4(1+2), EOS™ software first evaluates the portion of the expression inside the parentheses, 1+2, and then multiplies the result, 3, by 4.

The number of opening and closing parentheses, brackets and braces must be the same within an expression or equation. If not, an error message is displayed that

indicates the missing element. For example, (1+2)/(3+4 will display the error message "Missing)."

Note: Because the TI-Nspire™ CAS software allows you to define your own functions, a variable name followed by an expression in parentheses is considered a "function call" instead of implied multiplication. For example a(b+c) is the function a evaluated by b+c. To multiply the expression b+c by the variable a, use explicit multiplication: a* (b+c).

Indirection

The indirection operator (#) converts a string to a variable or function name. For example, #("x"&"y"&"z") creates the variable name xyz. Indirection also allows the creation and modification of variables from inside a programme. For example, if 10→r and "r" \rightarrow s1, then #s1=10.

Post Operators

Post operators are operators that come directly after an argument, such as 5!, 25%, or 60°15' 45". Arguments followed by a post operator are evaluated at the fourth priority level. For example, in the expression 4³!, 3! is evaluated first. The result, 6, then becomes the exponent of 4 to yield 4096.

Exponentiation

Exponentiation (^) and element-by-element exponentiation (.^) are evaluated from right to left. For example, the expression 2^3^2 is evaluated the same as 2^(3^2) to produce 512. This is different from (2^3)^2, which is 64.

Negation

To enter a negative number, press (-) followed by the number. Post operations and exponentiation are performed before negation. For example, the result of -x2 is a negative number, and -92 = -81. Use parentheses to square a negative number such as (-9)2 to produce 81.

Constraint ("|")

The argument following the constraint ("|") operator provides a set of constraints that affect the evaluation of the argument preceding the operator.

TI-Nspire CX II - TI-Basic Programming Features

Auto-indentation in Programming Editor

The TI-Nspire™ program editor now auto-indents statements inside a block command.

Block commands are If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

The editor will automatically prepend spaces to program commands inside a block command. The closing command of the block will be aligned with the opening command.

The example below shows auto-indentation in nested block commands.



Code fragments that are copied and pasted will retain the original indentation.

Opening a program created in an earlier version of the software will retain the original indentation.

Improved Error Messages for TI-Basic

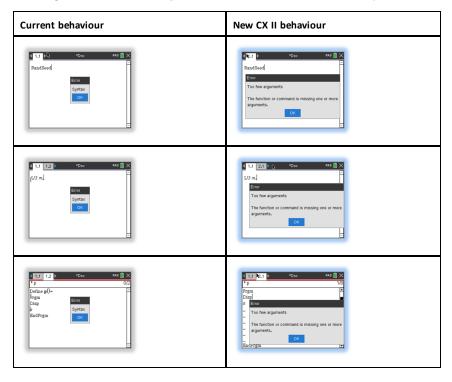
Errors

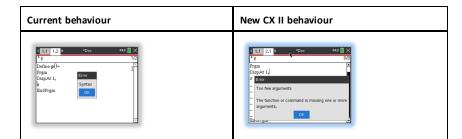
Error Condition	New message
Error in condition statement (If/While)	A conditional statement did not resolve to TRUE or FALSE NOTE: With the change to place the cursor on the line with the error, we no longer need to specify if the error is in an "If" statement or a "While" statement.
Missing EndIf	Expected EndIf but found a different end statement
Missing EndFor	Expected EndFor but found a different end statement
Missing EndWhile	Expected EndWhile but found a different end statement
Missing EndLoop	Expected EndLoop but found a different end statement

Error Condition	New message
Missing EndTry	Expected EndTry but found a different end statement
"Then" omitted after If <condition></condition>	Missing IfThen
"Then" omitted after ElseIf <condition></condition>	Then missing in block: Elself.
When "Then", "Else" and "Elself" were encountered outside of control blocks	Else invalid outside of blocks: IfThenEndIf or TryEndTry
"Elself" appears outside of "IfThenEndIf" block	ElseIf invalid outside of block: IfThenEndIf
"Then" appears outside of "IfEndIf" block	Then invalid outside of block: IfEndIf

Syntax Errors

In case commands that expect one or more arguments are called with an incomplete list of arguments, a "Too few argument error" will be issued instead of "syntax" error





Note: When an incomplete list of arguments is not followed by a comma, the error message is: "too few arguments". This is the same as previous releases.



Constants and Values

The following table lists the constants and their values that are available when performing unit conversions. They can be typed in manually or selected from the Constants list in Utilities > Unit Conversions (Handheld: Press a).

Constant	Name	Value
_c	Speed of light	299792458 _m/_s
_Cc	Coulomb constant	8987551787.3682 _m/_F
_Fc	Faraday constant	96485.33289 _coul/_mol
_g	Acceleration of gravity	9.80665 _m/_s ²
_Gc	Gravitational constant	6.67408E-11 _m ³ /_kg/_s ²
_h	Planck's constant	6.626070040E-34_J_s
k	Boltzmann's constant	1.38064852 E -23_J/_°K
_μ0	Permeability of a vacuum	1.2566370614359E-6_N/_A ²
_μb	Bohr magneton	9.274009994E-24_J_m ² /_Wb
_Me	Electron rest mass	9.10938356E-31_kg
_Μμ	Muon mass	1.883531594E-28_kg
_Mn	Neutron rest mass	1.674927471E-27_kg
_Mp	Proton rest mass	1.672621898E-27_kg
_Na	Avogadro's number	6.022140857E23 /_mol
_q	Electron charge	1.6021766208E-19 _coul
_Rb	Bohr radius	5.2917721067 E -11_m
_Rc	Molar gas constant	8.3144598_J/_mol/_°K
_Rdb	Rydberg constant	10973731.568508/_m
_Re	Electron radius	2.8179403227 E -15 _m
_u	Atomic mass	1.660539040E-27_kg
_Vm	Molar volume	2.2413962 E -2 _m ³ /_mol
_ε0	Permittivity of a vacuum	8.8541878176204E-12_F/_m
_σ	Stefan-Boltzmann constant	5.670367E-8 _W/_m ² /_°K ⁴
_ \$ 0	Magnetic flux quantum	2.067833831E-15 _Wb

Error Codes and Messages

When an error occurs, its code is assigned to variable *errCode*. User-defined programs and functions can examine errCode to determine the cause of an error. For an example of using errCode, See Example 2 under the Try command, page 188.

Note: Some error conditions apply only to TI-Nspire™ CAS products, and some apply only to TI-Nspire™ products.

Error code	Description	
10	A function did not return a value	
20	A test did not resolve to TRUE or FALSE.	
	Generally, undefined variables cannot be compared. For example, the test If a <b a="" b="" cause="" either="" error="" executed.<="" if="" is="" or="" statement="" td="" the="" this="" undefined="" when="" will="">	
30	Argument cannot be a folder name.	
40	Argument error	
50	Argument mismatch	
	Two or more arguments must be of the same type.	
60	Argument must be a Boolean expression or integer	
70	Argument must be a decimal number	
90	Argument must be a list	
100	Argument must be a matrix	
130	Argument must be a string	
140	Argument must be a variable name.	
	Make sure that the name:	
	does not begin with a digit	
	does not contain spaces or special characters	
	does not use underscore or period in invalid manner	
	does not exceed the length limitations	
	See the Calculator section in the documentation for more details.	
160	Argument must be an expression	
165	Batteries too low for sending or receiving	
	Install new batteries before sending or receiving.	
170	Bound	
	The lower bound must be less than the upper bound to define the search interval.	

Error code	Description
180	Break
	The esc or a m key was pressed during a long calculation or during programme execution.
190	Circular definition
	This message is displayed to avoid running out of memory during infinite replacement of variable values during simplification. For example, a+1->a, where a is an undefined variable, will cause this error.
200	Constraint expression invalid
	For example, solve $(3x^2-4=0,x) \mid x<0 \text{ or } x>5 would produce this error message because the constraint is separated by "or" instead of "and."$
210	Invalid Data type
	An argument is of the wrong data type.
220	Dependent limit
230	Dimension
	A list or matrix index is not valid. For example, if the list $\{1, 2, 3, 4\}$ is stored in L1, then L1[5] is a dimension error because L1 only contains four elements.
235	Dimension Error. Not enough elements in the lists.
240	Dimension mismatch
	Two or more arguments must be of the same dimension. For example, [1,2]+[1,2,3] is a dimension mismatch because the matrices contain a different number of elements.
250	Divide by zero
260	Domain error
	An argument must be in a specified domain. For example, rand(0) is not valid.
270	Duplicate variable name
280	Else and Elself invalid outside of IfEndIf block
290	EndTry is missing the matching Else statement
295	Excessive iteration
300	Expected 2 or 3-element list or matrix
310	The first argument of nSolve must be an equation in a single variable. It cannot contain a non-valued variable other than the variable of interest.
320	First argument of solve or cSolve must be an equation or inequality

Error code	Description					
	For example, solve $(3x^2-4,x)$ is invalid because the first argument is not an equation.					
345	Inconsistent units					
350	Index out of range					
360	Indirection string is not a valid variable name					
380	Undefined Ans					
	Either the previous calculation did not create Ans, or no previous calculation was entered.					
390	Invalid assignment					
400	Invalid assignment value					
410	Invalid command					
430	Invalid for the current mode settings					
435	Invalid guess					
440	Invalid implied multiply					
	For example, $x(x+1)$ is invalid; whereas, $x*(x+1)$ is the correct syntax. This is to avoid confusion between implied multiplication and function calls.					
450	Invalid in a function or current expression					
	Only certain commands are valid in a user-defined function.					
490	Invalid in TryEndTry block					
510	Invalid list or matrix					
550	Invalid outside function or programme					
	A number of commands are not valid outside a function or programme. For example, Local cannot be used unless it is in a function or programme.					
560	Invalid outside LoopEndLoop, ForEndFor, or WhileEndWhile blocks					
	For example, the Exit command is valid only inside these loop blocks.					
565	Invalid outside programme					
570	Invalid pathname					
	For example, \var is invalid.					
575	Invalid polar complex					
580	Invalid programme reference					
	Programs cannot be referenced within functions or expressions such as $1+p(x)$ where p is a programme.					

Error code	Description				
600	Invalid table				
605	Invalid use of units				
610	Invalid variable name in a Local statement				
620	Invalid variable or function name				
630	Invalid variable reference				
640	Invalid vector syntax				
650	Link transmission				
	A transmission between two units was not completed. Verify that the connecting cable is connected firmly to both ends.				
665	Matrix not diagonalisable				
670	Low Memory				
	1. Delete some data in this document				
	2. Save and close this document				
	If 1 and 2 fail, pull out and re-insert batteries				
672	Resource exhaustion				
673	Resource exhaustion				
680	Missing (
690	Missing)				
700	Missing "				
710	Missing]				
720	Missing }				
730	Missing start or end of block syntax				
740	Missing Then in the IfEndIf block				
750	Name is not a function or programme				
765	No functions selected				
780	No solution found				
800	Non-real result				
	For example, if the software is in the Real setting, $\sqrt{(-1)}$ is invalid.				
	To allow complex results, change the "Real or Complex" Mode Setting to RECTANGULAR or POLAR.				

Error code	Description
830	Overflow
850	programme not found
	A programme reference inside another programme could not be found in the provided path during execution.
855	Rand type functions not allowed in graphing
860	Recursion too deep
870	Reserved name or system variable
900	Argument error
1	Median-median model could not be applied to data set.
910	Syntax error
920	Text not found
930	Too few arguments
I	The function or command is missing one or more arguments.
940	Too many arguments
	The expression or equation contains an excessive number of arguments and cannot be evaluated.
950	Too many subscripts
955	Too many undefined variables
960	Variable is not defined
	No value is assigned to variable. Use one of the following commands:
	• sto →
	• := • Define
	to assign values to variables.
965	Unlicensed OS
970	Variable in use so references or changes are not allowed
980	Variable is protected
990	Invalid variable name
	Make sure that the name does not exceed the length limitations
1000	Window variables do main

Error code	Description
1010	Zoom
1020	Internal error
1030	Protected memory violation
1040	Unsupported function. This function requires Computer Algebra System. Try TI-Nspire™ CAS.
1045	Unsupported operator. This operator requires Computer Algebra System. Try TI-Nspire™ CAS.
1050	Unsupported feature. This operator requires Computer Algebra System. Try TI-Nspire™ CAS.
1060	Input argument must be numeric. Only inputs containing numeric values are allowed.
1070	Trig function argument too big for accurate reduction
1080	Unsupported use of Ans. This application does not support Ans.
1090	Function is not defined. Use one of the following commands: • Define • := • sto → to define a function.
1100	Non-real calculation For example, if the software is in the Real setting, $\sqrt{(-1)}$ is invalid. To allow complex results, change the "Real or Complex" Mode Setting to RECTANGULAR or POLAR.
1110	Invalid bounds
1120	No sign change
1130	Argument cannot be a list or matrix
1140	Argument error The first argument must be a polynomial expression in the second argument. If the second argument is omitted, the software attempts to select a default.
1150	Argument error The first two arguments must be polynomial expressions in the third argument. If the third argument is omitted, the software attempts to select a default.
1160	Invalid library pathname A pathname must be in the form xxx\yyy, where: The xxx part can have 1 to 16 characters.

Error code	e Description					
	The <i>yyy</i> part can have 1 to 15 characters.					
	See the Library section in the documentation for more details.					
1170	 Invalid use of library pathname A value cannot be assigned to a pathname using Define, :=, or sto →. A pathname cannot be declared as a Local variable or be used as a parameter in a function or programme definition. 					
1180	Invalid library variable name.					
	Make sure that the name: Does not contain a period Does not begin with an underscore Does not exceed 15 characters See the Library section in the documentation for more details.					
1190	Library document not found: Verify library is in the MyLib folder. Refresh Libraries.					
	See the Library section in the documentation for more details.					
1200	 Library variable not found: Verify library variable exists in the first problem in the library. Make sure library variable has been defined as LibPub or LibPriv. Refresh Libraries. 					
	See the Library section in the documentation for more details.					
1210	Invalid library shortcut name. Make sure that the name: Does not contain a period Does not begin with an underscore Does not exceed 16 characters Is not a reserved name See the Library section in the documentation for more details.					
1220	Domain error:					
	The tangentLine and normalLine functions support real-valued functions only.					
1230	Domain error. Trigonometric conversion operators are not supported in Degree or Gradian angle modes.					
1250	Argument Error					

Error code	Description
	Use a system of linear equations.
	Example of a system of two linear equations with variables x and y:
	3x+7y=5
	2y-5x=-1
1260	Argument Error:
	The first argument of nfMin or nfMax must be an expression in a single variable. It cannot contain a non-valued variable other than the variable of interest.
1270	Argument Error
	Order of the derivative must be equal to 1 or 2.
1280	Argument Error
	Use a polynomial in expanded form in one variable.
1290	Argument Error
	Use a polynomial in one variable.
1300	Argument Error
	The coefficients of the polynomial must evaluate to numeric values.
1310	Argument error:
	A function could not be evaluated for one or more of its arguments.
1380	Argument error:
	Nested calls to domain() function are not allowed.

Warning Codes and Messages

You can use the warnCodes() function to store the codes of warnings generated by evaluating an expression. This table lists each numeric warning code and its associated message.

For an example of storing warning codes, see warnCodes(), page 197.

Warning code	Message					
10000	Operation might introduce false solutions.					
10001	Differentiating an equation may produce a false equation.					
10002	Questionable solution					
10003	Questionable accuracy					
10004	Operation might lose solutions.					
10005	cSolve might specify more zeroes.					
10006	Solve may specify more zeroes.					
10007	More solutions may exist. Try specifying appropriate lower and upper bounds and/or a guess.					
	Examples using solve(): solve(Equation, Var=Guess) lowBound <var<upbound solve(equation,="" var="Guess)</td" var) lowbound<var<upbound=""></var<upbound>					
10008	Domain of the result might be smaller than the domain of the input.					
10009	Domain of the result might be larger than the domain of the input.					
10012	Non-real calculation					
10013	∞^0 or undef^0 replaced by 1					
10014	undef^0 replaced by 1					
10015	1^∞ or 1^undef replaced by 1					
10016	1^undef replaced by 1					
10017	Overflow replaced by ∞ or ⁻∞					
10018	Operation requires and returns 64 bit value.					
10019	Resource exhaustion, simplification might be incomplete.					
10020	Trig function argument too big for accurate reduction.					
10021	Input contains an undefined parameter.					

Warning code	Message
	Result might not be valid for all possible parameter values.
10022	Specifying appropriate lower and upper bounds might produce a solution.
10023	Scalar has been multiplied by the identity matrix.
10024	Result obtained using approximate arithmetic.
10025	Equivalence cannot be verified in EXACT mode.
10026	Constraint might be ignored. Specify constraint in the form "\" 'Variable MathTestSymbol Constant' or a conjunct of these forms, for example 'x<3 and x>-12'

General Information

Online Help

education.ti.com/eguide

Select your country for more product information.

Contact TI Support

education.ti.com/ti-cares

Select your country for technical and other support resources.

Service and Warranty Information

education.ti.com/warranty

Select your country for information about the length and terms of the warranty or about product service.

Limited Warranty. This warranty does not affect your statutory rights.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Index		^, power	210
-		_	
-, subtract	207	_, unit designation	227
	207	1	
!		, constraint operator	229
!, factorial	217	,	
п		(asia ata a statica	
", second notation	225	'minute notation', prime	225 227
#			,
		+	
#, indirection #, indirection operator	223 253	+, add	207
	233	=	
%		≠, not equal	214
%, percent	213	≤, less than or equal	215
&		≥, greater than or equal	216 215
&, append	217	=, equal	213
*	21,	π	
*		∏, product	220
*, multiply	208		220
•		Σ	
, dot subtraction	211	Σ(), sum	221
.*, dot multiplication	212	ΣInt() ΣPrn()	222 222
./, dot division	212 212		
.+, dot addition	211	٧	
1		V, square root	220
/, divide	200	∠	
,, divide	209	∠ (angle)	226
:			-
		ſ	
:=, assign	231	ı	
:=, assign	231	, integral	218

1

, convert units	227	10^(), power of ten	228
▶approxFraction()	14		
Base10, display as decimal integer	19	2	
▶Base16, display as hexadecimal	19	2-sample F Test	75
▶Base2, display as binary	17	2 sample rest ::::::::	/3
cos, display in terms of cosine	29	Α	
►Cylind, display as cylindrical vector	42		
DD, display as decimal angle	45	abs(), absolute value	8
Decimal, display result as decimal	45	absolute value	
►DMS, display as		template for	3-4
degree/minute/second	54	add, +	207
▶exp, display in terms of e	63	amortisation table, amortTbl()	8, 17
Grad, convert to gradian angle	85	amortTbl(), amortisation table	8, 17
▶Polar, display as polar vector	131	and, Boolean operator	9
Rad, convert to radian angle	141	angle(), angle	10
▶Rect, display as rectangular vector	144	angle, angle()	10
sin, display in terms of sine	163	ANOVA, one-way variance analysis	10
Sphere, display as spherical vector .	172	ANOVA2way, two-way variance	
		analysis	11
⇒		Ans, last answer	13
⇒, logical implication216,	250	answer (last), Ans	13
, 10510011111111111111111111111111111111	230	append, &	217
\rightarrow		approx(), approximate	13, 15
		approximate, approx()	13, 15
→, store variable	230	approxRational()	14
⇔		arc length, arcLen()	15
~		arccos(), cos ⁻¹ ()	14
⇔, logical double implication217,	250	arccosh(), cosh ⁻¹ ()	14
,		arccot(), cot ⁻¹ ()	14
©		arccoth(), coth ⁻¹ ()	14
©, comment	222	arccsc(), csc ⁻¹ ()	15
©, comment	232	arccsch(), csch ⁻¹ ()	15
0		arcLen(), arc length	15
		arcsec(), sec ⁻¹ ()	15
°, degree notation	225	arcsech(), sech ⁻¹ ()	15
°, degrees/minutes/seconds	225	arcsin(), sin ⁻¹ ()	15
		arcsinh(), sinh ⁻¹ ()	15
0		arctan(), tan ⁻¹ ()	15
Ob, binary indicator	232	arctanh(), tanh ⁻¹ ()	16
Oh, hexadecimal indicator	232	arguments in TVM functions	192
		augment(), augment/concatenate .	16
		augment/concatenate, augment() .	16

average rate of change, avgRC()	16	colNorm(), matrix column norm	26
avgRC(), average rate of change	16	combinations, nCr()	118
		comDenom(), common	
В		denominator	26
binary		comment, ©	232
display, ►Base2	17	common denominator, comDenom	
indicator, 0b	232	()	26
binomCdf()	20, 91	completeSquare(), complete square	27
binomPdf()	20, 91	complex	
Boolean operators	20	conjugate, conj()	28
⇒2	16 250	factor, cFactor()	22
⇔	217	solve, cSolve()	38
and	9	zeros, cZeros()	42
nand	117	conj(), complex conjugate	28
nor	121	constant	
not		in solve()	169
	123	constants	
or	127	in cSolve()	39
xor	198	in cZeros()	43
С		in deSolve()	49
•		in solve()	170
Cdf()	68	in zeros()	201
ceiling(), ceiling	20	shortcuts for	250
ceiling, ceiling()20	-21, 36	constraint operator " "	229
centralDiff()	21	constraint operator, order of	
cFactor(), complex factor	22	evaluation	252
char(), character string	23	construct matrix, constructMat()	28
character string, char()	23	constructMat(), construct matrix	28
characters		convert	0.5
numeric code, ord()	128	►Grad	85
string, char()	23	►Rad	141
charPoly()	23	units	227
χ²2way	23	copy variable or function, CopyVar	29
$\chi^2 Cdf()$	24	correlation matrix, corrMat()	29
$\chi^2 GOF$	24	corrMat(), correlation matrix	29
$\chi^2 Pdf()$	25	cos ⁻¹ , arccosine	31
clear		cos(), cosine	30
error, ClrErr	25	cosh ⁻¹ (), hyperbolic arccosine	32
Clear	236	cosh(), hyperbolic cosine	32
ClearAZ	25	cosine	
ClrErr, clear error	25	display expression in terms of	29
colAugment	26	cosine, cos()	30
colDim(), matrix column dimension	26	cot ⁻¹ (), arccotangent	33
	-	cot(), cotangent	33

cotangent, cot()	33	definite integral	
coth ⁻¹ (), hyperbolic arccotangent	34	template for	6
coth(), hyperbolic cotangent	34	degree notation, °	225
count days between dates, dbd()	44	degree/minute/second display,	
count items in a list conditionally,		►DMS	54
countif()	35	degree/minute/second notation	225
count items in a list, count()	34	delete	
count(), count items in a list	34	void elements from list	48
countif(), conditionally count items	-	deleting	
in a list	35	variable, DelVar	48
cPolyRoots()	36	deltaList()	47
cross product, crossP()	36	deltaTmpCnv()	48
crossP(), cross product	36	DelVar, delete variable	48
csc ⁻¹ (), inverse cosecant	37	delVoid(), remove void elements	48
csc(), cosecant	36	denominator	26
csch ⁻¹ (), inverse hyperbolic cosecant	37	derivative or nth derivative	
csch(), hyperbolic cosecant	37	template for	6
cSolve(), complex solve	38	derivative()	48
cubic regression, CubicReg	40	derivatives	
CubicReg, cubic regression	40	first derivative, d()	218
cumulative sum, cumulativeSum().	41	numeric derivative, nDeriv()	120
cumulativeSum(), cumulative sum	41	numeric derivative, nDerivative(
cycle, Cycle	41)	119
Cycle, cycle	41	deSolve(), solution	49
cylindrical vector display, ►Cylind	42	det(), matrix determinant	51
cZeros(), complex zeros	42	diag(), matrix diagonal	51
ezeros(), complex zeros :::::::	42	dim(), dimension	51
D		dimension, dim()	51
		Disp, display data	52, 156
d(), first derivative	218	DispAt	52
days between dates, dbd()	44	display as	
dbd(), days between dates	44	binary, ►Base2	17
decimal		cylindrical vector, ▶Cylind	42
angle display, ▶DD	45	decimal angle, > DD	45
integer display, ►Base10	19	decimal integer, ►Base10	19
Define	45	degree/minute/second, DMS .	54
Define LibPriv	46	hexadecimal, Base16	19
Define LibPub	47	polar vector, ▶Polar	131
define, Define	45	rectangular vector, ▶Rect	144
Define, define	45	spherical vector, ▶Sphere	172
defining		display data, Disp	52, 156
private function or programme	46	distribution functions	
public function or programme .	47	binomCdf()	20, 91
		binomPdf()	20

invNorm()	91	else, Else	86
invt()	92	ElseIf, else if	59
Invχ²()	92 90	empty (void) elements	248
normCdf()		end	248
normPdf()	123	for, EndFor	72
poissCdf()	123	function, EndFunc	75
	130	if, EndIf	86
poissPdf()	130	loop, EndLoop	108
tCdf()	182	try, EndTry	188
tPdf()	187	while, EndWhile	
χ ² 2way()	23	end function, EndFunc	198
χ²Cdf()	24		75
χ²GOF()	24	end if, EndIf	86
χ²Pdf()	25	end loop, EndLoop	108
divide, /	209	end while, EndWhile	198
domain function, domain()	55	EndTry, end try	188
domain(), domain function	55	EndWhile, end while	198
dominant term, dominantTerm()	55	EOS (Equation Operating System)	252
dominantTerm(), dominant term	55	equal, =	213
dot		Equation Operating System (EOS)	252
addition, .+	211	error codes and messages	258
division, ./	212	errors and troubleshooting	
multiplication, .*	212	clear error, ClrErr	25
power, .^	212	pass error, PassErr	129
product, dotP()	56	euler(), Euler function	60
subtraction,	211	evaluate polynomial, polyEval()	133
dotP(), dot product	56	evaluation, order of	252
draw2	37-239	exact(), exact	62
		exact, exact()	62
E		exclusion with " " operator	229
e exponent		exit, Exit	62
template for	2	Exit, exit	62
e to a power, e^()	57, 63	exp(), e to a power	63
e, display expression in terms of	63	exp list(), expression to list	63
E, exponent	224	expand(), expand	64
e^(), e to a power	57	expand, expand()	64
eff(), convert nominal to effective	37	exponent, E	224
rate	57	exponential regession, ExpReg	65
effective rate, eff()	5 <i>7</i>	exponents	
eigenvalue, eigVl()	58	template for	1
eigenvector, eigVc()	58	expr(), string to expression	65, 106
eigVc(), eigenvector	58	ExpReg, exponential regession	65
eigVI(), eigenvalue	58	expressions	
else if, Elself	59	expression to list, exp►list()	63
Cioc ii, Liocii	29		

F		g, gradians	224
factor(), factor	67	gcd(), greatest common divisor	76
factor, factor()	67	geomCdf()	76
factorial, !	217	geomPdf()	77
fill24		Get 7	7, 242
Fill, matrix fill	69	get/return	
financial functions, tvmFV()	191	denominator, getDenom()	78
financial functions, tvml()	191	number, getNum()	83
financial functions, tvmN()	191	variables injformation,	01 01
financial functions, tvmPmt()		getVarInfo() getDenom(), get/return	81, 84
financial functions, tvmPV()	191	denominator	78
first derivative	192	getKey()	78
	-	getLangInfo(), get/return language	70
template for	5	information	81
FiveNumSummary	69	getLockInfo(), tests lock status of	01
floor(), floor	70	variable or variable group	82
floor, floor()	70	getMode(), get mode settings	82
fMax(), function maximum	70	getNum(), get/return number	83
fMin(), function minimum	71	GetStr	83
For	72	getType(), get type of variable	84
for, For	72	getVarInfo(), get/return variables	04
For, for	72	information	84
format string, format()	72	go to, Goto	85
format(), format string	72	Goto, go to	85
fpart(), function part	73	gradian notation, g	224
fractions		greater than or equal, ≥	216
propFrac	137	greater than, >	215
template for	1	greatest common divisor, gcd()	76
freqTable()	73	groups, locking and unlocking10	
frequency()	74	groups, testing lock status	•
Frobenius norm, norm()	122	groups, testing lock status	82
Func, function	75	Н	
Func, programme function	75		
functions		hexadecimal	
maximum, fMax()	70	display, ►Base16	19
minimum, fMin()	71	indicator, 0h	232
part, fpart()	73	hyperbolic	
programme function, Func	75	arccosine, cosh ⁻¹ ()	32
user-defined	45	arcsine, sinh ⁻¹ ()	166
functions and variables	.5	arctangent, tanh ⁻¹ ()	182
copying	29	cosine, cosh()	32
		sine, sinh()	165

tangent, tanh()	181	language	
		get language information	81
I		Lbl, label	94
identity matrix, identity()	86	lcm, least common multiple	94
identity(), identity matrix	86	least common multiple, lcm	94
if, If	86	left(), left	94
If, if	86	left, left()	94
ifFn()	87	length of string	51
imag(), imaginary part	88	less than or equal, ≤	215
imaginary part, imag()	88	LibPriv	46
ImpDif(), implicit derivative	88	LibPub	47
implicit derivative, Impdif()	88	library	
indefinite integral	00	create shortcuts to objects	95
template for	6	libShortcut(), create shortcuts to	
indirection operator (#)	253	library objects	95
indirection,#	223	limit	0.6
input, Input	88	lim()	96
Input, input	88	limit()	96
inString(), within string	89	template for	6
int(), integer	89	limit() or lim(), limit	96
intDiv(), integer divide	89	linear regression, LinRegAx	97
integer divide, intDiv()	89	linear regression, LinRegBx	96, 98
integer part, iPart()	92	LinRegBx, linear regression	96
integer, int()	92 89	LinRegMx, linear regression	97
integral, f		LinRegtIntervals, linear regression	98
interpolate(), interpolate	218	LinRegtTest	100
inverse cumulative normal	90	linSolve()	101
distribution (invNorm()	91	Δlist(), list difference	102
inverse, ^-1	229	list to matrix, list mat()	102
invF()	90	list, conditionally count items in	35
invNorm(), inverse cumulative	50	list, count items in	34
normal distribution)	91	list > mat(), list to matrix	102
invt()	92	lists	
Invχ²()	90	augment/concatenate,	16
iPart(), integer part	92	augment() cross product, crossP()	36
irr(), internal rate of return		cumulative sum,	30
internal rate of return, irr()	92	cumulative sum()	41
isPrime(), prime test	93	differences in a list, Δlist()	102
isVoid(), test for void	93	dot product, dotP()	56
		empty elements in	248
L		expression to list, exp list()	63
label, Lbl	94	list to matrix, list mat()	102
iddel, LDI	94	matrix to list, mat list()	102
			109

maximum, max()	110	dot division, ./ 21	2
mid-string, mid()	112	dot multiplication, .* 21	2
minimum, min()	113	dot power, .^ 21	2
new, newList()	119	dot subtraction, 21	1
product, product()	136	eigenvalue, eigVl() 5	
sort ascending, SortA	171	eigenvector, eigVc() 5	_
sort descending, SortD	172	filling, Fill	
summation, sum()17		identity, identity()	_
In(), natural logarithm	102	list to matrix, list mat() 10	
LnReg, logarithmic regression	102	lower-upper decomposition, LU 10	
local variable, Local			
local, Local	104		
•	104		-
Local, local variable	104	minimum, min()	
Lock, lock variable or variable group	105	new, newMat()	_
locking variables and variable groups	105	product, product()	-
Log	_	QR factorization, QR	
template for	2	random, randMat() 14	2
logarithmic regression, LnReg	103	reduced row echelon form, rref(
logarithms	102)	
logical double implication, ⇔	217	row addition, rowAdd() 15	
logical implication, ⇒21	6, 250	row dimension, rowDim() 15	3
logistic regression, Logistic	106	row echelon form, ref() 14.	5
logistic regression, LogisticD	107	row multiplication and addition,	
Logistic, logistic regression	106	mRowAdd() 11	
LogisticD, logistic regression	107	row norm, rowNorm() 15	3
loop, Loop	108	row operation, mRow() 11	4
Loop, loop	108	row swap, rowSwap() 15	
LU, matrix lower-upper		submatrix, subMat()177, 179	
decomposition	109	summation, sum()177-17	8
		transpose, T 17	9
M		matrix (1 × 2)	
mat▶list(), matrix to list	109	template for	4
matrices	109	matrix (2 × 1)	
augment/concatenate,		•	4
augment()	16	matrix (2 × 2)	
column dimension, colDim()	26	•	4
column norm, colNorm()	26	matrix (m × n)	
cumulative sum,	20	•	4
cumulativeSum()	41	matrix to list, mat list() 10	9
determinant, det()	51	max(), maximum 11	0
diagonal, diag()	51	maximum, max() 11	0
dimension, dim()	51	mean(), mean 11	0
dot addition, .+	211	mean, mean() 11	0
	211	median(), median 11	1

median, median()	111	nfMin(), numeric function minimum	120
medium-medium line regression,		nInt(), numeric integral	120
MedMed	111	nom), convert effective to nominal	
MedMed, medium-medium line		rate	121
regression	111	nominal rate, nom()	121
mid-string, mid()	112	nor, Boolean operator	121
mid(), mid-string	112	norm(), Frobenius norm	122
min(), minimum	113	normal distribution probability,	
minimum, min()	113	normCdf()	123
minute notation, '	225	normal line, normalLine()	123
mirr(), modified internal rate of		normalLine()	123
return	113	normCdf()	123
mixed fractions, using propFrac()		normPdf()	123
with	137	not equal, ≠	214
mod(), modulo	114	not, Boolean operator	123
mode settings, getMode()	82	nPr(), permutations	123
modes		npv(), net present value	124
setting, setMode()	159	nSolve(), numeric solution	_
modified internal rate of return, mirr			125
()	113	nth root	2
modulo, mod()	114	template fornumeric	2
mRow(), matrix row operation	114	derivative, nDeriv()	120
mRowAdd(), matrix row		derivative, nDerivative()	_
multiplication and addition	115		119
Multiple linear regression t test	116	integral, nInt()	120
multiply, *	208	solution, nSolve()	125
MultReg	115	0	
MultRegIntervals()	115	9	
MultRegTests()	116	objects	
		create shortcuts to library	95
N		one-variable statistics, OneVar	126
nand, Boolean operator	117	OneVar, one-variable statistics	126
natural logarithm, ln()	117	operators	
	102	order of evaluation	252
nCr(), combinations	118	or (Boolean), or	127
nDerivative(), numeric derivative	119	or, Boolean operator	127
negation, entering negative numbers	253	ord(), numeric character code	128
net present value, npv()	125		
new		P	
list, newList()	119	DNPv() restangular v coordinate	420
matrix, newMat()	119	P>Rx(), rectangular x coordinate	129
newList(), new list	119	P•Ry(), rectangular y coordinate	129
newMat(), new matrix	119	pass error, PassErr	129
nfMax(), numeric function		PassErr, pass error	129
maximum	120	Pdf()	73

percent, %	213	programs and programming	
permutations, nPr()	124	clear error, ClrErr	25
piecewise function (2-piece)	124	display I/O screen, Disp	52
template for	2	end try, EndTry	188
piecewise function (N-piece)	_	try, Try	188
template for	3	proper fraction, propFrac	137
piecewise()	130	propFrac, proper fraction	137
poissCdf()	130	propried, proportion	137
poissPdf()	130	Q	
polar	200		
coordinate, R Pr()	141	QR factorization, QR	137
coordinate, R•Pθ()	140	QR, QR factorization	137
vector display, ▶Polar	131	quadratic regression, QuadReg	138
polyCoef()	131	QuadReg, quadratic regression	138
polyDegree()	132	quartic regression, QuartReg	139
polyEval(), evaluate polynomial	133	QuartReg, quartic regression	139
polyGcd()13		-	
polynomials		R	
evaluate, polyEval()	133	R, radian	224
random, randPoly()	143	R Pr(), polar coordinate	141
PolyRoots()	134	R►Pθ(), polar coordinate	140
power of ten, 10^()	228	radian, R	224
power regression,		rand(), random number	141
PowerReg 134, 147-14	8, 184	randBin, random number	142
power, ^	210	randInt(), random integer	142
DawarDag namer regression			
PowerReg, power regression	134		
Prgm, define programme	134 135	randMat(), random matrix	142
Prgm, define programme	135	randMat(), random matrix randNorm(), random norm random	142
Prgm, define programme prime number test, isPrime()	135 93	randMat(), random matrix randNorm(), random norm	142 143
Prgm, define programme prime number test, isPrime() prime, '	135 93 227	randMat(), random matrix randNorm(), random norm random matrix, randMat()	142 143 142 143
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf()	135 93 227 123	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm()	142 143 142 143 143
Prgm, define programme	135 93 227 123 136	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly()	142 143 142 143 143 143
Prgm, define programme	135 93 227 123 136 136	randMat(), random matrix	142 143 142 143 143 143 143
Prgm, define programme	135 93 227 123 136 136 220	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial	142 143 142 143 143 143 143
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, $\Pi()$ template for	135 93 227 123 136 136 220 5	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp()	142 143 142 143 143 143 143 143
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product()	135 93 227 123 136 136 220 5	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed	142 143 142 143 143 143 143 143 143
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product() programmes and programming	135 93 227 123 136 136 220 5	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real	142 143 143 143 143 143 143 143 143 144
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product() programmes and programming display I/O screen, Disp programming define programme, Prgm	135 93 227 123 136 136 220 5 136	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real real, real()	142 143 143 143 143 143 143 143 144 144
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product() programmes and programming display I/O screen, Disp programming	135 93 227 123 136 136 220 5 136	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real real, real() reciprocal, ^-1	142 143 143 143 143 143 143 143 144 144 229
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product() programmes and programming display I/O screen, Disp programming define programme, Prgm	135 93 227 123 136 136 220 5 136	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real real, real() rectangular-vector display, *Rect	142 143 143 143 143 143 143 143 144 144 229 144
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() product(), product product, ∏() template for product, product() programmes and programming display I/O screen, Disp programming define programme, Prgm display data, Disp pass error, PassErr programs	135 93 227 123 136 136 220 5 136 156	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real real, real() reciprocal, ^-1 rectangular-vector display, *Rect rectangular x coordinate, P*Rx()	142 143 143 143 143 143 143 143 144 144 229 144 129
Prgm, define programme prime number test, isPrime() prime, ' probability densiy, normPdf() prodSeq() product(), product product, ∏() template for product, product() programmes and programming display I/O screen, Disp programming define programme, Prgm display data, Disp pass error, PassErr	135 93 227 123 136 136 220 5 136 156	randMat(), random matrix randNorm(), random norm random matrix, randMat() norm, randNorm() number seed, RandSeed polynomial, randPoly() random sample randPoly(), random polynomial randSamp() RandSeed, random number seed real(), real real, real() rectangular-vector display, *Rect	142 143 143 143 143 143 143 143 144 144 229 144

ref(), row echelon form	145	rref(), reduced row echelon form	154
RefreshProbeVars	146		
regressions		S	
cubic, CubicReg	40	sec ⁻¹ (), inverse secant	155
exponential, ExpReg	65	sec(), secant	154
linear regression, LinRegAx	97	sech ⁻¹ (), inverse hyperbolic secant	155
linear regression, LinRegBx	96, 98	sech(), hyperbolic secant	
logarithmic, LnReg	103	second derivative	155
Logistic	106	template for	6
logistic, Logistic	107	second notation, "	225
medium-medium line, MedMed	111	seq(), sequence	_
MultReg	115	seqGen()	156
power regression,			157
PowerReg134, 147-14	8, 184	seqn()sequence, seq()	
quadratic, QuadReg	138		
quartic, QuartReg	139	series (), series	158
sinusoidal, SinReg	166	series, series()	158
remain(), remainder	147	set mode, setMode()	150
remainder, remain()	147	setMode(), set mode	159
remove		settings, get current	
void elements from list	48		
Request	147	shift(), shift	161
RequestStr	148	shift, shift()	161
result		sign(), sign	162
display in terms of cosine	29	sign, sign()	162
display in terms of e	63	simult(), simultaneous equations	163
display in terms of sine	163	simultaneous equations, simult()	163
result values, statistics	175	sin ⁻¹ (), arcsine	
results, statistics	174	sin(), sine	164
return, Return	149	sine	
Return, return	149	display expression in terms of	163
right(), right	149	sine, sin()	164
right, right()27, 60, 90, 149-150	0. 197	sinh ⁻¹ (), hyperbolic arcsine	
rk23(), Runge Kutta function	150	sinh(), hyperbolic sine	165
rotate(), rotate	151	SinReg, sinusoidal regression	
rotate, rotate()	151	sinusoidal regression, SinReg	
round(), round	153	solution, deSolve()	49
round, round()	153	solve(), solve	168
row echelon form, ref()	145	solve, solve()	168
rowAdd(), matrix row addition	153	SortA, sort ascending	171
rowDim(), matrix row dimension	153	SortD, sort descending	172
rowNorm(), matrix row norm	153	sorting	
rowSwap(), matrix row swap	154	ascending, SortA	171
	134	descending, SortD	172

spherical vector display, ▶Sphere	172	right, right() - 27, 60, 90, 149-150, 19
sqrt(), square root	173	rotate, rotate()
square root		shift, shift()
template for	1	string to expression, expr() 65, 106
square root, v()	, 220	using to create variable names . 253
standard deviation, stdDev() - 175-176	, 195	within, InString
stat.results	174	student-t distribution probability,
stat.values	175	tCdf() 183
statistics		student-t probability density, tPdf() 187
combinations, nCr()	118	subMat(), submatrix177, 179
factorial, !	217	submatrix, subMat()177, 179
mean, mean()	110	substitution with " " operator 229
median, median()	111	subtract, 20
one-variable statistics, OneVar	126	sum of interest payments 222
permutations, nPr()	124	sum of principal payments 222
random norm, randNorm()	143	sum(), summation 17
random number seed,		sum, ∑()
RandSeed	143	template for
standard deviation, stdDev		sumIf() 178
()175-176	, 195	summation, sum()
two-variable results, Two Var	192	sumSeq()
variance, variance()	195	system of equations (2-equation)
stdDevPop(), population standard		template for
deviation	175	system of equations (N-equation)
stdDevSamp(), sample standard	476	template for
deviation	176	
Stop command	176	Т
store variable (→)	230	that trat
storing	224	t test, tTest
symbol, &	231	T, transpose
string dimension, dim()	51	tan ⁻¹ (), arctangent
length	51	tan(), tangent
string(), expression to string	_	tangent line, tangentLine() 18:
strings	177	tangent, tan()
append, &	217	tangentLine()
character code, ord()	128	tanh ⁻¹ (), hyperbolic arctangent 183
character string, char()	23	tanh(), hyperbolic tangent 183
expression to string, string()	23 177	Taylor polynomial, taylor() 182
format, format()		taylor(), Taylor polynomial 182
formatting	72 72	tCdf(), studentt distribution
indirection, #	72	probability
	223	tCollect(), trigonometric collection 183
left, left()	94	templates
mid-string, mid()	112	absolute value 3-4

definite integral	6	trigonometric expansion, tExpand()	183
derivative or nth derivative	6	Try, error handling command	188
e exponent	2	tTest, t test	189
exponent	1	tTest_2Samp, two-sample t test	190
first derivative	5	TVM arguments	192
fraction	1	tvmFV()	191
indefinite integral	6	tvml()	191
limit	6	tvmN()	191
Log	2	tvmPmt()	191
matrix (1 × 2)	4	tvmPV()	192
matrix (2 × 1)	4	two-variable results, Two Var	192
matrix (2 × 2)	4	Two Var, two-variable results	192
matrix (m × n)	4		
nth root	2	U	
piecewise function (2-piece)	2	underscore,	227
piecewise function (N-piece)	3	unit vector, unitV()	227
product, ∏()	5	units	194
second derivative	6	convert	227
square root	1	unitV(), unit vector	194
sum, ∑()	5	unLock, unlock variable or variable	194
system of equations (2-		group	195
equation)	3	unlocking variables and variable	
system of equations (N-		groups	195
equation)	3	user-defined functions	45
test for void, isVoid()	93	user-defined functions and	
Test_2S, 2-sample F test	75	programs	46-47
tExpand(), trigonometric expansion	183	.,	
Text command	184	V	
time value of money, Future Value _	191	variable	
time value of money, Interest	191	creating name from a character	
time value of money, number of		string	253
payments	191	variable and functions	
time value of money, payment	101	copying	29
amounttime value of money, present value .	191	variables	
tInterval, t confidence interval	192	clear all single-letter	25
tinterval, t comidence interval	185	delete, DelVar	48
confidence interval	185	local, Local	104
ΔtmpCnv()	187	variables, locking and unlocking 82, 10	
tmpCnv()18		variance, variance()	195
tPdf(), studentt probability density	187	varPop()	195
trace()	188	varSamp(), sample variance	195
transpose, T	179	vectors cross product, crossP()	20
trigonometric collection, tCollect().	183	cross product, cross*()	36
	103		

cylindrical vector display,	
►Cylind	42
dot product, dotP()	56
unit, unitV()	194
void elements	248
void elements, remove	48
void, test for	93
W	
Wait command	196
warnCodes(), Warning codes	197
warning codes and messages	266
when(), when	197
when, when()	197
while, While	198
While, while	198
with,	229
within string, inString()	89
x	
x², square	211
XNOR	217
xor, Boolean exclusive or	198
Z	
zeroes(), zeroes	199
zeroes, zeroes()	199
zInterval, z confidence interval	201
zInterval_1Prop, one-proportion z	
confidence interval	202
zInterval_2Prop, two-proportion z	
confidence interval	202
zInterval_2Samp, two-sample z confidence interval	203
zTest	203
zTest_1Prop, one-proportion z test .	204
zTest 2Prop, two-proportion z test	204
zTest_2Samp, two-sample z test	206