Texas Instruments

TI-Nspire™/TI-Nspire™ CX

参考指南

本指导手册适用于 TI-Nspire™ 软件 3.2 版本。要获得最新版本的文档，请访问 *education.ti.com/guides*。

重要信息

除非在程序附带的《许可证》中明示声明，否则 Texas Instruments 不对任何程序或书面材料做出任何明示或暗示担保，包括但不限于对某个特定用途的适销性和适用性的暗示担保，并且这些材料均以“原样”提供。任何情况下，Texas Instruments 对因购买或使用这些材料而蒙受特殊、附带、偶然或连带损失的任何人都不承担任何责任。无论采用何种赔偿方式，Texas Instruments 的唯一且排他性义务不得超出本程序许可证规定的数额。此外，对于任何其他方因使用这些材料而提起的任何类型的索赔，Texas Instruments 概不负责。

许可证

请查阅安装于 **C:\Program Files\TI Education\<TI-Nspire™ Product Name>\license** 中的完整许可证。

目录

表达式模板

A

B

C

D

O

P

Q

R

S

T

TI-Nspire™ 参考指南

本指南列出了可用于计算数学表达式的模板、函数、命令和运算符。

表达式模板

表达式模板提供了用标准数学符号输入数学表达式的简单方法。插入模板时，模板将在输入行中显示，您可以在小方块位置输入元素。此时光标将显示您可以输入的元素。

用箭头键或按 tab 将光标移动到每个元素的位置，然后键入该元素的值或表达式。按 enter 或 ctrl enter 以计算表达式。

分数模板 ctrl ÷ 键

注意：另请参阅 **/（除）**（第 120 页）。

示例：

$\frac{12}{8 \cdot 2} \quad \frac{3}{4}$

指数模板 ^ 键

注意：键入第一个值 按 ^ 然后键入指数。要使光标返回到基准行 请按右箭头 (▶)。

注意：另请参阅 **^（乘方）**（第 120 页）。

示例：

$2^3 \quad 8$

平方根模板 ctrl x^2 键

注意：另请参阅 **√()（平方根）**（第 126 页）。

示例：

$\sqrt{4} \quad 2$

$\sqrt{\{9,16,4\}} \quad \{3,4,2\}$

N 次方根模板 ctrl ^ 键

注意：另请参阅 **root()**（第 86 页）。

示例：

$\sqrt[3]{8} \quad 2$

$\sqrt[3]{\{8,27,15\}} \quad \{2,3,2.46621\}$

***e* 指数模板** e^x 键

自然指数 *e* 求乘方

注意：另请参阅 **e^()**（第 32 页）。

示例：

$e^1 \quad 2.71828182846$

对数模板

ctrl 10^x 键

$\log_{\square}(\square)$

计算指定底数的对数。默认情况下　若底数为 10　则省略底数。

注意：另请参阅 **log()**（ 第 57 页 ）。

示例：

$\log_4(2.) \quad 0.5$

分段函数模板（ 2 段式 ）

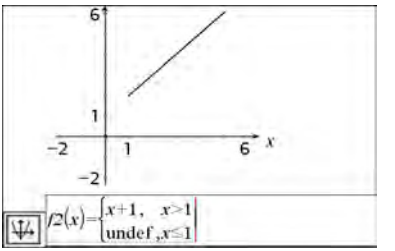
目录 >

$\begin{cases}\square, & \square \\ \square, & \square\end{cases}$

可让您创建二段式分段函数的表达式和条件。- 要添加分段请单击模板　然后重复使用该模板。

注意：另请参阅 **piecewise()**（ 第 73 页 ）。

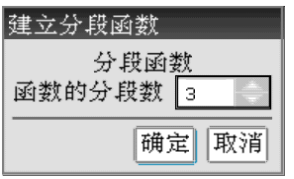
示例：



分段函数模板（ N 段式 ）

目录 >

可让您创建 *N* 段式分段函数的表达式和条件。- 提示输入 *N* 值。



注意：另请参阅 **piecewise()**（ 第 73 页 ）。

示例：
请参阅分段函数模板（ 2 段式 ）示例。

二元方程组模板

目录 >

$\begin{cases}\square \\ \square\end{cases}$

创建二元线性方程组。要向现有的方程组添加一个方程　请单击模板　然后重复使用该模板。

注意：另请参阅 **system()**（ 第 101 页 ）。

示例：

$\text{solve}\left(\begin{cases}x+y=0 \\ x-y=5\end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=\frac{-5}{2}$

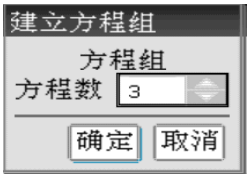
$\text{solve}\left(\begin{cases}y=x^2-2 \\ x+2\cdot y=-1\end{cases}, x, y\right)$

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

N 元方程组模板

目录 >

可让您创建 *N* 元线性方程组。提示输入 *N* 值。



注意：另请参阅 **system()**（ 第 101 页 ）。

示例：
请参阅方程组模板（ 二元方程 ）示例。

绝对值模板

目录 >

$|\square|$ **注意：**另请参阅 **abs()**（ 第 6 页 ）。

示例：

$$\left|\left\{2,-3,4,-4^3\right\}\right| \qquad \{2,3,4,64\}$$

dd°mm'ss.ss'' 模板

目录 >

$\square°\square'\square''$

可让您以 **dd°mm'ss.ss''** 格式输入角度　其中 **dd** 为十进制度数　**mm** 为分数　**ss.ss** 为秒数。

示例：

30°15'10" 0.528011

矩阵模板 (2 x 2)

目录 >

$\begin{bmatrix}\square & \square\\ \square & \square\end{bmatrix}$

创建 2 x 2 矩阵。

示例：

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\cdot 5 \qquad \begin{bmatrix}5 & 10\\ 15 & 20\end{bmatrix}$$

矩阵模板 (1 x 2)

目录 >

$\begin{bmatrix}\square & \square\end{bmatrix}$。

示例：

$$\text{crossP}\left(\begin{bmatrix}1 & 2\end{bmatrix},\begin{bmatrix}3 & 4\end{bmatrix}\right) \qquad \begin{bmatrix}0 & 0 & -2\end{bmatrix}$$

矩阵模板 (2 x 1)

目录 >

$\begin{bmatrix}\square\\ \square\end{bmatrix}$

示例：

$$\begin{bmatrix}5\\ 8\end{bmatrix}\cdot 0.01 \qquad \begin{bmatrix}0.05\\ 0.08\end{bmatrix}$$

矩阵模板 (m x n)

目录 >

您收到指定行数和列数的提示后　模板将显示。

注意：如果您创建有许多行和列的矩阵　可能需要较长时间才会显示。

示例：

$$\text{diag}\begin{pmatrix}4 & 2 & 6\\ 1 & 2 & 3\\ 5 & 7 & 9\end{pmatrix} \qquad \begin{bmatrix}4 & 2 & 9\end{bmatrix}$$

求和模板 (Σ)

目录 >

$$\sum_{\square=\square}^{\square}(\square)$$

注意：另请参阅 **Σ() (sumSeq)**（ 第 127 页 ）。

示例：

$$\sum_{n=3}^{7}(n) \qquad 25$$

乘积模板 (Π)

目录 >

$$\prod_{\square=\square}^{\square}(\square)$$

注意：另请参阅 **Π() (prodSeq)**（ 第 127 页 ）。

示例：

$$\prod_{n=1}^{5}\left(\frac{1}{n}\right) \qquad \frac{1}{120}$$

一阶导数模板

目录 >

$$\frac{d}{d\square}(\square)$$

一阶导数模板还可用于计算某一数值点的一阶导数（ 使用自动微分方法 ）。

注意：另请参阅 **d() (导数)**（ 第 126 页 ）。

示例：

$$\frac{d}{dx}(|x|)\Big|_{x=0} \qquad \text{undef}$$

二阶导数模板

目录 >

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

二阶导数模板还可用于计算某一数值点的二阶导数数值（ 使用自动微分方法 ）。

注意：另请参阅 **d()（ 导数)**（ 第 126 页 ）。

示例：

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

定积分模板

目录 >

$$\int_{\square}^{\square} \square \, d\square$$

定积分模板可用于计算定积分数值（ 使用与 nInt() 相同的方法 ）。

注意：另请参阅 **nInt()**（ 第 67 页 ）。

示例：

$$\int_{0}^{10} x^2 \, dx \qquad 333.333$$

字母顺序列表

名称非字母的项（ 例如 **+**、 **!** 和 **>** ）在本节的结尾处列出（ 从第 **118** 页开始 ）。除非另行指定，本节中的所有示例都将在默认的复位模式下执行，并且所有变量都假定为未定义。

A

abs()

目录 >

abs(*Value1***)** ⇒ *值*

abs(*List1***)** ⇒ *数组*

abs(*Matrix1***)** ⇒ *矩阵*

返回自变量的绝对值。

注意：另请参阅**绝对值模板**（ 第 3 页 ）。

如果自变量为复数　将返回该复数的模数。

$$\left|\left\{\frac{\pi}{2},\frac{-\pi}{3}\right\}\right| \qquad \{1.5708,1.0472\}$$

$$|2-3\cdot i| \qquad 3.60555$$

amortTbl()

目录 >

amortTbl(*NPmt***,***N***,***I***,***PV***,** [*Pmt*]**,** [*FV*]**,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**,** [*roundValue*]**)** ⇒ *矩阵*

分期偿还函数将返回一个矩阵作为一组 TVM 自变量的分期偿还表。

NPmt 是要添加至该表的支付次数。该表从第一次支付开始。

N　*I*　*PV*、*Pmt*、*FV*、*PpY*、*CpY* 和 *PmtAt* 在 TVM 自变量表中有介绍（ 第 109 页 ）。

- 如果您省略 *Pmt*　则使用其默认值 *Pmt*=**tvmPmt(***N***,***I***,***PV***,***FV***,***PpY***,***CpY***,***PmtAt***)**。
- 如果您省略 *FV*　则使用其默认值 *FV*=0。
- *PpY*　*CpY* 和 *PmtAt* 的默认值与用于 TVM 函数的值相同。

roundValue 指定四舍五入的小数位数。默认保留两位小数。

结果矩阵中的列顺序如下：支付次数　利息支付金额　本金支付金额和结余。

第 *n* 行中显示的结余为第 *n* 次支付后的结余。

您可以使用该输出矩阵作为其他分期偿还函数 **ΣInt()** 和 **ΣPrn()**（ 第 128 页 ）以及 **bal()**（ 第 12 页 ）的输入矩阵。

amortTbl(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

and

目录 >

BooleanExpr1 **and** *BooleanExpr2* ⇒ *布尔表达式*。

BooleanList1 **and** *BooleanList2* ⇒ *布尔数组*

BooleanMatrix1 **and** *BooleanMatrix2* ⇒ *布尔矩阵*

返回 **true** 或 **false**　或者原始输入的简化形式。

and

目录 >

Integer1 **and** *Integer2* ⇒ *整数*

使用 **and** 操作逐位比较两个实整数。在内部运算中，两个整数都将转换为带符号的 64 位二进制数字。当相应位进行比较时，如果两个位值均为 1，则结果为 1，否则结果为 0。返回的值代表位结果，将根据 Base 模式显示。

您可以输入任何进位制的整数。对于按二进制或十六进制输入的整数，您必须分别使用 0b 或 0h 前缀。不带前缀的整数都将被视为十进制（基数为 10）。

在 Hex 模式下：

0h7AC36 and 0h3D5F → 0h2C16

重要信息：零，非字母 O。

在 Bin 模式下：

0b100101 and 0b100 → 0b100

在 Dec 模式下：

37 and 0b100 → 4

注意：二进制输入最多可为 64 位（不包括 0b 前缀）。十六进制输入最多可为 16 位。

angle()

目录 >

angle(*Value1***)** ⇒ *值*

返回自变量的角度（自变量代表复数）。

在 Degree 角度模式下：

$\text{angle}(0+2\cdot i)$ → 90

在 Gradian 角度模式下：

$\text{angle}(0+3\cdot i)$ → 100

在 Radian 角度模式下：

$\text{angle}(1+i)$ → 0.785398

$\text{angle}(\{1+2\cdot i, 3+0\cdot i, 0-4\cdot i\})$ → $\{1.10715, 0., -1.5708\}$

angle(*List1***)** ⇒ *数组*

angle(*Matrix1***)** ⇒ *矩阵*

返回一个数组或矩阵，其元素为 *List1* 或 *Matrix1* 中各元素的角度，将每个元素均视为代表二维直角坐标点的复数处理。

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

ANOVA

目录 >

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

进行单因素方差分析，比较 2 个到 20 个总体的平均值。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

对于数据：Flag=0，对于统计：*Flag*=1

输出变量	说明
stat.F	F 统计值
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	组的自由度
stat.SS	组的平方和
stat.MS	组的均值平方

输出变量	说明
stat.dfError	误差的自由度
stat.SSError	误差的平方和
stat.MSError	误差的均值平方
stat.sp	合并标准差
stat.xbarlist	数组输入平均值
stat.CLowerList	每个输入数组平均值的 95% 置信区间
stat.CUpperList	每个输入数组平均值的 95% 置信区间

ANOVA2way 目录 >

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

计算双因素方差分析　比较 2 个到 10 个总体的平均值。结果摘要存储在 *stat.results* 变量中。(请参阅第 98 页。)

块的*行水平*=0

双因素的*行水平*=2,3,...,*Len*-1　其中 *Len*= 长度(*列表 1*) = 长度(*列表 2*) =　= 长度(*列表 10*)且 *Len* / 行水平　∈ {2,3,　}

输出：块设计

输出变量	说明
stat.F	列因素的 F 统计
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	列因素的自由度
stat.SS	列因素的平方和
stat.MS	列因素的均值平方
stat.FBlock	因素的 F 统计
stat.PValBlock	可拒绝零假设的最小概率
stat.dfBlock	因素的自由度
stat.SSBlock	因素的平方和
stat.MSBlock	因素的均值平方
stat.dfError	误差的自由度
stat.SSError	误差的平方和
stat.MSError	误差的均值平方
stat.s	误差的标准差

COLUMN FACTOR 输出

输出变量	说明
stat.Fcol	列因素的 F 统计
stat.PValCol	列因素的概率值
stat.dfCol	列因素的自由度
stat.SSCol	列因素的平方和
stat.MSCol	列因素的均值平方

ROW FACTOR 输出

输出变量	说明
stat.FRow	行因素的 F 统计
stat.PValRow	行因素的概率值
stat.dfRow	行因素的自由度
stat.SSRow	行因素的平方和
stat.MSRow	行因素的均值平方

INTERACTION 输出

输出变量	说明
stat.FInteract	交互的 F 统计
stat.PValInteract	交互的概率值
stat.dfInteract	交互的自由度
stat.SSInteract	交互的平方和
stat.MSInteract	交互的均值平方

ERROR 输出

输出变量	说明
stat.dfError	误差的自由度
stat.SSError	误差的平方和
stat.MSError	误差的均值平方
s	误差的标准差

Ans

ctrl (–) 键

Ans ⇒ *值*

返回最近计算的表达式的结果。

56	56
56+4	60
60+4	64

approx()

目录 >

approx(*Value1***)** ⇒ *数值*

在可能的情况下　无论当前的 **Auto or Approximate** 是何种模式　都以十进制的形式返回自变量的估计值。

此运算等同于输入自变量并按下 ctrl enter。

approx(1/3)	0.333333
approx({1/3, 1/9})	{0.333333, 0.111111}
approx({sin(π), cos(π)})	{0., -1.}
approx([√2 √3])	[1.41421 1.73205]
approx([1/3 1/9])	[0.333333 0.111111]

approx(*List1***)** ⇒ *数组*

approx(*Matrix1***)** ⇒ *矩阵*

在可能的情况下　返回一个数组或*矩阵*　其元素均以十进制数字表示。

approx({sin(π), cos(π)})	{0., -1.}
approx([√2 √3])	[1.41421 1.73205]

▶approxFraction()

目录 >

Value **▶approxFraction([***Tol***])** ⇒ *值*

List **▶approxFraction([***Tol***])** ⇒ *数组*

Matrix **▶approxFraction([***Tol***])** ⇒ *矩阵*

使用公差 *Tol* 以分数形式返回输入值。如果 *Tol* 省略　则使用 5.E-14 作为公差。

注意：您可以通过在计算机键盘上键入 **@>approxFraction(...)** 插入此函数。

1/2 + 1/3 + tan(π)	0.833333
0.83333333333333▶approxFraction(5.E-14)	5/6
{π, 1.5}▶approxFraction(5.E-14)	{5419351/1725033, 3/2}

approxRational()

目录 >

approxRational(*Value*[**,** *Tol*]**)** ⇒ *值*

approxRational(*List*[**,** *Tol*]**)** ⇒ *数组*

approxRational(*Matrix*[**,** *Tol*]**)** ⇒ *矩阵*

使用公差 *Tol*. 以分数形式返回自变量。如果 *Tol* 省略　则使用 5.E-14 作为公差。

approxRational(0.333, 5·10^-5)	333/1000
approxRational({0.2, 0.33, 4.125}, 5.E-14)	{1/5, 33/100, 33/8}

arccos()

请参阅 cos⁻¹()（第 20 页）。

arccosh() 请参阅 $\cosh^{-1}()$（第 21 页）。

arccot() 请参阅 $\cot^{-1}()$（第 22 页）。

arccoth() 请参阅 $\coth^{-1}()$（第 22 页）。

arccsc() 请参阅 $\csc^{-1}()$（第 24 页）。

arccsch() 请参阅 $\text{csch}^{-1}()$（第 24 页）。

arcsec() 请参阅 $\sec^{-1}()$（第 89 页）。

arcsech() 请参阅 $\text{sech}^{-1}()$（第 89 页）。

arcsin() 请参阅 $\sin^{-1}()$（第 94 页）。

arcsinh() 请参阅 $\sinh^{-1}()$（第 95 页）。

arctan() 请参阅 $\tan^{-1}()$（第 102 页）。

arctanh() 请参阅 $\tanh^{-1}()$（第 103 页）。

augment() 目录 >

augment(*List1*, *List2*) ⇒ *数组*

返回将 *List2* 附加到 *List1* 末尾组成的新数组。

augment(*Matrix1*, *Matrix2*) ⇒ *矩阵*

返回将 *Matrix2* 附加到 *Matrix1* 组成的新矩阵。使用 "," 字符时　两个矩阵的行维数必须相同　并且 *Matrix2* 作为新的列附加到 *Matrix1*。此运算不会更改 *Matrix1* 或 *Matrix2*。

$\text{augment}(\{\{1,-3,2\},\{5,4\}\)$ $\{1,-3,2,5,4\}$

$\begin{bmatrix}1&2\\3&4\end{bmatrix}\rightarrow m1$ $\begin{bmatrix}1&2\\3&4\end{bmatrix}$

$\begin{bmatrix}5\\6\end{bmatrix}\rightarrow m2$ $\begin{bmatrix}5\\6\end{bmatrix}$

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

avgRC()

目录 >

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

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

avgRC(*List1***, *Var* [=*Value*] [, *Step*])** ⇒ *数组*

avgRC(*Matrix1***, *Var* [=*Value*] [, *Step*])** ⇒ *矩阵*

返回前向差商（ 平均变化率 ）。

Expr1 可以是用户定义的函数名（ 请参阅 **Func** ）。

指定 *值*之后 该值会覆盖之前的所有变量分配或变量的所有当前 "|" 代入值。

Step 为步长值。如果 *Step* 省略 则使用其默认值 0.001。

请注意 函数 **centralDiff()** 功能与之类似 只是使用中心差商。

$x:=2$ → 2

$\text{avgRC}(x^2-x+2,x)$ → 3.001

$\text{avgRC}(x^2-x+2,x,.1)$ → 3.1

$\text{avgRC}(x^2-x+2,x,3)$ → 6

B

bal()

目录 >

bal(*NPmt***,*N*,*I*,*PV* ,[*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*])** ⇒ *值*

bal(*NPmt***,*amortTable*)** ⇒ *值*

计算指定支付后预定结余的分期偿还函数。

N、*I*、*PV*、*Pmt*、*FV*、*PpY*、*CpY* 和 *PmtAt* 在 TVM 自变量表中有介绍（ 第 109 页 ）。

NPmt 指定支付次数 您希望在该次支付后计算数据。

N I PV、*Pmt*、*FV*、*PpY*、*CpY* 和 *PmtAt* 在 TVM 自变量表中有介绍（ 第 109 页 ）。

- 如果您省略 *Pmt* 则使用其默认值 *Pmt*=**tvmPmt(***N*,*I*,*PV*,*FV*,*PpY*,*CpY*,*PmtAt***)**。
- 如果您省略 *FV* 则使用其默认值 *FV*=0。
- *PpY* *CpY* 和 *PmtAt* 的默认值与用于 TVM 函数的值相同。

roundValue 指定四舍五入的小数位数。默认保留两位小数。

bal(*NPmt***,*amortTable***)** 根据分期偿还表 *amortTable* 计算支付次数 *NPmt* 后的结余。 *amortTable* 自变量必须为 **amortTbl()**（ 第 6 页 ）下所介绍形式的矩阵。

注意：另请参阅 **ΣInt()** 和 **ΣPrn()**（ 第 128 页 ）。

$\text{bal}(5,6,5.75,5000,,12,12)$ → 833.11

$tbl:=\text{amortTbl}(6,6,5.75,5000,,12,12)$ →

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

$\text{bal}(4,tbl)$ → 1674.27

▶Base2

目录 >

Integer1 **▶Base2** ⇒ *整数*

注意：您可以通过在计算机键盘上键入 **@>Base2** 插入此运算符。

将 *Integer1* 转换为二进制数字。二进制或十六进制数字始终分别带有 0b 或 0h 前缀。

256▶Base2 → 0b100000000

0h1F▶Base2 → 0b11111

▶Base2

目录 >

零（ 非字母 O ）后跟 b 或 h。

0b *二进制数字*
0h *十六进制数字*

二进制数字最多可为 64 位。十六进制数字最多可为 16 位。

不带前缀的 *Integer1* 将被视为十进制 (base 10)。不论 Base 模式如何 结果都将显示为二进制。

负数将显示为 " 二进制补码 " 形式。例如

−1 显示为
0hFFFFFFFFFFFFFFFF（ 在 Hex 模式下 ）
0b111...111（ 64 个 1 ）（ 在 Binary 模式下 ）

-2^{63} 显示为
0h8000000000000000（ 在 Hex 模式下 ）
0b100...000（ 63 个 0 ）（ 在 Binary 模式下 ）

如果您输入的十进制整数超出带符号的 64 位二进制形式的范围 可使用对称的模数运算将该值纳入合理的范围。考虑以下超出范围的值的示例。

2^{63} 变为 -2^{63} 并显示为
0h8000000000000000（ 在 Hex 模式下 ）
0b100...000（ 63 个 0 ）（ 在 Binary 模式下 ）

2^{64} 变为 0 并显示为
0h0（ 在 Hex 模式下 ）
0b0（ 在 Binary 模式下 ）

$-2^{63} - 1$ 变为 $2^{63} - 1$ 并显示为
0h7FFFFFFFFFFFFFFF（ 在 Hex 模式下 ）
0b111...111（ 64 个 1 ）（ 在 Binary 模式下 ）

▶Base10

目录 >

Integer1 **▶Base10** ⟹ *整数*

注意： 您可以通过在计算机键盘上键入 **@>Base10** 插入此运算符。

将 *Integer1* 转换为十进制 (base 10) 数字。二进制或十六进制条目必须始终分别带有 0b 或 0h 前缀。

0b *二进制数字*
0h *十六进制数字*

零（ 非字母 O ）后跟 b 或 h。

二进制数字最多可为 64 位。十六进制数字最多可为 16 位。

不带前缀的 *Integer1* 将被视为十进制。不论进位制模式如何 结果都将以十进制显示。

0b10011▶Base10	19
0h1F▶Base10	31

▶Base16

目录 >

Integer1 **▶Base16** ⟹ *整数*

注意：您可以通过在计算机键盘上键入 **@>Base16** 插入此运算符。

将 *Integer1* 转换为十六进制数字。二进制或十六进制数字始终分别带有 0b 或 0h 前缀。

0b *二进制数字*
0h *十六进制数字*

零（非字母 O）后跟 b 或 h。

二进制数字最多可为 64 位。十六进制数字最多可为 16 位。

不带前缀的 *Integer1* 将被视为十进制 (base 10)。不论进位制模式如何，结果将显示为十六进制。

如果您输入的十进制整数对于带符号的 64 位二进制形式来说过大，可使用对称的模数运算将该值纳入合理的范围。更多信息，请参阅 **▶Base2**（第 12 页）。

256▶Base16	0h100
0b111100001111▶Base16	0hF0F

binomCdf()

目录 >

binomCdf(*n*,*p***)** ⟹ *数值*

binomCdf(*n*,*p*,*lowBound*,*upBound***)** ⟹ 如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*；如果 *lowBound* 和 *upBound* 适数组，则结果为 *数组*

binomCdf(*n*,*p*,*upBound***)** for P(0≤X≤*upBound*) ⟹ 如果 *upBound* 是数值，则结果为 *数值*；如果 *upBound* 是数组，则结果为 *数组*

计算 *n* 次尝试的离散二项式分布累积概率以及每次尝试的成功概率 *p*。

对于 P(X ≤ *upBound*)，设置 *lowBound*=0

binomPdf()

目录 >

binomPdf(*n*,*p***)** ⟹ *数值*

binomPdf(*n*,*p*,*XVal***)** ⟹ 如果 *XVal* 是数值，则结果为 *数值*；如果 *XVal* 是数组，则结果为 *数组*

计算 *n* 次尝试的离散二项式分布概率以及每次尝试的成功概率 *p*。

C

ceiling()

目录 >

ceiling(*Value1***)** ⟹ *值*

返回 ≥ 自变量的最接近的整数。

自变量可以是实数，也可以是复数。

注意：另请参阅 **floor()**。

ceiling(.456)	1.

ceiling()

目录 >

ceiling(*List1***)** ⇒ *数组*

ceiling(*Matrix1***)** ⇒ *矩阵*

返回每个元素向上取整的数组或矩阵。

$\text{ceiling}\left\{\left\{-3.1,1,2.5\right\}\right\}$ $\left\{-3.,1,3.\right\}$

$\text{ceiling}\begin{bmatrix} 0 & -3.2\cdot i \\ 1.3 & 4 \end{bmatrix}$ $\begin{bmatrix} 0 & -3.\cdot i \\ 2. & 4 \end{bmatrix}$

centralDiff()

目录 >

centralDiff(*Expr1***,***Var* [=*Value*][,*Step*]**)** ⇒ *表达式*

centralDiff(*Expr1***,***Var* [,*Step*]**)|***Var*=*Value* ⇒ *表达式*

centralDiff(*Expr1***,***Var* [=*Value*][,*List*]**)** ⇒ *数组*

centralDiff(*Expr1***,***Var* [=*Value*][,*List*]**)** ⇒ *数组*

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

返回使用中心差商公式计算得出的数值导数。

指定*值*之后　该值会覆盖之前的所有变量分配或变量的所有当前 "|" 代入值。

Step 为步长值。如果 *Step* 省略　则使用其默认值 0.001。

使用 *List1* 或 *Matrix1* 时　运算将通过数组中的值或矩阵元素来映射。

注意：另请参阅 **avgRC()**。

$\text{centralDiff}\left(\cos(x),x\right)\left|x=\frac{\pi}{2}\right.$ -1.

char()

目录 >

char(*Integer***)** ⇒ *字符*

返回一个字符串　其中包含手持设备字符集中编号为 *Integer* 的字符。 *Integer* 的有效范围是 0–65535。

$\text{char}(38)$ "&"

$\text{char}(65)$ "A"

$\chi^2$2way

目录 >

$\chi^2$2way *obsMatrix*

chi22way *obsMatrix*

计算观测矩阵 *obsMatrix* 中双向计数表关联性的 χ^2 检验。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关矩阵中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.χ^2	卡方统计：sum（ 实际值 - 预计值 $)^2$ 预计值
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	卡方统计的自由度
stat.ExpMat	预期元素计数表的矩阵　假定为零假设
stat.CompMat	元素卡方统计计算值的矩阵

χ^2Cdf()

目录 >

χ^2Cdf(*lowBound,upBound,df***)** ⇒ 如果 *lowBound* 和 *upBound* 适数组，则结果为 *数值*；如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

chi2Cdf(*lowBound,upBound,df***)** ⇒ 如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*；如果 *lowBound* 和 *upBound* 适数组，则结果为 *数组*

计算指定自由度 *df* *lowBound* 与 *upBound* 之间的 χ^2 分布概率。

对于 P(*X* ≤ *upBound*) 设置为 *lowBound*=0

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

χ^2GOF

目录 >

χ^2GOF *obsList,expList,df*
chi2GOF *obsList,expList,df*

执行检验以确认样本数据来自于符合指定分布的总体。*obsList* 是计数的数组 必须包含整数。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.χ^2	卡方统计：sum（ 实际值 - 预计值 $)^2$ 预计值
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	卡方统计的自由度
stat.CompList	元素卡方统计贡献值

χ^2Pdf()

目录 >

χ^2Pdf(*XVal,df***)** ⇒ 如果 *XVal* 是数值，则结果为 *数值*；如果 *XVal* 是数组，则结果为 *数组*

chi2Pdf(*XVal,df***)** ⇒ 如果 *XVal* 是数值，则结果为 *数值*；如果 *XVal* 是数组，则结果为 *数组*

计算 *XVal* 为指定值时 指定自由度 *df* 的 χ^2 分布概率密度函数 (pdf)。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

ClearAZ

目录 >

ClearAZ

清除当前问题空间中的所有单字符变量。

如果有一个或多个变量锁定　此命令将显示错误消息并仅删除未锁定变量。请参阅 **unLock**（ 第 111 页 ）。

$5 \rightarrow b$	5
b	5
ClearAZ	*Done*
b	"Error: Variable is not defined"

ClrErr

目录 >

ClrErr

清除错误状态并将系统变量 *errCode* 设置为零。

Try...Else...EndTry 块的 **Else** 语句应使用 **ClrErr** 或 **PassErr**。如果要处理或忽略错误　请使用 **ClrErr**。如果不知道如何处理错误　请使用 **PassErr** 将其发送到下一个错误处理句柄。如果没有其他未完成的 **Try...Else...EndTry** 错误处理句柄　错误对话框将正常显示。

注意：另请参阅第 73 页的 **PassErr** 和第 106 页的 **Try**。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

有关 **ClrErr** 的示例　请参阅 **Try** 命令下的示例 2（ 第 106 页 ）。

colAugment()

目录 >

colAugment(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

返回将 *Matrix2* 附加到 *Matrix1* 组成的新矩阵。两个矩阵的列维数必须相等　并且 *Matrix2* 作为新的列附加到 *Matrix1*。此运算不会更改 *Matrix1* 或 *Matrix2*。

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

colDim()

目录 >

colDim(*Matrix***)** ⇒ *表达式*

返回 *Matrix* 所包含的列数。

注意：另请参阅 **rowDim()**。

$\text{colDim}\left(\begin{bmatrix}0 & 1 & 2\\3 & 4 & 5\end{bmatrix}\right)$	3

colNorm()

目录 >

colNorm(*Matrix***)** ⇒ *表达式*

返回 *Matrix* 中列元素绝对值之和的最大值。

注意：不允许使用未定义的矩阵元素。另请参阅 **rowNorm()**。

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

completeSquare()

目录 >

completeSquare(*表达式或方程*, *变量***)** ⟹ *表达式或方程*

completeSquare(*表达式或方程*, *变量* ^*Power***)** ⟹ *表达式或方程*

completeSquare(*表达式或方程*, *变量1*, *变量2 [,...]***)**⟹ *表达式或方程*

completeSquare(*表达式或方程*, { *变量1*, *变量2 [,...]*}**)**⟹ *表达式或方程*

将二次多项式表达式从 a·x²+b·x+c 形式转换为 a·(x-h)²+k 形式

- 或 -

将二次方程从 a·x²+b·x+c=d 形式转换为 a·(x-h)²=k 形式

相对于第二个自变量　第一个自变量必须是标准形式的二次表达式或方程。

第二个自变量必须是单变量项或单变量项求有理幂级数　例如　x　y² 或 z^(1/3)。

第三个和第四个句法尝试完成与变量 *变量1*, *变量2* [,...]) 有关的平方。

completeSquare($x^2+2\cdot x+3,x$) $(x+1)^2+2$

completeSquare($x^2+2\cdot x=3,x$) $(x+1)^2=4$

completeSquare($x^6+2\cdot x^3+3,x^3$) $(x^3+1)^2+2$

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

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

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

conj()

目录 >

conj(*Value1***)** ⟹ *值*

conj(*List1***)** ⟹ *数组*

conj(*Matrix1***)** ⟹ *矩阵*

返回自变量的共轭复数。

conj($1+2\cdot i$) $1-2\cdot i$

conj($\begin{bmatrix}2 & 1-3\cdot i\\ -i & -7\end{bmatrix}$) $\begin{bmatrix}2 & 1+3\cdot i\\ i & -7\end{bmatrix}$

constructMat()

目录 >

constructMat(*Expr***,***Var1***,***Var2***,***numRows***,***numCols***)**
⟹ *矩阵*

返回基于自变量的矩阵。

Expr 是用变量 *Var1* 和 *Var2* 表示的表达式。结果矩阵中的元素通过计算每个 *Var1* 和 *Var2* 增量值的 *Expr* 得出。

Var1 自动从 **1** 递增到 *numRows*。在每一行内　*Var2* 从 **1** 递增到 *numCols*。

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

CopyVar

目录 >

CopyVar *Var1*, *Var2*

CopyVar *Var1*., *Var2*。

CopyVar *Var1*, *Var2* 将变量 *Var1* 的值复制到变量 *Var2* 若 *Var2* 不存在　CopyVar 将创建此变量。变量 *Var1* 必须有一个值。

如果 *Var1* 是现有用户定义之函数的名称　可将该函数的定义复制到函数 *Var2*。必须定义函数 *Var1*。

Var1 必须满足变量命名要求　或者必须是满足该要求的变量名称的化简间接表达式。

Define $a(x)=\frac{1}{x}$ *Done*

Define $b(x)=x^2$ *Done*

CopyVar $a,c: c(4)$ $\frac{1}{4}$

CopyVar $b,c: c(4)$ 16

CopyVar

目录 >

CopyVar *Var1.*, *Var2.* 将 *Var1.* 变量组的所有成员都复制到 *Var2.* 组　若 *Var2.* 不存在　CopyVar 将创建此变量。

Var1. 必须为现有变量组（如统计 *stat.nn* 结果或使用 **LibShortcut()** 函数创建的变量）的名称。如果 *Var2.* 已经存在　此命令将替换两组共有的所有成员并添加不存在的成员。如果 *Var2.* 的一个或多个成员锁定　则 *Var2.* 的所有成员将保持不变更。

aa.a:=45	45
aa.b:=6.78	6.78
aa.c:=8.9	8.9
getVarInfo()	$\begin{bmatrix} aa.a & \text{"NUM"} & \text{"}\square\text{"} \\ aa.b & \text{"NUM"} & \text{"}\square\text{"} \\ aa.c & \text{"NUM"} & \text{"}\square\text{"} \end{bmatrix}$
CopyVar *aa.*,*bb.*	*Done*
getVarInfo()	$\begin{bmatrix} aa.a & \text{"NUM"} & \text{"}\square\text{"} \\ aa.b & \text{"NUM"} & \text{"}\square\text{"} \\ aa.c & \text{"NUM"} & \text{"}\square\text{"} \\ bb.a & \text{"NUM"} & \text{"}\square\text{"} \\ bb.b & \text{"NUM"} & \text{"}\square\text{"} \\ bb.c & \text{"NUM"} & \text{"}\square\text{"} \end{bmatrix}$

corrMat()

目录 >

corrMat(*List1*,*List2*[,...[,*List20*]])

计算增加矩阵 [*List1, List2, ..., List20*] 的关联矩阵。

cos()

trig 键

cos(*Value1***)** ⇒ *值*

cos(*List1***)** ⇒ *数组*

cos(*Value1***)** 以值的形式返回自变量的余弦值。

cos(*List1***)** 返回一个数组　其元素为 *List1* 中所有元素的余弦值。

注意：自变量可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。您可以使用 ° G 或 r 临时更改角度模式。

在 Degree 角度模式下：

$\cos\left(\left\{\left(\frac{\pi}{4}\right)^r\right\}\right)$	0.707107
cos(45)	0.707107
cos({0,60,90})	{1.,0.5,0.}

在 Gradian 角度模式下：

cos({0,50,100})	{1.,0.707107,0.}

在 Radian 角度模式下：

$\cos\left(\left\{\frac{\pi}{4}\right\}\right)$	0.707107
cos(45°)	0.707107

cos()

trig 键

cos(*squareMatrix1*) ⟹ *方阵*

返回 *squareMatrix1* 的矩阵余弦。此运算不同于计算每个元素的余弦值。

当使用标量函数 f(A) 对 *squareMatrix1* (A) 进行运算时，结果按代数方法计算：

计算特征值 (λ_i) 和 A 的特征向量 (V_i)。

squareMatrix1 必须可对角化。同时不得包含未赋值的符号变量。

构建矩阵：

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

然后令 $A = X\ B\ X^{-1}$ 且 $f(A) = X\ f(B)\ X^{-1}$。例如，cos(A) = X cos(B) X^{-1}，其中：

cos(B) =

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

所有运算均使用浮点计算进行。

在 Radian 角度模式下：

$$\cos\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$

$$\begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$$

$\cos^{-1}()$

trig 键

$\cos^{-1}$(*Value1*) ⟹ *值*

$\cos^{-1}$(*List1*) ⟹ *数组*

$\cos^{-1}$(*Value1*) 返回一个角度值，其余弦值为 *Value1*。

$\cos^{-1}$(*List1*) 返回一个数组，其元素为 *List1* 中所对应元素的反余弦值。

注意：返回的结果可以是度、弧度或百分度形式。具体取决于当前的角度模式设置。

注意：您可以通过在计算机键盘上键入 **arccos(...)** 插入此函数。

$\cos^{-1}$(*squareMatrix1*) ⟹ *方阵*

返回 *squareMatrix1* 的矩阵反余弦。此运算不同于计算每个元素的反余弦值。有关计算方法的信息，请参阅 **cos()**。

squareMatrix1 必须可对角化。结果始终包含浮点数。

在 Degree 角度模式下：

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

在 Gradian 角度模式下：

$\cos^{-1}(0)$ 100

在 Radian 角度模式下：

$\cos^{-1}(\{0,0.2,0.5\})$

$\{1.5708, 1.36944, 1.0472\}$

在 Radian 角度模式和 Rectangular 复数格式下：

$$\cos^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1.73485+0.064606\cdot i & -1.49086+2.10514 \\ -0.725533+1.51594\cdot i & 0.623491+0.77836\cdot \\ -2.08316+2.63205\cdot i & 1.79018-1.27182\cdot \end{bmatrix}$$

要查看整个结果，请按 ▲，然后使用 ◀ 和 ▶ 移动光标。

cosh()

目录 >

cosh(*Value1***)** ⇒ *值*

cosh(*List1***)** ⇒ *数组*

cosh(*Value1***)** 返回自变量的双曲余弦值。

cosh(*List1***)** 返回一个数组　其元素为 ***List1*** 中所对应元素的双曲余弦值。

cosh(*squareMatrix1***)** ⇒ *方阵*

返回 *squareMatrix1* 的矩阵双曲余弦　此运算不同于计算每个元素的双曲余弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

$\cosh\left(\left(\frac{\pi}{4}\right)^r\right)$ 1.74671E19

在 Radian 角度模式下：

$$\cosh\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix}$$

$$\begin{bmatrix}421.255&253.909&216.905\\327.635&255.301&202.958\\226.297&216.623&167.628\end{bmatrix}$$

$\cosh^{-1}()$

目录 >

cosh^{-1}(*Value1***)** ⇒ *值*

cosh^{-1}(*List1***)** ⇒ *数组*

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

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

cosh^{-1}(*Value1***)** 返回自变量的反双曲余弦值。

cosh^{-1}(*List1***)** 返回一个数组　其元素为 ***List1*** 中所对应元素的反双曲余弦值。

注意：您可以通过在计算机键盘上键入 **arccosh(...)** 插入此函数。

cosh^{-1}(*squareMatrix1***)** ⇒ *方阵*

返回 *squareMatrix1* 的矩阵反双曲余弦　此运算不同于计算每个元素的反双曲余弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Radian 角度模式下和 Rectangular 复数格式下：

$$\cosh^{-1}\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix}$$

$$\begin{bmatrix}2.52503+1.73485\cdot i & -0.009241-1.4908\\0.486969-0.725533\cdot i & 1.66262+0.623491\\-0.322354-2.08316\cdot i & 1.26707+1.79018\end{bmatrix}$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

cot()

trig 键

cot(*Value1***)** ⇒ *值*

cot(*List1***)** ⇒ *数组*

返回 *Value1* 的余切值　或返回一个数组　其元素为 *List1* 中所对应元素的余切值。

注意：自变量可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。您可以使用 ° G 或 r 临时更改角度模式。

在 Degree 角度模式下：

$\cot(45)$ 1

在 Gradian 角度模式下：

$\cot(50)$ 1

在 Radian 角度模式下：

$\cot(\{1,2.1,3\})$

$\{0.642093,-0.584848,-7.01525\}$

cot^{-1}()

trig 键

cot^{-1}(*Value1*) ⇒ *值*

cot^{-1}(*List1*) ⇒ *数组*

返回余切值为 *Value1* 的角度　或返回一个数组　其元素为 *List1* 所对应元素的反余切值。

注意：返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

注意：您可以通过在计算机键盘上键入 **arccot(...)** 插入此函数。

在 Degree 角度模式下：

$\cot^{-1}(1)$ 45

在 Gradian 角度模式下：

$\cot^{-1}(1)$ 50

在 Radian 角度模式下：

$\cot^{-1}(1)$ .785398

coth()

目录 >

coth(*Value1*) ⇒ *值*

coth(*List1*) ⇒ *数组*

返回 *Value1* 的双曲余切　或返回一个数组　其元素为 *List1* 中所对应元素的双曲余切值。

$\coth(1.2)$ 1.19954

$\coth(\{1,3.2\})$ $\{1.31304,1.00333\}$

coth^{-1}()

目录 >

coth^{-1}(*Value1*) ⇒ *值*

coth^{-1}(*List1*) ⇒ *数组*

返回 *Value1* 的反双曲余切或返回一个数组　其元素为 *List1* 所对应元素的反双曲余切值。

注意：您可以通过在计算机键盘上键入 **arccoth(...)** 插入此函数。

$\coth^{-1}(3.5)$ 0.293893

$\coth^{-1}(\{-2,2.1,6\})$ $\{-0.549306,0.518046,0.168236\}$

count()

目录 >

count(*Value1orList1* [,*Value2orList2* [,...]]) ⇒ *值*

返回自变量中所有元素的累积个数　结果为一个数值。

自变量可以是表达式　值　数组或矩阵。您可以混合数据类型并使用各种维数的自变量。

对于数组　矩阵或单元格范围　应评估每个元素以　确定其是否应包括在计数中。

在 Lists & Spreadsheet 应用程序中　您可以使用单元格范围代替任何自变量。

空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

count(2,4,6) 3

count({2,4,6}) 3

$\text{count}\left(2,\{4,6\},\begin{bmatrix}8 & 10\\ 12 & 14\end{bmatrix}\right)$ 7

countif()

目录 >

countif(*List,Criteria***)** ⇒ *值*

返回 *List* 中符合指定 *Criteria* 的所有元素的累积个数。

Criteria 可以是：

- 值　表达式或字符串。例如　**3** 仅计数 *List* 中值等于 3 的元素。
- 布尔表达式　使用符号 **?** 作为各元素的占位符。例如 **?<5** 仅计数 *List* 中小于 5 的元素。

在 Lists & Spreadsheet 应用程序中　您可以使用单元格范围代替 *List*。

数组中的空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

注意：另请参阅第 101 页的 **sumIf()** 和第 39 页的 **frequency()**。

countIf({1,3,"abc",undef,3,1},3)　2

计数等于 3 的元素。

countIf({"abc","def","abc",3},"def")　1

计数等于 "def." 的元素。

countIf({1,3,5,7,9},?<5)　2

计数 1 和 3。

countIf({1,3,5,7,9},2<?<8)　3

计数 3　5 和 7。

countIf({1,3,5,7,9},?<4 or ?>6)　4

计数 1　3　7 和 9。

cPolyRoots()

目录 >

cPolyRoots(*Poly,Var***)** ⇒ *数组*

cPolyRoots(*ListOfCoeffs***)** ⇒ *数组*

第一种句法 **cPolyRoots(***Poly***, *Var***)** 返回一个数组　其元素为关于变量 *Var* 的多项式 *Poly* 的复数根。

Poly 必须为扩展形式的单变量多项式。请勿使用非扩展形式　如 $y^2 \cdot y+1$ 或 $x \cdot x+2 \cdot x+1$

第二种句法 **cPolyRoots(***ListOfCoeffs***)** 返回一个数组　其元素为 *ListOfCoeffs 中系数的复数根*。

注意：另请参阅 **polyRoots()**（ 第 74 页 ）。

polyRoots(y^3+1,y)　{-1}

cPolyRoots(y^3+1,y)
{-1,0.5−0.866025·*i*,0.5+0.866025·*i*}

polyRoots($x^2+2 \cdot x+1,x$)　{-1,-1}

cPolyRoots({1,2,1})　{-1,-1}

crossP()

目录 >

crossP(*List1***, *List2***)** ⇒ *数组*

以数组形式返回 *List1* 和 *List2* 的交叉乘积。

List1 和 *List2* 必须有相同的维数　必须为 2 维或 3 维。

crossP(*Vector1***, *Vector2***)** ⇒ *向量*

返回一个行向量或列向量（ 根据自变量的不同 ） 其值为 *Vector1* 和 *Vector2* 的交叉乘积。

Vector1 和 *Vector2* 必须都为行向量　或必须都为列向量。两个向量必须有相同的维数　且维数必须为 2 或 3。

crossP({0.1,2.2,-5},{1,-0.5,0})
{-2.5,-5.,-2.25}

crossP([1　2　3],[4　5　6])　[-3　6　-3]

crossP([1　2],[3　4])　[0　0　-2]

csc()

trig 键

csc(*Value1*) ⇒ *值*

csc(*List1*) ⇒ *数组*

返回 *Value1* 的余割值　或返回一个数组　其元素为 *List1* 中所对应元素的余割值。

在 Degree 角度模式下：

$\csc(45)$ 1.41421

在 Gradian 角度模式下：

$\csc(50)$ 1.41421

在 Radian 角度模式下：

$\csc\left(\left\{1,\frac{\pi}{2},\frac{\pi}{3}\right\}\right)$ $\{1.1884,1.,1.1547\}$

$csc^{-1}()$

trig 键

csc^{-1}(*Value1*) ⇒ *值*

csc^{-1}(*List1*) ⇒ *数组*

返回余割值为 *Value1* 的角度　或返回一个数组　其元素为 *List1* 所对应元素的反余割值。

注意：返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

注意：您可以通过在计算机键盘上键入 **arccsc(...)** 插入此函数。

在 Degree 角度模式下：

$\csc^{-1}(1)$ 90

在 Gradian 角度模式下：

$\csc^{-1}(1)$ 100

在 Radian 角度模式下：

$\csc^{-1}(\{1,4,6\})$ $\{1.5708,0.25268,0.167448\}$

csch()

目录 >

csch(*Value1*) ⇒ *值*

csch(*List1*) ⇒ *数组*

返回 *Value1* 的双曲余割　或返回一个数组　其元素为 *List1* 中所对应元素的双曲余割值。

$\operatorname{csch}(3)$ 0.099822

$\operatorname{csch}(\{1,2.1,4\})$ $\{0.850918,0.248641,0.036644\}$

$csch^{-1}()$

目录 >

$csch^{-1}$(*Value*) ⇒ *值*

$csch^{-1}$(*List1*) ⇒ *数组*

返回 *Value1* 的反双曲余割或返回一个数组　其元素为 *List1* 所对应元素的反双曲余割值。

注意：您可以通过在计算机键盘上键入 **arccsch(...)** 插入此函数。

$\operatorname{csch}^{-1}(1)$ 0.881374

$\operatorname{csch}^{-1}(\{1,2.1,3\})$ $\{0.881374,0.459815,0.32745\}$

CubicReg

目录 >

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

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算三次多项式回归 $y = a \cdot x^3+b \cdot x^2+c \cdot x+d$ 。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：$a \cdot x^3+b \cdot x^2+c \cdot x+d$
stat.a stat.b stat.c stat.d	回归系数
stat.R^2	确定系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

cumulativeSum()

目录 >

cumulativeSum(*List1***)** ⇒ *数组*

返回一个数组 其组成为 *List1* 从元素 1 开始的元素的累积和。

cumulativeSum(*Matrix1***)** ⇒ *矩阵*

返回一个矩阵 其组成为 *Matrix1* 中元素的累积和。其各元素为 Matrix1 中每列从上到下的累积和。

List1 或 *Matrix1* 中的空（ 空值 ）元素会在结果数组或矩阵中生成一个空值元素。有关空元素的更多信息 请参阅第 134 页。

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

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

$$\text{cumulativeSum}(m1) \qquad \begin{bmatrix}1 & 2\\ 4 & 6\\ 9 & 12\end{bmatrix}$$

Cycle

立即将控制转入当前循环（ **For** 　**While** 或 **Loop** ）的下一轮迭代。

Cycle 只能在三种循环结构（ **For** 　**While** 或 **Loop** ）内使用。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

函数数组为对从 1 到 100 的整数求和　跳过 50。

```
Define g()=Func                    Done
            Local temp,i
            0→temp
            For i,1,100,1
            If i=50
            Cycle
            temp+i→temp
            EndFor
            Return temp
            EndFunc
g()                                5000
```


Vector **▶Cylind**

注意：您可以通过在计算机键盘上键入 **@>Cylind** 插入此运算符。

以圆柱坐标形式 [r,∠θ, z] 显示行向量或列向量。

Vector 必须恰好包含三个元素　可以是行向量　也可以是列向量。

```
[2  2  3]▶Cylind
                [2.82843  ∠0.785398  3.]
```

D

dbd(*date1,date2***)** ⟹ *值*

使用实际天数计数法返回 *date1* 和 *date2* 间的间隔天数。

date1 和 *date2* 可为标准日历上日期范围内的数值或数值数组。如果 *date1* 和 *date2* 均为数组　则两个数组的长度必须相同。

date1 和 *date2* 必须介于 1950 到 2049 年之间。

您可按两种格式中的任何一种输入日期。两种日期格式的小数点位置不同。

MM.DDYY（ 美国常用格式 ）
DDMM.YY（ 欧洲常用格式 ）

```
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

目录 >

Expr1 **▶DD** ⇒ *值*

List1 **▶DD** ⇒ *数组*

Matrix1 **▶DD** ⇒ *矩阵*

注意：您可以通过在计算机键盘上键入 **@>DD** 插入此运算符。

返回角度形式的自变量的十进制等效值。自变量可以是角度模式设置为度　弧度或百分度的数字　数组或矩阵。

在 Degree 角度模式下：

$(1.5°)$▶DD　1.5°

$(45°22'14.3")$▶DD　45.3706°

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

在 Gradian 角度模式下：

1▶DD　$\frac{9}{10}$g

在 Radian 角度模式下：

(1.5)▶DD　85.9437°

▶Decimal

目录 >

Number1 **▶Decimal** ⇒ *值*

List1 **▶Decimal** ⇒ *值*

Matrix1 **▶Decimal** ⇒ *值*

注意：您可以通过在计算机键盘上键入 **@>Decimal** 插入此运算符。

显示自变量的十进制形式。此运算符只能在输入行的末尾处使用。

$\frac{1}{3}$▶Decimal　0.333333

Define

目录 >

Define *Var* = *Expression*

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

定义变量 *Var* 或用户定义的函数 *Function*。

参数（ 如 *Param1* ）提供占位符用于将自变量传递到函数。调用用户定义的函数时　您必须提供对应于参数的自变量（ 如值或变量 ）。调用时　函数会使用提供的自变量计算 *Expression*。

Var 和 *Function* 不得是系统变量或者内置函数或命令的名称。

注意：此形式的 **Define** 指令等同于执行以下表达式：*表达式* ➞ *Function*(*Param1*,*Param2*)。

Define $g(x,y)=2\cdot x-3\cdot y$　*Done*

$g(1,2)$　-4

$1\to a$: $2\to b$: $g(a,b)$　-4

Define $h(x)=$when$(x<2,2\cdot x-3,-2\cdot x+3)$　*Done*

$h(-3)$　-9

$h(4)$　-5

Define

目录 >

Define *Function*(*Param1*, *Param2*, ...)**= Func**
Block
EndFunc

Define *Program*(*Param1*, *Param2*, ...)**= Prgm**
Block
EndPrgm

此格式中，用户定义的函数或程序可执行多条语句组成的块。

Block 可以是一条语句，也可以是单独行上的一系列语句。*Block* 还可以包含表达式和指令（如 **If**、**Then**、**Else** 和 **For**）。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（而不是 enter）输入多行定义。在计算机键盘上，按住 **Alt** 然后按 **Enter**。

注意：另请参阅第 28 页的 **Define LibPriv** 和第 29 页的 **Define LibPub**。

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

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

Define LibPriv

目录 >

Define LibPriv *Var* **=** *Expression*

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

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

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

除定义的是专用库变量、函数或程序外，操作与 **Define** 操作相同。专用函数和程序不在 Catalog 中显示。

注意：另请参阅第 27 页的 **Define** 和第 29 页的 **Define L bPub**。

Define LibPub

目录 >

Define LibPub *Var* **=** *Expression*

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

Define LibPub *Function*(*Param1*, *Param2*, ...)= **Func**
Block
EndFunc

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

除定义的是公用库变量 函数或程序外 操作与 **Define** 操作相同。保存并刷新库后 公用函数和程序将在 Catalog 中显示。

注意：另请参阅第 27 页的 **Define** 和第 28 页的 **Define LibPriv**。

deltaList()

请参阅 ΔList()，（ 第 54 页 ）。

DelVar

目录 >

DelVar *Var1*[, *Var2*] [, *Var3*] ...

DelVar *Var*.

从内存删除指定变量或变量组。

如果有一个或多个变量锁定 此命令将显示错误消息并仅删除未锁定变量。请参阅 **unLock**（ 第 111 页 ）。

$2 \rightarrow a$	2
$(a+2)^2$	16
DelVar *a*	*Done*
$(a+2)^2$	"Error: Variable is not defined"

DelVar *Var*. 删除 *Var*. 变量组（ 如统计 *stat.nn* 结果或使用 **LibShortcut()** 函数*创建的变量 ）中的所有成员。*

DelVar 命令中这种格式的点 (**.**) 限制其仅用于删除变量组 而单个变量 *Var* 不受影响。

aa.a:=45	45
aa.b:=5.67	5.67
aa.c:=78.9	78.9
getVarInfo()	$\begin{bmatrix} aa.a & "NUM" & "\square" \\ aa.b & "NUM" & "\square" \\ aa.c & "NUM" & "\square" \end{bmatrix}$
DelVar *aa*.	*Done*
getVarInfo()	"NONE"

delVoid()

目录 >

delVoid(*List1***)** ⇒ *数组*

返回一个数组 其元素为 *List1* 删除所有空（ 空值 ）元素后的内容。

有关空元素的更多信息 请参阅第 134 页。

delVoid({1,void,3})	{1,3}

det()

目录 >

det(*squareMatrix*[, *Tolerance*]) ⇒ *表达式*

返回 *squareMatrix* 的行列式。

或者 如果矩阵中任何元素的绝对值小于 *Tolerance* 则将该元素视为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时 使用此公差。否则 *Tolerance* 将被忽略。

- 如果您使用 ctrl enter 或将 **Auto or Approximate** 设定为 Approximate 模式 则运算会使用浮点算法完成。
- 如果 *Tolerance* 被省略或未使用 则默认的公差算法为：

  5E−14 ·**max(dim(***squareMatrix***))**·**rowNorm(***squareMatrix***)**

$\det\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ → -2

$\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix} \rightarrow mat1$ → $\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix}$

$\det(mat1)$ → 0

$\det(mat1,.1)$ → $1.\text{E}20$

diag()

目录 >

diag(*List***)** ⇒ *矩阵*

diag(*rowMatrix***)** ⇒ *矩阵*

diag(*columnMatrix***)** ⇒ *矩阵*

返回一个矩阵 其主对角线上为自变量数组或矩阵中的值。

$\text{diag}(\begin{bmatrix}2 & 4 & 6\end{bmatrix})$ → $\begin{bmatrix}2 & 0 & 0\\ 0 & 4 & 0\\ 0 & 0 & 6\end{bmatrix}$

diag(*squareMatrix***)** ⇒ *行矩阵*

返回一个行矩阵 包含 *squareMatrix* 主对角线上的元素。

squareMatrix 必须为矩形。

$\begin{bmatrix}4 & 6 & 8\\ 1 & 2 & 3\\ 5 & 7 & 9\end{bmatrix}$ → $\begin{bmatrix}4 & 6 & 8\\ 1 & 2 & 3\\ 5 & 7 & 9\end{bmatrix}$

$\text{diag}(Ans)$ → $\begin{bmatrix}4 & 2 & 9\end{bmatrix}$

dim()

目录 >

dim(*List***)** ⇒ *整数*

返回 *List* 的维数。

$\dim(\{0,1,2\})$ → 3

dim(*Matrix***)** ⇒ *数组*

以二维数组 { 行 列 } 的形式返回矩阵的维数。

$\dim\left(\begin{bmatrix}1 & -1\\ 2 & -2\\ 3 & 5\end{bmatrix}\right)$ → $\{3,2\}$

dim(*String***)** ⇒ *整数*

返回字符串 *String* 中包含的字符数量。

$\dim(\text{"Hello"})$ → 5

$\dim(\text{"Hello "\&"there"})$ → 11

Disp

目录 >

Disp [*exprOrString1*] [**,** *exprOrString2*] ...

显示 Calculator 历史记录中的自变量。这些自变量将连续显示　以窄空格作为分隔符。

此功能主要应用于程序和函数中　以确保显示计算的中间过程。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

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

▶DMS

目录 >

Value **▶DMS**

List **▶DMS**

Matrix **▶DMS**

注意：您可以通过在计算机键盘上键入 **@>DMS** 插入此运算符。

以角度形式表示自变量并显示等效的 DMS (DDDDDD°MM' SS.ss'') 值。请参阅 °, ', '' （ 第 130 页 ） 了解 DMS（ 度　分　秒 ）的格式。

注意：在弧度模式下使用时　▶DMS 会从弧度转换为度。如果输入值后跟度符号 °　则不会进行转换。您只能在输入行结尾处使用 **▶DMS**。

在 Degree 角度模式下：

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

dotP()

目录 >

dotP(*List1***,** *List2***)** ⇒ *表达式*

返回两个数组的 " 点 " 乘积。

dotP(*Vector1***,** *Vector2***)** ⇒ *表达式*

返回两个向量的 " 点 " 乘积。

两个向量必须同时为行向量　或同时为列向量。

```
dotP({1,2},{5,6})                17
dotP([1  2  3],[4  5  6])        32
```

E

e^()

e^ 键

e^(*Value1*) ⇒ *值*

返回以 ***e*** 为底　以 *Value1* 为乘方的指数值。

注意：另请参阅 **e 指数模板**（ 第 1 页 ）。

注意：按 e^ 可显示 e^（ 不同于在键盘上按字母 E ）。

e^1 → 2.71828

e^{3^2} → 8103.08

您可以输入形式为 $re^{i\theta}$ 的极坐标复数。不过　只能在 Radian 角度模式下使用此形式　在 Degree 或 Gradian 角度模式下会导致 Domain error。

e^(*List1*) ⇒ *数组*

返回以 **e** 为底　以 *List1* 各元素为乘方的指数值。

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

e^(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵指数。该运算不同于计算以 **e** 为底　以矩阵各元素为乘方的指数值。有关计算方法的信息请参阅 **cos()**。

$e^{\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix}}$ → $\begin{bmatrix}782.209&559.617&456.509\\680.546&488.795&396.521\\524.929&371.222&307.879\end{bmatrix}$

squareMatrix1 必须可对角化　结果始终包含浮点数。

eff()

目录 >

eff(*nominalRate,CpY*) ⇒ *值*

将名义利率 *nominalRate* 转换为年度有效利率的财务函数指定 *CpY* 作为每年复利期数的数量。

eff(5.75,12) → 5.90398

nominalRate 必须为实数　*CpY* 必须为 > 0 的实数。

注意：另请参阅 **nom()**（ 第 67 页 ）。

eigVc()

目录 >

eigVc(*squareMatrix*) ⇒ *矩阵*

返回一个矩阵　其中包含实数或复数 *squareMatrix* 的特征向量　结果中每列对应于一个特征值。请注意　特征变量并不唯一　改变常数因子可得到不同的特征向量。特征向量应规范化　即如果 V = [x_1, x_2, … , x_n]　那么：

$x_1^2 + x_2^2 + \ldots + x_n^2 = 1$

squareMatrix 首先通过近似变换进行平衡　直到行范数和列范数最大程度地接近。然后将 *squareMatrix* 化简为上 Hessenberg 形式　并通过 Schur 因式分解计算特征向量。

在 Rectangular 复数格式下：

$\begin{bmatrix}-1&2&5\\3&-6&9\\2&-5&7\end{bmatrix} \rightarrow m1$ → $\begin{bmatrix}-1&2&5\\3&-6&9\\2&-5&7\end{bmatrix}$

eigVc(*m1*)

$\begin{bmatrix}-0.800906 & 0.767947 & (\\ 0.484029 & 0.573804+0.052258\cdot i & 0.5738\\ 0.352512 & 0.262687+0.096286\cdot i & 0.2626\end{bmatrix}$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

eigVl()

目录 >

eigVl(*squareMatrix***)** ⇒ *数组*

返回由实数或复数 *squareMatrix* 特征值组成的数组。

squareMatrix 首先通过近似变换进行平衡　直到行范数和列范数最大程度地接近。然后将 *squareMatrix* 化简为上 Hessenberg 形式　并通过上 Hessenberg 矩阵计算特征值。

在 Rectangular 复数格式模式下：

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

$$\text{eigVl}(m1)$$

$$\{-4.40941, 2.20471+0.763006\cdot i, 2.20471-0.$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

Else

请参阅 If（ 第 45 页 ）。

ElseIf

目录 >

If *BooleanExpr1* **Then**
　　Block1
ElseIf *BooleanExpr2* **Then**
　　Block2
　　⋮
ElseIf *BooleanExprN* **Then**
　　BlockN
EndIf
　　⋮

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上，按住 Alt 然后按 Enter。

```
Define g(x)=Func
              If x≤-5 Then
              Return 5
              ElseIf x>-5 and x<0 Then
              Return -x
              ElseIf x≥0 and x≠10 Then
              Return x
              ElseIf x=10 Then
              Return 3
              EndIf
              EndFunc

                                    Done
```

EndFor

请参阅 For（ 第 38 页 ）。

EndFunc

请参阅 Func（ 第 40 页 ）。

EndIf

请参阅 If（ 第 45 页 ）。

EndLoop

请参阅 Loop（ 第 59 页 ）。

EndPrgm

请参阅 Prgm（ 第 76 页 ）。

EndTry

请参阅 Try（ 第 106 页 ）。

euler()

目录 >

euler(*表达式*, *变量*, *因变量*, { *变量 0*, *变量最大值*}, *因变量 0*, *变量步长*

[, *欧拉步长*]) ⟹ *矩阵*

euler(*表达式方程组*, *变量*, *因变量数组*, { *变量 0*, *变量最大值*}, *因变量数组 0*, *变量步长* [, *欧拉步长*]) ⟹ *矩阵*

euler(*表达式数组*, *变量*, *因变量数组*, { *变量 0*, *变量最大值*}, *因变量数组 0*, *变量步长* [, *欧拉步长*]) ⟹ *矩阵*

使用欧拉方法求解方程组

$\frac{d\,depVar}{d\,Var}$ = *Expr*(*变量*, *因变量*)

其中 *depVar*(*变量 0*)= *因变量 0* 位于区间 [*变量 0*, *变量最大值*] 中。返回一个矩阵　其第一行定义 *变量*输出值　而第二行定义相应的 *变量*值处第一个求解分量的值　依此类推。

*表达式*是定义常微分方程 (ODE) 的右侧内容。

*表达式方程组*是定义 ODE 方程组的右侧方程组（ 对应*因变量数组*中因变量的阶数 ）。

*表达式数组*是定义 ODE 方程组的右侧数组（ 对应*因变量数组*中因变量的阶数 ）。

*变量*是自变量。

*因变量数组*是因变量的数组。

{ *变量 0*, *变量最大值* } 是两个元素的数组　告知函数从 *变量 0* 到 *变量最大值*为一个整体。

因变量数组 0 是因变量初始值的数组。

*变量步长*是一个非零数字　满足 **sign(** *变量步长* **) = sign(** *变量最大值* **-** *变量 0***)**　而解在 *变量 0*+*i*· *变量步长*处返回（ 对于所有满足 *变量 0*+*i*· *变量步长*位于 [*变量 0*, *变量最大值*] 区间条件的 i=0,1,2,…　*变量最大值*处可能没有解值 ）。

*欧拉步长*是一个正整数（ 默认设为 1 ） 它定义输出值之间的欧拉步长数。欧拉方法使用的实际步长大小为 *变量步长*/*欧拉步长*。

微分方程：
y'=0.001*y*(100-y) 和 y(0)=10

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

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

要查看完整结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

方程组：

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

其中 *y1*(0)=2 并且 *y2*(0)=5

$$\text{euler}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases},t,\{y1,y2\},\{0,5\},\{2,5\},1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix}$$

Exit

目录 >

Exit

退出当前的 **For**　**While** 或 **Loop** 块。

Exit 只能在三种循环结构（ **For**　**While** 或 **Loop** ）内使用。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

函数清单：

```
Define g()=Func                    Done
    Local temp,i
    0→temp
    For i,1,100,1
    temp+i→temp
    If temp>20 Then
    Exit
    EndIf
    EndFor
    EndFunc
g()                                21
```

exp()

e^x 键

exp(*Value1***)** ⟹ *值*

返回以 **e** 为底 以 *Value1* 为乘方的指数值。

注意：另请参阅 **e** 指数模板（ 第 1 页 ）。

您可以输入形式为 $re^{i\theta}$ 的极坐标复数。不过 只能在 Radian 角度模式下使用此形式 在 Degree 或 Gradian 角度模式下会导致 Domain error。

exp(*List1***)** ⟹ *数组*

返回以 **e** 为底 以 *List1* 各元素为乘方的指数值。

exp(*squareMatrix1***)** ⟹ *方阵*

返回 *squareMatrix1* 的矩阵指数。该运算不同于计算以 **e** 为底 以矩阵各元素为乘方的指数值。有关计算方法的信息请参阅 **cos()**。

squareMatrix1 必须可对角化 结果始终包含浮点数。

e^{1} → 2.71828

e^{3^2} → 8103.08

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

$e^{\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}}$ → $\begin{bmatrix}782.209 & 559.617 & 456.509\\ 680.546 & 488.795 & 396.521\\ 524.929 & 371.222 & 307.879\end{bmatrix}$

expr()

目录 >

expr(*String***)** ⟹ *表达式*

以表达式形式返回 *String* 中包含的字符串并立即执行该表达式。

"Define cube(x)=x^3" → *funcstr*

"Define cube(x)=x^3"

expr(*funcstr*) *Done*

cube(2) 8

ExpReg

目录 >

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

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算指数回归 $y = a\cdot(b)^x$。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：$a\cdot(b)^x$
stat.a stat.b	回归系数
stat.r^2	变换数据的线性确定系数

输出变量	说明
stat.r	变换数据的相关系数 (x, ln(y))
stat.Resid	与指数模型相关的残差
stat.ResidTrans	与变换数据的线性拟合相关的残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

F

factor()

目录 >

factor(*rationalNumber***)** 返回有理数的素数分解。对于合数　运算时间将随着第二大因式的位数呈指数增长。例如分解一个 30 位的整数可能需要一天多的时间　而分解一个 100 位的数可能需要超过一个世纪的时间。

手动停止计算：

- Windows®：按住 **F12** 键并反复按 **Enter** 键。
- Macintosh®: 按住 **F5** 键并反复按 **Enter** 键。
- **手持设备：**按住 on 键并反复按 enter 。

如果您只是想确定一个数是否为质数　请使用 **isPrime()**。这样运算速度更快　特别是当 *rationalNumber* 不是质数且第二大因式超过五位时更为高效。

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

F Cdf()

目录 >

F Cdf(*lowBound***,***upBound***,***dfNumer***,***dfDenom***)** ⇒　如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*；如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

FCdf(*lowBound***,***upBound***,***dfNumer***,***dfDenom***)** ⇒　如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*；如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

计算指定 *dfNumer*（ 分子自由度 ）和 *dfDenom*（ 分母自由度 ）的 *下界*和 *上界*之间的 F 分布概率。

对于 P（ *X* ≤ *上界* ） 设定 *下界* = 0。

Fill

目录 >

Fill *Value*, *matrixVar* ⇒ *矩阵*

用 *Value* 替换变量 *matrixVar* 中的各元素。

matrixVar 必须已经存在。

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

Fill

目录 >

Fill *Value, listVar* ⇒ *数组*

用 *Value* 替换变量 *listVar* 中的各元素。

listVar 必须已经存在。

$\{1,2,3,4,5\} \rightarrow alist$ $\{1,2,3,4,5\}$

Fill 1.01,*alist* *Done*

alist $\{1.01,1.01,1.01,1.01,1.01\}$

FiveNumSummary

目录 >

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

提供关于数组 *X* 单变量统计的摘要。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。）

X 表示包含数据的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。

Category 是相应 *X* 数据类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

数组 *X*　*Freq* 或 *Category* 中任意一个数组的空（ 空值 ）元素都会导致所有这些数组中对应元素为空值。有关空元素的更多信息　请参阅第 134 页。

输出变量	说明
stat.MinX	x 值的最小值。
stat.Q_1X	x 的第一个四分位数。
stat.MedianX	x 的中位数。
stat.Q_3X	x 的第三个四分位数。
stat.MaxX	x 值的最大值。

floor()

目录 >

floor(*Value1***)** ⇒ *整数*

返回 ≤ 自变量的最大整数。此函数类似于 **int()**。

自变量可以是实数　也可以是复数。

floor(*List1***)** ⇒ *数组*

floor(*Matrix1***)** ⇒ *矩阵*

返回一个数组或矩阵　其组成为各元素向下取整的函数值。

注意：另请参阅 **ceiling()** 和 **int()**。

$\text{floor}(-2.14)$ $-3.$

$\text{floor}\left(\left\{\frac{3}{2},0,-5.3\right\}\right)$ $\{1,0,-6.\}$

$\text{floor}\begin{bmatrix}1.2 & 3.4\\ 2.5 & 4.8\end{bmatrix}$ $\begin{bmatrix}1. & 3.\\ 2. & 4.\end{bmatrix}$

For

目录 >

For *Var*, *Low*, *High* [, *Step*]
Block
EndFor

对 *Var* 的每个值 从 *Low* 到 *High* 以 *Step* 为增量 反复执行 *Block* 中的语句。

Var 不得为系统变量。

Step 可以是正数或 也可以是负数。默认值为 1。

Block 可以是一条语句 也可以是以 ":" 字符分隔的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中 您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

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

format()

目录 >

format(*Value*[, *formatString*]**)** ⇒ *字符串*

以基于格式模板的字符串的形式返回 *Value*。

formatString 必须是如下形式的字符串："F[n]" "S[n]" "E[n]" "G[n][c]" 其中 [ ] 表示可选的部分。

F[n]：Fixed 格式。 n 为小数点后显示的位数。

S[n]：Scientific 格式。 n 为小数点后显示的位数。

E[n]：Engineering 格式。 n 为第一个有效数字后的位数。指数将调整为三的倍数 并且小数点向右移零位 一位或两位。

G[n][c]：与固定格式相同 但也将小数点左边的数位每三个分为一组。如果 c 为句号 则小数点将显示为逗号。

[Rc]：上述指定符可以加上一个以 Rc 小数点标记的后缀 其中 c 是单个字符 指明替代小数点的符号。

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"

fPart()

目录 >

fPart(*Expr1***)** ⇒ *表达式*
fPart(*List1***)** ⇒ *数组*
fPart(*Matrix1***)** ⇒ *矩阵*

返回自变量的分数部分。

对于数组或矩阵 返回各元素的分数部分。

自变量可以是实数 也可以是复数。

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

FPdf()

目录 >

FPdf(*XVal*,*dfNumer*,*dfDenom***)** ⇒ 如果 *XVal* 是数值，则结果为*数值*，如果 *XVal* 是数组，则结果为*数组*。

计算指定 *dfNumer*（ 自由度 ）和 *dfDenom* 在 *XVal* 的 F 分布概率。

freqTable▶list()

目录 >

freqTable▶list(*List1***,***freqIntegerList***)** ⇒ *数组*

返回一个数组　其组成为 *List1* 的元素根据 *freqIntegerList* 中的频率展开的数值。此函数可用于生成 Data & Statistics 应用程序的频率表。

List1 可以是任何有效的数组。

freqIntegerList 的维数必须与 *List1* 相同　且必须只包含非负的整数元素。每个元素指定相应的 *List1* 元素将在结果数组中重复的次数。值为零时将排除相应的 *List1* 元素。

注意：您可以通过在计算机键盘上键入 **freqTable@>list(...)** 插入此函数。

空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

freqTable▶list({1,2,3,4},{1,4,3,1})
{1,2,2,2,2,3,3,3,4}

freqTable▶list({1,2,3,4},{1,4,0,1})
{1,2,2,2,2,4}

frequency()

目录 >

frequency(*List1***,***binsList***)** ⇒ *数组*

返回一个数组　其组成为 *List1* 中元素的计数。计数以您在 *binsList* 中定义的范围（ 块 ）为基础。

如果 *binsList* 是 {b(1), b(2), …, b(n)}　则指定的范围是 {**?**≤b(1), b(1)<**?**≤b(2),…,b(n-1)<**?**≤b(n), b(n)>**?**}。结果数组中的元素比 *binsList* 多一个。

结果的每个元素对应于 *List1* 在该块范围内的元素的个数。结果将以 **countIf()** 函数形式表达　为 { countIf(list, **?**≤b(1)),　countIf(list, b(1)<**?**≤b(2)), …, countIf(list, b(n-1)<**?**≤b(n)), countIf(list, b(n)>**?**)}。

List1 中不能 " 放在任何块中 " 的元素将被忽略。空（ 空值 ）元素也将被忽略。有关空元素的更多信息　请参阅第 134 页。

在 Lists & Spreadsheet 应用程序中　您可以使用单元格范围代替上述的两个自变量。

注意：另请参阅 **countIf()**（ 第 23 页 ）。

datalist:={1,2,e,3,π,4,5,6,"hello",7}
{1,2,2.71828,3,3.14159,4,5,6,"hello",7}

frequency(*datalist*,{2.5,4.5})　{2,4,3}

结果说明：
Datalist 中有 2 个元素 ≤2.5
Datalist 中有 4 个元素 >2.5 且 ≤4.5
Datalist 中有 3 个元素 >{4.5
元素 "hello" 是一个字符串　不能放在任何定义的块中。

Test_2Samp

目录 >

Test_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]
FTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

（ 数据数组输入 ）

Test_2Samp *sx1*,*n1*,*sx2*,*n2*[,*Hypoth*]
FTest_2Samp *sx1*,*n1*,*sx2*,*n2*[,*Hypoth*]

（ 摘要统计输入 ）

执行双样本 F 检验。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

对于 H_a：σ1 > σ2　设置 *Hypoth*>0
对于 H_a：σ1 ≠ σ2（ 默认值 ）设置 *Hypoth*0
对于 H_a：σ1 < σ2　设置 *Hypoth*<0

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.F	为数据序列计算的 F 统计
stat.PVal	可拒绝零假设的最小显著性水平
stat.dfNumer	分子自由度 = n1-1
stat.dfDenom	分母自由度 = n2-1
stat.sx1 stat.sx2	*List 1* 和 *List 2* 中数据序列的样本标准差
stat.x1_bar stat.x2_bar	*List 1* 和 *List 2* 中数据序列的样本平均值
stat.n1 stat.n2	样本的大小

Func

目录 >

Func
Block
EndFunc

用于创建用户定义函数的模板。

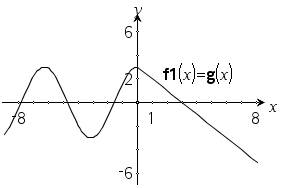
Block 可以是一条语句 也可以是以 **":"** 字符分隔的或者单独行上的一系列语句。函数可以使用 **Return** 指令返回特定的结果。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中 您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

定义分段函数：

```
Define g(x)=Func                    Done
          If x<0 Then
          Return 3·cos(x)
          Else
          Return 3−x
          EndIf
          EndFunc
```

绘制 g(x) 的结果



G

gcd()

目录 >

gcd(*Number1***, *Number2*)** ⇒ *表达式*

返回两个自变量的最大公约数。两个分数的 **gcd** 值是其分子的 **gcd** 值除以其分母的 **lcm** 值。

在 Auto 或 Approximate 模式下 浮点分数的 **gcd** 值是 1.0。

```
gcd(18,33)                          3
```

gcd(*List1***, *List2*)** ⇒ *数组*

返回 *List1* 和 *List2* 中对应元素的最大公约数。

```
gcd({12,14,16},{9,7,5})             {3,7,1}
```

gcd(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

返回 *Matrix1* 和 *Matrix2* 中对应元素的最大公约数。

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

geomCdf()

目录 >

geomCdf(*p*,*lowBound*,*upBound*) ⇒ 如果 *lowBound* 和 *upBound* 是数值，则结果为*数值*；如果 *lowBound* 和 *upBound* 是数组，则结果为*数组*

geomCdf(*p*,*upBound*) ，P(1≤X≤*upBound*) ⇒ 如果 *upBound* 是数值，则结果为 *数值*；如果 *upBound* 是数组，则结果为*数组*

计算具有指定成功概率 *p* 的从 *lowBound* 到 *upBound* 的累积几何概率。

对于 P(X ≤ *upBound*) 设置 *lowBound*=1

geomPdf()

目录 >

geomPdf(*p*,*XVal*) ⇒ 如果 *XVal* 是数值，则结果为*数值*，如果 *XVal* 是数组，则结果为*数组*

计算具有指定成功概率 p 的离散几何分布的 *XVal*（ 即出现第一次成功的尝试次数 ）的概率。

getDenom()

目录 >

getDenom(*Fraction1*) ⇒ *值*

将自变量转换为带有化简公分母的表达式 然后返回其公分母。

$x:=5: y:=6$	6
$\text{getDenom}\left(\frac{x+2}{y-3}\right)$	3
$\text{getDenom}\left(\frac{2}{7}\right)$	7
$\text{getDenom}\left(\frac{1}{x}+\frac{y^2+y}{y^2}\right)$	30

getLangInfo()

目录 >

getLangInfo() ⇒ *字符串*

返回一个字符串 其对应于当前活动语言的缩写名称。例如 您可以在程序或函数中使用它来确定当前语言。

getLangInfo()	"en"

英语 = “en”
丹麦语 = “da”
德语 = “de”
芬兰语 = “fi”
法语 = “fr”
意大利语 = “it”
荷兰语 = “nl”
荷兰语（ 比利时 ）= “nl_BE”
挪威语 = “no”
葡萄牙语 = “pt”
西班牙语 = “es”
瑞典语 = “sv”

getLockInfo()

目录 >

getLockInfo(*Var*) ⇒ *值*

返回变量 *Var* 的当前锁定 / 解锁状态。

值 =**0**： *Var* 已解锁或不存在。

值 =**1**： *Var* 已锁定且无法修改或删除。

请参阅 **Lock**（ 第 56 页 ）和 **unLock**（ 第 111 页 ）。

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()

目录 >

getMode(*ModeNameInteger*) ⇒ *值*

getMode(0) ⇒ *数组*

getMode(*ModeNameInteger*) 返回一个数值 该值代表 *ModeNameInteger* 模式的当前设置。

getMode(0) 返回一个包含数字对的数组。每对包含一个模式整数和一个设置整数。

有关各种模式及其设置的清单 请参阅下表。

如果您使用 **getMode(0)** ➞ *var* 保存设置 则可以在函数或程序中使用 **setMode(***var***)** 来临时还原设置以仅在该函数或程序内执行。请参阅 **setMode()**（ 第 91 页 ）。

getMode(0)	{1,1,2,1,3,1,4,1,5,1,6,1,7,1}
getMode(1)	1
getMode(7)	1

模式 名称	模式 整数	设置整数
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
Vector Format	**6**	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base	**7**	**1**=Decimal, **2**=Hex, **3**=Binary

getNum()

目录 >

getNum(*Fraction1*) ⇒ *值*

将自变量转换为化简公分母的表达式　然后返回其分子。

$x:=5: y:=6$	6
$\text{getNum}\left(\frac{x+2}{y-3}\right)$	7
$\text{getNum}\left(\frac{2}{7}\right)$	2
$\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$	11

getType()

目录 >

getType(*变量***)** ⇒ *字符串*

返回表示变量 *变量* 数据类型的字符串。

如果没有定义 *变量*　则返回字符串 **"NONE"**。

$\{1,2,3\} \rightarrow temp$	$\{1,2,3\}$
$\text{getType}(temp)$	"LIST"
$2{:}3\cdot i \rightarrow temp$	$3\cdot i$
$\text{getType}(temp)$	"EXPR"
DelVar *temp*	*Done*
$\text{getType}(temp)$	"NONE"

getVarInfo()

目录 >

getVarInfo() ⇒ *矩阵*或 *字符串*

getVarInfo(*LibNameString***)** ⇒ *矩阵*或 *字符串*

getVarInfo() 返回当前问题中定义的所有变量和库对象的信息矩阵（ 变量名称　类型　库可访问性和锁定 / 解锁状态 ）。

如果没有定义任何变量　**getVarInfo()** 会返回字符串 **"NONE"**。

getVarInfo(*LibNameString***)** 返回库 *LibNameString* 中定义的所有库对象的信息矩阵。 *LibNameString* 必须为字符串（ 引号中包含的文本 ）或字符串变量。

如果库 *LibNameString* 不存在　则会出现错误。

$\text{getVarInfo}()$	"NONE"
Define $x=5$	*Done*
Lock x	*Done*
Define LibPriv $y=\{1,2,3\}$	*Done*
Define LibPub $z(x)=3\cdot x^2-x$	*Done*
$\text{getVarInfo}()$	$\begin{bmatrix} x & \text{"NUM"} & \text{"}\square\text{"} & 1 \\ y & \text{"LIST"} & \text{"LibPriv "} & 0 \\ z & \text{"FUNC"} & \text{"LibPub "} & 0 \end{bmatrix}$
$\text{getVarInfo}(tmp3)$	"Error: Argument must be a string"
$\text{getVarInfo}(\text{"tmp3"})$	$\begin{bmatrix} volcyl2 & \text{"NONE"} & \text{"LibPub "} & 0 \end{bmatrix}$

getVarInfo()

目录 >

请注意左侧示例　其中 **getVarInfo()** 的结果分配给变量 *vs*。由于 *vs* 的第 2 行或第 3 行中至少有一个元素（ 如变量 *b* ）重新计算为矩阵　因此尝试显示这些行时返回一条 "Invalid list or matrix" 的错误消息。

当使用 *Ans* 重新计算 **getVarInfo()** 结果时也可能出现此错误。

系统报出上述错误是因为当前版本的软件不支持广义的矩阵结构（ 其中矩阵的元素可以是矩阵　也可以是数组 ）。

$a:=1 \quad 1$

$b:=[1\ \ 2] \quad [1\ \ 2]$

$c:=[1\ \ 3\ \ 7] \quad [1\ \ 3\ \ 7]$

$vs:=\text{getVarInfo}() \quad \begin{bmatrix} a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ b & \text{"MAT"} & \text{"}\square\text{"} & 0 \\ c & \text{"MAT"} & \text{"}\square\text{"} & 0 \end{bmatrix}$

$vs[1] \quad [1\ \ \text{"NUM"}\ \ \text{"}\square\text{"}\ \ 0]$

$vs[1,1] \quad 1$

$vs[2] \quad \text{"Error: Invalid list or matrix"}$

$vs[2,1] \quad [1\ \ 2]$

Goto

目录 >

Goto *labelName*

将控制转至标签 *labelName* 处。

labelName 必须在同一函数中使用 **Lbl** 指令定义。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

```
Define g()=Func                    Done
  Local temp,i
  0→temp
  1→i
  Lbl top
  temp+i→temp
  If i<10 Then
  i+1→i
  Goto top
  EndIf
  Return temp
  EndFunc
g()                                55
```

▶Grad

目录 >

Expr1 ▶ **Grad** ⇒ *表达式*

将 *Expr1* 转换为百分度角度测量值。

注意：您可以通过在计算机键盘上键入 **@>Grad** 插入此运算符。

在 Degree 角度模式下：

$(1.5)\blacktriangleright\text{Grad} \quad (1.66667)^g$

在 Radian 角度模式下：

$(1.5)\blacktriangleright\text{Grad} \quad (95.493)^g$

I

identity()

目录 >

identity(*Integer***)** ⇒ *矩阵*

返回维数为 *Integer* 的单位矩阵。

Integer 必须为正整数。

$\text{identity}(4) \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

If *BooleanExpr Statement*

If *BooleanExpr* **Then**
Block
EndIf

如果 *BooleanExpr* 计算结果为 true，则执行一条语句 *Statement* 或语句块 *Block*。然后继续执行。

如果 *BooleanExpr* 计算结果为 false，则继续执行而不执行该语句或语句块。

Block 可以是一条语句，也可以是以 **":"** 字符分隔的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（而不是 enter）输入多行定义。在计算机键盘上，按住 **Alt** 然后按 **Enter**。

If *BooleanExpr* **Then**
Block1
Else
Block2
EndIf

如果 *BooleanExpr* 计算结果为 true，则执行 *Block1* 然后跳过 *Block2*。

如果 *BooleanExpr* 计算结果为 false，则跳过 *Block1* 但执行 *Block2*。

Block1 和 *Block2* 可以是一条语句。

If *BooleanExpr1* **Then**
Block1
ElseIf *BooleanExpr2* **Then**
Block2
⋮
ElseIf *BooleanExprN* **Then**
BlockN
EndIf

允许程序有分支。如果 *BooleanExpr1* 计算结果为 true，则执行 *Block1*。如果 *BooleanExpr1* 计算结果为 false，则计算 *BooleanExpr2* 的值，以此类推。

```
Define g(x)=Func                    Done
              If x<0 Then
              Return x^2
              EndIf
              EndFunc
g(-2)                               4
```

```
Define g(x)=Func                    Done
              If x<0 Then
              Return -x
              Else
              Return x
              EndIf
              EndFunc
g(12)                               12
g(-12)                              12
```

```
Define g(x)=Func
              If x<-5 Then
              Return 5
              ElseIf x>-5 and x<0 Then
              Return -x
              ElseIf x≥0 and x≠10 Then
              Return x
              ElseIf x=10 Then
              Return 3
              EndIf
              EndFunc
                                    Done
g(-4)                               4
g(10)                               3
```

ifFn()

目录 >

ifFn(*BooleanExpr*,*Value_If_true* [,*Value_If_false* [,*Value_If_unknown*]]) ⇒ *表达式 数组或矩阵*

计算布尔表达式 *BooleanExpr*（ 或 *BooleanExpr* 中的每个元素 ）并基于以下规则生成结果：

- *BooleanExpr* 可以检验单个值 数组或矩阵。
- 如果 *BooleanExpr* 中元素的计算结果为 true 则返回 *Value_If_true* 中的对应元素。
- 如果 *BooleanExpr* 中元素的计算结果为 true 则返回 *Value_If_false* 中的对应元素。如果您省略 *Value_If_false* 则返回 undef。
- 如果 *BooleanExpr* 中元素的计算结果为既不是 true 也不是 false 则返回 *Value_If_unknown* 中的对应元素。如果您省略 *Value_If_unknown* 则返回 undef。
- 如果 **ifFn()** 函数中的第二个 第三个或第四个自变量是一个表达式 则布尔检验将应用到 *BooleanExpr* 的所有位置。

注意：如果简化的 *BooleanExpr* 语句涉及数组或矩阵 则所有其他数组或矩阵自变量必须拥有相同的维数 并且结果也将拥有相同的维数。

ifFn({1,2,3}<2.5,{5,6,7},{8,9,10})
{5,6,10}

检验值 **1** 小于 2.5 因此其对应的 *Value_If_True* 元素 **5** 将复制到结果数组。

检验值 **2** 小于 2.5 因此其对应的 *Value_If_True* 元素 **6** 将复制到结果数组。

检验值 **3** 大于 2.5 因此其对应的 *Value_If_False* 元素 **10** 将复制到结果数组。

ifFn({1,2,3}<2.5,4,{8,9,10}) {4,4,10}

Value_If_true 为单一值 对应于任意选定位置。

ifFn({1,2,3}<2.5,{5,6,7}) {5,6,undef}

Value_If_false 未指定 因此使用 Undef。

ifFn({2,"a"}<2.5,{6,7},{9,10},"err")
{6,"err"}

一个元素选自 *Value_If_true*。一个元素选自 *Value_If_unknown*。

imag()

目录 >

imag(*Value1***)** ⇒ *值*

返回自变量的虚部。

imag(1+2·*i*) 2

imag(*List1***)** ⇒ *数组*

返回一个数组 其组成为自变量数组中各元素的虚部。

imag({-3,4-*i*,*i*}) {0,-1,1}

imag(*Matrix1***)** ⇒ *矩阵*

返回一个矩阵 其组成为自变量数组中各元素的矩阵。

$$\text{imag}\left(\begin{bmatrix} 1 & 2 \\ i\cdot 3 & i\cdot 4 \end{bmatrix}\right) \qquad \begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$$

Indirection

请参阅 **#()**（ 第 129 页 ）。

inString()

目录 >

inString(*srcString, subString*[*, Start*]) ⇒ *整数*

返回字符串 *srcString* 中首次出现字符串 *subString* 的起始字符位置。

如果指令中含 Start 则指定了在 *srcString* 内进行搜索的起始字符位置。默认值 = 1（ 即 *srcString* 的第一个字符 ）。

如果 *srcString* 不包含 *subString* 或 *Start* > *srcString* 的长度 则会返回零。

inString("Hello there","the") 7

inString("ABCEFG","D") 0

int()

目录 >

int(*Expr***)** ⇒ *整数*
int(*List1***)** ⇒ *数组*
int(*Matrix1***)** ⇒ *矩阵*

返回小于或等于自变量的最大整数。此函数类似于 **floor()**。

自变量可以是实数　也可以是复数。

对于数组或矩阵　返回各元素的最大整数。

int(-2.5)	-3.
int([-1.234 0 0.37])	[-2. 0 0.]

intDiv()

目录 >

intDiv(*Number1***, *Number2***)** ⇒ *整数*
intDiv(*List1***, *List2***)** ⇒ *数组*
intDiv(*Matrix1***, *Matrix2***)** ⇒ *矩阵*

返回 (*Number1* ÷ *Number2*) 的带符号整数部分。

对于数组和矩阵　返回每个元素对 (argument 1 ÷ argument 2) 的带符号整数部分。

intDiv(-7,2)	-3
intDiv(4,5)	0
intDiv({12,-14,-16},{5,4,-3})	{2,-3,5}

interpolate()

目录 >

interpolate(*x 值***, *x 数组***, *y 数组***, *y 导数数组***)** ⇒ *数组*

此函数进行以下操作：

给定 *x 数组*, *y 数组* =**f(***x 数组***)** 和 *y 导数数组* =**f'(***x 数组***)** 其中 **f** 为未知函数　使用三次插值求解函数 **f** 在 *x 值*处的近似值。假设 *x 数组*是单调递增或递减数字的数组　但即使不是　此函数也可返回值。此函数在 *x 数组*中查找包含 *x 值*的区间 [*x 数组* [i], *x 数组* [i+1]]。如果找到这类区间　它将返回一个 **f(***x 值***)** 的插值　否则　它将返回 **undef**。

x 数组　*y 数组* 和 *y 导数数组*必须为相同的维度 (≥ 2) 并且包含简化为数字的表达式。

*x 值*可以是数字或数字数组。

微分方程：
y'=-3·*y*+6·*t*+5 和 *y*(0)=5

rk:=rk23(-3·*y*+6·*t*+5,*t*,*y*,{0,10},5,1)

0.	1.	2.	3.	4.
5.	3.19499	5.00394	6.99957	9.00593

要查看完整结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

使用 interpolate() 函数可计算 x 值数组的函数值：

xvaluelist:=seq(*i*,*i*,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,

xlist:=mat▶list(*rk*[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}

ylist:=mat▶list(*rk*[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.9978

yprimelist:=-3·*y*+6·*t*+5|*y*=*ylist* and *t*=*xlist*
{-10.,1.41503,1.98819,2.00129,1.98221,2.006

interpolate(*xvaluelist*,*xlist*,*ylist*,*yprimelist*)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011

invχ²()

目录 >

invχ²(*Area***,*df*)**
invChi2(*Area***,*df*)**

计算对于曲线下的给定 *Area*　由自由度 *df* 指定的反向累积 χ^2（卡方）概率函数。

invF()

目录 >

invF(*Area*,*dfNumer*,*dfDenom***)**
invF(*Area*,*dfNumer*,*dfDenom***)**

计算对于曲线下的给定 *Area* 由 *dfNumer* 和 *dfDenom* 指定的反向累积 F 分布函数。

invNorm()

目录 >

invNorm(*Area*[,μ[,σ]])

计算对于正态分布曲线下的给定 *Area* 由 μ 和 σ 指定的反向累积正态分布函数。

invt()

目录 >

invt(*Area*,*df***)**

计算对于曲线下的给定 *Area* 由自由度 *df* 指定的反向累积学生 t 概率函数。

iPart()

目录 >

iPart(*Number***)** ⇒ *整数*
iPart(*List1***)** ⇒ *数组*
iPart(*Matrix1***)** ⇒ *矩阵*

返回自变量的整数部分。

对于数组和矩阵 返回各元素的整数部分。

自变量可以是实数 也可以是复数。

iPart(-1.234)	-1.
$\text{iPart}\left(\left\{\frac{3}{2},-2.3,7.003\right\}\right)$	{1,-2.,7.}

irr()

目录 >

irr(*CF0*,*CFList* [,*CFFreq*]) ⇒ *值*

计算投资内部收益率的财务函数。

CF0 是时间为 0 时的初始现金流 该值必须为实数。

CFList 是一个由初始现金流 CF0 之后的现金流金额组成的数组。

CFFreq 是一个可选的数组 其中各元素指定归组（ 连续 ）现金流金额（ 即 *CFList* 中的对应元素 ）的出现频率。默认值为 1 如果您输入值 这些值必须为 < 10,000 的正整数。

注意：另请参阅 **mirr()**（ 第 62 页 ）。

list1:={6000,-8000,2000,-3000}	{6000,-8000,2000,-3000}
list2:={2,2,2,1}	{2,2,2,1}
irr(5000,*list1*,*list2*)	-4.64484

isPrime()

目录 >

isPrime(*Number***)** ⇒ *布尔常数表达式*

若 *数字* 为 ≥2 的整数，则返回 true 或 false 来指明该整数是否只能被自身和 1 整除。

如果 *Number* 超过 306 位且没有 ≤1021 的因数，则 **isPrime(***Number***)** 会显示出错消息。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（而不是 enter）输入多行定义。在计算机键盘上，按住 **Alt** 然后按 **Enter**。

```
isPrime(5)                                  true
isPrime(6)                                  false
```

用于找出指定数值后下一个质数的函数：

```
Define nextprim(n)=Func                     Done
                   Loop
                   n+1→n
                   If isPrime(n)
                   Return n
                   EndLoop
                   EndFunc
nextprim(7)                                 11
```

isVoid()

目录 >

isVoid(*Var***)** ⇒ *布尔常数表达式*

isVoid(*Expr***)** ⇒ *布尔常数表达式*

isVoid(*List***)** ⇒ *布尔常数表达式数组*

返回 true 或 false 以指明自变量是否为空值数据类型。

有关空值元素的更多信息，请参阅第 134 页。

```
a:=_                                        _
isVoid(a)                                   true
isVoid({1,_,3})                             {false,true,false}
```

L

Lbl

目录 >

Lbl *labelName*

在函数内定义名称为 *labelName* 的标签。

您可以使用 **Goto** *labelName* 指令将控制转移到紧跟标签之后的指令。

labelName 必须符合与变量名称相同的命名要求。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（而不是 enter）输入多行定义。在计算机键盘上，按住 **Alt** 然后按 **Enter**。

```
Define g()=Func                             Done
           Local temp,i
           0→temp
           1→i
           Lbl top
           temp+i→temp
           If i<10 Then
           i+1→i
           Goto top
           EndIf
           Return temp
           EndFunc
g()                                         55
```

lcm()

目录 >

lcm(*Number1***, *Number2***)** ⇒ *表达式*

lcm(*List1***, *List2***)** ⇒ *数组*

lcm(*Matrix1***, *Matrix2***)** ⇒ *矩阵*

返回两个自变量的最小公倍数。两个分数的 **lcm** 值是其分子的 **lcm** 值除以其分母的 **gcd** 值。浮点分数的 **lcm** 是其乘积。

对于两个数组或矩阵　将返回各对应元素的最小公倍数。

$\text{lcm}(6,9)$ → 18

$\text{lcm}\left\{\left\{\frac{1}{3},-14,16\right\},\left\{\frac{2}{15},7,5\right\}\right\}$ → $\left\{\frac{2}{3},14,80\right\}$

left()

目录 >

left(*sourceString***[, *Num***])** ⇒ *字符串*

返回字符串 *sourceString* 中最左边的 *Num* 个字符。

如果您省略 *Num*　则会返回整个 *sourceString*。

left(*List1***[, *Num***])** ⇒ *数组*

返回 *List1* 中最左边的 *Num* 个元素。

如果您省略 *Num*　则会返回整个 *List1*。

left(*Comparison***)** ⇒ *表达式*

返回方程或不等式左侧的内容。

$\text{left}(\text{"Hello"},2)$ → "He"

$\text{left}(\{1,3,-2,4\},3)$ → $\{1,3,-2\}$

libShortcut()

目录 >

libShortcut(*LibNameString***, *ShortcutNameString***[, *LibPrivFlag*])** ⇒ *变量数组*

在当前问题中创建变量组　该变量组包含指定库文档 *libNameString* 中引用的所有对象。此函数还会将组成员添加到 Variables 菜单。然后　您可以使用其 *ShortcutNameString* 引用各对象。

设置 *LibPrivFlag*=**0** 可排除专用库对象（ 默认值 ）
设置 *LibPrivFlag*=**1** 可添加专用库对象

要复制变量组　请参阅 **CopyVar**（ 第 18 页 ）。
要删除变量组　请参阅 **DelVar**（ 第 29 页 ）。

本例假定正确存储并刷新了名为 **linalg2** 的库文档 该文档包含定义为 *clearmat*　*gauss1* 和 *gauss2* 的对象。

$\text{getVarInfo}(\text{"linalg2"})$ →

clearmat	"FUNC"	"LibPub "
gauss1	"PRGM"	"LibPriv "
gauss2	"FUNC"	"LibPub "

$\text{libShortcut}(\text{"linalg2"},\text{"la"})$ → $\{la.clearmat, la.gauss2\}$

$\text{libShortcut}(\text{"linalg2"},\text{"la"},1)$ → $\{la.clearmat, la.gauss1, la.gauss2\}$

LinRegBx *X*,*Y*[,[*Freq*][,*Category*,*Include*]]

在数组 X 和 Y 上使用频率 *Freq* 计算线性回归 y = a+b·x 。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

除 *Include* 外，所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。*Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息，请参阅 "空（空值）元素"（第 134 页）。

输出变量	说明
stat.RegEqn	回归方程：a+b·x
stat.a　stat.b	回归系数
stat.r^2	确定系数
stat.r	相关系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* 限制的回归中
stat.YReg	被修改后的数组 *Y List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* 限制的回归中
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* 的频率所组成的数组

LinRegMx *X*,*Y*[,[*Freq*][,*Category*,*Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算线性回归 y = m·x+b。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

除 *Include* 外，所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。*Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息，请参阅 "空（空值）元素"（第 134 页）。

输出变量	说明
stat.RegEqn	回归方程：y = m•x+b
stat.m　stat.b	回归系数
stat.r^2	确定系数
stat.r	相关系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

LinRegtIntervals　　目录 >

LinRegtIntervals *X*,*Y*[,*F*[,**0**[,*CLev*]]]

适用于 Slope。计算斜率的 C 级置信区间。

LinRegtIntervals *X*,*Y*[,*F*[,**1**,*Xval*[,*CLev*]]]

适用于 Response。计算预测的 y 值　针对单次观察的 C 级预测区间和针对平均响应的 C 级置信区间。

结果摘要存储在 *stat.results* **变量中。（请参阅第 98 页。）**

所有数组必须维数相同。

X 和 *Y* 分别是自变量和因变量的数组。

F 是频率值组成的可选数组。*Freq* 中的每个元素指定各对应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

有关数组中空元素结果的信息　请参阅 **"** 空（ 空值 ）元素 **"**（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：a+b•x
stat.a　stat.b	回归系数
stat.df	自由度
stat.r^2	确定系数
stat.r	相关系数
stat.Resid	回归残差

仅限 Slope 类型

输出变量	说明
[stat.CLower, stat.CUpper]	斜率的置信区间
stat.ME	置信区间误差范围
stat.SESlope	斜率的标准误差
stat.s	直线的标准误差

仅限 Response 类型

输出变量	说明
[stat.CLower, stat.CUpper]	平均响应的置信区间
stat.ME	置信区间误差范围
stat.SE	平均响应的标准误差
[stat.LowerPred, stat.UpperPred]	单次观察的预测区间
stat.MEPred	预测区间误差范围
stat.SEPred	预测的标准误差
stat.$\hat{y}$	a + b·XVal

LinRegtTest 目录 >

LinRegtTest *X*,*Y*[,*Freq*[,*Hypoth*]]

计算 *X* 和 *Y* 数组的线性回归 并对方程式 $y=\alpha+\beta x$ 的斜率值 β 和相关系数 ρ 执行 *t* 检验。它对照以下三个备选假设中的一个检验零假设 $H_0:\beta=0$（ 等同于 $\rho=0$ ）。

所有数组必须维数相同。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Hypoth 是一个可选值 它指定零假设 ($H_0:\beta=\rho=0$) 将对照三个备选假设中的哪一个进行检验。

对于 $H_a:\ \beta\neq0$ 且 $\rho\neq0$（ 默认值 ） 设定 *Hypoth*=0
对于 $H_a:\ \beta<0$ 且 $\rho<0$ 设定 *Hypoth*<0
对于 $H_a:\ \beta>0$ 且 $\rho>0$ 设定 *Hypoth*>0

结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：a + b·x
stat.t	*显著性检验的 t* 统计
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	自由度
stat.a　stat.b	回归系数
stat.s	直线的标准误差
stat.SESlope	斜率的标准误差
stat.r^2	确定系数
stat.r	相关系数
stat.Resid	回归残差

linSolve()

目录 >

linSolve(*SystemOfLinearEqns*, *Var1*, *Var2*, ...**)** ⟹ *数组*
linSolve(*LinearEqn1* **and** *LinearEqn2* **and** ...,
Var1, *Var2*, ...**)** ⟹ *数组*
linSolve({*LinearEqn1*, *LinearEqn2*, ...**}**, *Var1*, *Var2*, ...**)**
⟹ *数组*
linSolve(*SystemOfLinearEqns*, **{***Var1*, *Var2*, ...**})**
⟹ *数组*
linSolve(*LinearEqn1* **and** *LinearEqn2* **and** ...,
{*Var1*, *Var2*, ...**})** ⟹ *数组*
linSolve({*LinearEqn1*, *LinearEqn2*, ...**}**, **{***Var1*, *Var2*, ...**})**
⟹ *数组*

返回一个数组　其元素为变量 *Var1*　*Var2*　.. 的解。

第一个变量必须计算为线性方程组或单个线性方程。否则将出现自变量错误。

例如　计算 **linSolve(**x=1 and x=2,x**)** 时会生成 “Argument Error”。

$\text{linSolve}\left\{\begin{bmatrix}2\cdot x+4\cdot y=3\\5\cdot x-3\cdot y=7\end{bmatrix},\{x,y\}\right\}$　$\left\{\frac{37}{26},\frac{1}{26}\right\}$

$\text{linSolve}\left\{\begin{bmatrix}2\cdot x=3\\5\cdot x-3\cdot y=7\end{bmatrix},\{x,y\}\right\}$　$\left\{\frac{3}{2},\frac{1}{6}\right\}$

$\text{linSolve}\left\{\begin{bmatrix}apple+4\cdot pear=23\\5\cdot apple-pear=17\end{bmatrix},\{apple,pear\}\right\}$
$\left\{\frac{13}{3},\frac{14}{3}\right\}$

$\text{linSolve}\left\{\begin{bmatrix}apple\cdot 4+\frac{pear}{3}=14\\-apple+pear=6\end{bmatrix},\{apple,pear\}\right\}$
$\left\{\frac{36}{13},\frac{114}{13}\right\}$

ΔList()

目录 >

ΔList(*List1***)** ⟹ *数组*

注意：您可以通过在计算机键盘上键入 **deltaList(...)** 插入此函数。

返回一个数组　其组成为 *List1* 中两个相邻元素间的差值。*List1* 中的每个元素均与 *List1* 的下一元素相减。结果数组始终比原来的 *List1* 少一个元素。

$\Delta\text{List}\{20,30,45,70\}$　$\{10,15,25\}$

list▶mat()

目录 >

list▶mat(*List* **[,** *elementsPerRow***])** ⟹ *矩阵*

返回一个将 *List* 中的元素逐行填入所得的矩阵。

如果指令中包含 elementsPerRow 则指定了每行的元素个数。默认值是 *List* 中单行的元素个数。

如果 *List* 不能填满结果矩阵 则添加零。

注意：您可以通过在计算机键盘上键入 **list@>mat(...)** 插入此函数。

$\text{list▶mat}(\{1,2,3\})$ → $\begin{bmatrix}1 & 2 & 3\end{bmatrix}$

$\text{list▶mat}(\{1,2,3,4,5\},2)$ →
$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 0\end{bmatrix}$$

ln()

ctrl ex 键

ln(*Value1***)** ⟹ *值*

ln(*List1***)** ⟹ *数组*

返回自变量的自然对数。

对于数组 返回各元素的自然对数。

$\ln(2.)$ → 0.693147

如果复数格式模式为 Real：

$\ln(\{-3,1.2,5\})$ → "Error: Non−real calculation"

如果复数格式模式为 Rectangular：

$\ln(\{-3,1.2,5\})$ →
$\{1.09861+3.14159\cdot i, 0.182322, 1.60944\}$

ln(*squareMatrix1***)** ⟹ *方阵*

返回 *squareMatrix1* 的矩阵自然对数 此计算不同于计算每个元素的自然对数。有关计算方法的信息 请参阅 **cos()**。

squareMatrix1 必须可对角化 结果始终包含浮点数。

在 Radian 角度模式和 Rectangular 复数格式下：

$$\ln\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow$$
$$\begin{bmatrix}1.83145+1.73485\cdot i & 0.009193-1.49086\\ 0.448761-0.725533\cdot i & 1.06491+0.623491\\ -0.266891-2.08316\cdot i & 1.12436+1.79018\cdot\end{bmatrix}$$

要查看整个结果 请按 ▲ 然后使用 ◀ 和 ▶ 移动光标。

LnReg

目录 >

LnReg *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算对数回归 y = a+b·ln(x)。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：a+b·ln(x)
stat.a　stat.b	回归系数
stat.r^2	变换数据的线性确定系数
stat.r	变换数据 (ln(x), y) 的相关系数
stat.Resid	与对数模型相关的残差
stat.ResidTrans	与变换数据的线性拟合相关的残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组　实际用在基于 *Freq*　*Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

Local

目录 >

Local *Var1*[, *Var2*] [, *Var3*] ...

指定的 *vars* 为局部变量。这些变量仅在函数求值过程中存在　函数执行结束后即被删除。

注意：由于局部变量只是临时存在　因此可以节省内存。此外　它们不会影响任何现有的全局变量值。由于函数中不允许对全局变量的值进行修改　因此局部变量必须用于 **For** 循环以及在多行函数中用于临时保存数值。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

```
Define rollcount()=Func
                     Local i
                     1→i
                     Loop
                     If randInt(1,6)=randInt(1,6)
                     Goto end
                     i+1→i
                     EndLoop
                     Lbl end
                     Return i
                     EndFunc
                                                  Done
rollcount()                                        16
rollcount()                                        3
```

Lock

目录 >

Lock *Var1*[, *Var2*] [, *Var3*] ...
Lock *Var***.**

锁定指定的变量或变量组。锁定的变量无法修改或删除。

您不能锁定或解锁系统变量 *Ans*　并且不能锁定系统变量组 *stat.* 或 *tvm*。

注意：Lock 命令应用到解锁的变量时会清除 Redo/Undo 历史记录。

请参阅 **unLock**（ 第 111 页 ）和 **getLockInfo()**（ 第 42 页 ）。

```
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
```

log()

ctrl 10^x 键

log(*Value1*[,*Value2*]**)** ⇒ *值*

log(*List1*[,*Value2*]**)** ⇒ *数组*

返回第一个自变量以 *Value2* 为底的对数值。

注意：另请参阅**对数模板**（ 第 2 页 ）。

对于数组　返回各元素以 *Value2* 为底的对数值。

如果第二个自变量省略　则使用 10 作为底数。

$\log_{10}(2.)$ → 0.30103

$\log_{4}(2.)$ → 0.5

$\log_{3}(10)-\log_{3}(5)$ → 0.63093

如果复数格式模式为 Real：

$\log_{10}(\{-3,1.2,5\})$

"Error: Non−real calculation"

如果复数格式模式为 Rectangular：

$\log_{10}(\{-3,1.2,5\})$

$\{0.477121+1.36438\cdot i,0.079181,0.69897\}$

log(*squareMatrix1*[,*Value*]**)** ⇒ *方阵*

返回一个矩阵　其组成为 *squareMatrix1* 以 *Value* 为底的对数。此运算不同于计算每个元素以 *Value* 为底的对数值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

如果底数自变量已省略　则使用 10 作为底数。

在 Radian 角度模式和 Rectangular 复数格式下：

$$\log_{10}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}0.795387+0.753438\cdot i & 0.003993-0.6474\\ 0.194895-0.315095\cdot i & 0.462485+0.2707\\ -0.115909-0.904706\cdot i & 0.488304+0.7774\end{bmatrix}$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

Logistic

目录 >

Logistic *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算逻辑回归　$y = (c/(1+a\cdot e^{-bx}))$ 。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外　所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：$c/(1+a\cdot e^{-bx})$
stat.a stat.b stat.c	回归系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

LogisticD

目录 >

LogisticD *X*, *Y* [, [*Iterations*], [*Freq*] [, *Category*, *Include*]]

在数组 *X* 和 *Y* 上使用指定的 *Iterations* 次数 频率 *Freq* 计算逻辑回归 $y = (c/(1+a\cdot e^{-bx})+d)$。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：$c/(1+a\cdot e^{-bx})+d)$
stat.a stat.b stat.c stat.d	回归系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中。*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

Loop

目录 >

Loop
Block
EndLoop

重复执行 *Block* 中的语句。请注意　必须在 *Block* 中执行 **Goto** 或 **Exit** 指令　否则会造成死循环。

Block 是以 **":"** 字符分隔的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 下不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

```
Define rollcount()=Func
                    Local i
                    1→i
                    Loop
                    If randInt(1,6)=randInt(1,6)
                    Goto end
                    i+1→i
                    EndLoop
                    Lbl end
                    Return i
                    EndFunc
```

Done

rollcount()	16
rollcount()	3

LU

目录 >

LU *Matrix, lMatrix, uMatrix, pMatrix[, Tol]*

计算实数或复数矩阵的 Doolittle LU（ 下 - 上 ）分解值。下三角矩阵存储在 *lMatrix* 中　上三角矩阵存储在 *uMatrix* 中　而置换矩阵（ 描述计算过程中完成的行交换 ）存储在 *pMatrix* 中。

lMatrix · *uMatrix* = *pMatrix* · *矩阵*

作为可选项　如果矩阵中任何元素的绝对值小于 *Tol*　则将该元素作为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时　使用此公差。否则　*Tol* 将被忽略。

- 如果您使用 ctrl enter 或将 **Auto or Approximate** 设定为 Approximate 模式　则运算会使用浮点算法完成。
- 如果 *Tol* 省略或未使用　则默认的公差计算方法为：5E−14 ·max(dim(*Matrix*)) ·rowNorm(*Matrix*)

LU 的因式分解算法使用带有行交换的部分回转法。

$$\begin{bmatrix}6 & 12 & 18\\ 5 & 14 & 31\\ 3 & 8 & 18\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}6 & 12 & 18\\ 5 & 14 & 31\\ 3 & 8 & 18\end{bmatrix}$$

LU *m1*,*lower*,*upper*,*perm* — *Done*

lower
$$\begin{bmatrix}1 & 0 & 0\\ \frac{5}{6} & 1 & 0\\ \frac{1}{2} & \frac{1}{2} & 1\end{bmatrix}$$

upper
$$\begin{bmatrix}6 & 12 & 18\\ 0 & 4 & 16\\ 0 & 0 & 1\end{bmatrix}$$

perm
$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

M

mat▶list()

目录 >

mat▶list(*Matrix***)** ⇒ *数组*

返回一个数组　其组成为 *Matrix* 中的元素。这些元素将从 *Matrix* 逐行复制。

注意：您可以通过在计算机键盘上键入 **mat@>list(...)** 插入此函数。

mat▶list($\begin{bmatrix}1 & 2 & 3\end{bmatrix}$) → $\{1,2,3\}$

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$$

mat▶list(*m1*) → $\{1,2,3,4,5,6\}$

max()

目录 >

max(*Value1***, *Value2*)** ⇒ *表达式*

max(*List1***, *List2*)** ⇒ *数组*

max(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

返回两个自变量中的最大值。如果自变量为两个数组或矩阵 则返回一个数组或矩阵 其组成为这两个数组或矩阵中两个对应元素中的最大值。

$\max(2.3,1.4)$ → 2.3

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

max(*List***)** ⇒ *表达式*

返回 *list* 中的最大元素。

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matrix1***)** ⇒ *矩阵*

返回一个行向量 其元素为 *Matrix1* 中每列的最大元素。

空（ 空值 ）元素将被忽略。有关空元素的更多信息 请参阅第 134 页。

$\max\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}$ → $\begin{bmatrix} 1 & 0 & 7 \end{bmatrix}$

注意：另请参阅 **min()**。

mean()

目录 >

mean(*List*[, *freqList*]) ⇒ *表达式*

返回 *List* 中各元素的平均值。

freqList 中的元素为 *List* 中各对应元素出现的次数。

$\text{mean}(\{0.2,0,1,-0.3,0.4\})$ → 0.26

$\text{mean}(\{1,2,3\},\{3,2,1\})$ → $\frac{5}{3}$

mean(*Matrix1*[, *freqMatrix*]) ⇒ *矩阵*

返回一个行向量 其元素为 *Matrix1* 中各对应列元素的平均值。

freqMatrix 中的元素为 *Matrix1* 中各对应元素出现的次数。

空（ 空值 ）元素将被忽略。有关空元素的更多信息 请参阅第 134 页。

在 Rectangular 向量格式模式下：

$\text{mean}\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}$ → $\begin{bmatrix} -0.133333 & 0.833333 \end{bmatrix}$

$\text{mean}\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & \frac{-1}{2} \end{bmatrix}$ → $\begin{bmatrix} \frac{-2}{15} & \frac{5}{6} \end{bmatrix}$

$\text{mean}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix},\begin{bmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{bmatrix}\right)$ → $\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

median()

目录 >

median(*List*[, *freqList*]) ⇒ *表达式*

返回 *List* 中元素的中位数。

freqList 中的元素为 *List* 中各对应元素出现的次数。

$\text{median}(\{0.2,0,1,-0.3,0.4\})$ → 0.2

median()

目录 >

median(*Matrix1*[, *freqMatrix*]) ⇒ *矩阵*

返回一个行向量 其组成为 *Matrix1* 中各列的中位数。

freqMatrix 中的元素为 *Matrix1* 中各对应元素出现的次数。

注意：

- 数组或矩阵中的所有条目必须简化为数值。
- 数组或矩阵中的空（ 空值 ）元素将被忽略。有关空元素的更多信息 请参阅第 134 页。

median([[0.2, 0], [1, −0.3], [0.4, −0.5]]) → [0.4 −0.3]

MedMed

目录 >

MedMed *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算中线 y = (m·x+b) 。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	中位数 - 中位数线方程：m·x+b
stat.m stat.b	模型系数
stat.Resid	中位数 - 中位数线残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

mid()

目录 >

mid(*sourceString*, *Start*[, *Count*]) ⇒ *字符串*

返回字符串 *sourceString* 中从第 *Start* 个字符开始的 *Count* 个字符。

如果 *Count* 已省略或大于 *sourceString* 的维数 则返回 *sourceString* 中从第 *Start* 个字符开始的所有字符。

Count 必须 ≥ 0。如果 *Count* = 0 则返回空字符串。

mid("Hello there",2) → "ello there"

mid("Hello there",7,3) → "the"

mid("Hello there",1,5) → "Hello"

mid("Hello there",1,0) → ""

mid()

目录 >

mid(*sourceList***, *Start* [, *Count*])** ⇒ *数组*

返回 *sourceList* 中从第 *Start* 个元素开始的 *Count* 个元素。

如果 *Count* 已省略或大于 *sourceList* 的维数　则返回 *sourceList* 中从第 *Start* 个字符开始的所有元素。

Count 必须 ≥ 0。如果 Count = 0　则会返回空数组。

mid(*sourceStringList***, *Start*[, *Count*])** ⇒ *数组*

返回字符串数组 *sourceStringList* 中从第 *Start* 个元素开始的 *Count* 个字符串。

```
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)                  {}

mid({"A","B","C","D"},2,2)
                                    {"B","C"}
```

min()

目录 >

min(*Value1***, *Value2*)** ⇒ *表达式*

min(*List1***, *List2*)** ⇒ *数组*

min(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

返回两个自变量中的最小值。如果自变量为两个数组或矩阵　则返回一个数组或矩阵　其组成为这两个数组或矩阵中两个对应元素中的最小值。

min(*List***)** ⇒ *表达式*

返回 *List* 中的最小元素。

min(*Matrix1***)** ⇒ *矩阵*

返回一个行向量　其元素为 *Matrix1* 中每列的最小元素。

注意：另请参阅 **max()**。

```
min(2.3,1.4)                        1.4
min({1,2},{-4,3})                   {-4,2}

min({0,1,-7,1.3,0.5})               -7

min([1  -3  7 ])                    [-4  -3  0.3]
   ([-4  0  0.3])
```

mirr()

目录 >

mirr(*financeRate***,***reinvestRate***,***CF0***,***CFList***[,***CFFreq***])**

返回投资修改的内部收益率的财务函数。

financeRate 是现金流款项的付款利率。

reinvestRate 是现金流再投资的利率。

CF0 是时间为 0 时的初始现金流　该值必须为实数。

CFList 是一个由初始现金流 CF0 之后的现金流金额组成的数组。

CFFreq 是一个可选的数组　其中各元素指定归组（ 连续 ）现金流金额（ 即 *CFList* 中的对应元素 ）的出现频率。默认值为 1　如果您输入值　这些值必须为 < 10,000 的正整数。

注意：另请参阅 **irr()**（ 第 48 页 ）。

```
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
```

mod()

目录 >

mod(*Value1***, *Value2*)** ⇒ *表达式*

mod(*List1***, *List2*)** ⇒ *数组*

mod(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

根据如下恒等式所定义　返回第一个自变量对第二个自变量取的模：

mod(x,0) = x

mod(x,y) = x − y floor(x/y)

当第二个自变量为非零时　其结果随该自变量呈周期性变化。结果要么为零　要么与第二个自变量有相同的符号。

如果自变量为两个数组或两个矩阵　则返回一个数组或矩阵　其组成为这两个数组或矩阵中两个对应元素的模数。

注意：另请参阅 **remain()**　页码 83

$\mathrm{mod}(7,0)$ 7

$\mathrm{mod}(7,3)$ 1

$\mathrm{mod}(-7,3)$ 2

$\mathrm{mod}(7,-3)$ −2

$\mathrm{mod}(-7,-3)$ −1

$\mathrm{mod}(\{12,-14,16\},\{9,7,-5\})$ $\{3,0,-4\}$

mRow()

目录 >

mRow(*Value***, *Matrix1***, *Index***)** ⇒ *矩阵*

返回 *Matrix1* 的副本　其中第 *Index* 行的元素被替换为 *Matrix1* 中的对应元素乘以 *Value* 的值。

$\mathrm{mRow}\left(\frac{-1}{3},\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},2\right)$ $\begin{bmatrix}1 & 2\\ -1 & \frac{-4}{3}\end{bmatrix}$

mRowAdd()

目录 >

mRowAdd(*Value***, *Matrix1***, *Index1***, *Index2***)** ⇒ *矩阵*

返回 *Matrix1* 的副本　其中 *Matrix1* 的第 *Index2* 行被替换为：

Value · row *Index1* + row *Index2*

$\mathrm{mRowAdd}\left(-3,\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},1,2\right)$ $\begin{bmatrix}1 & 2\\ 0 & -2\end{bmatrix}$

MultReg

目录 >

MultReg *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]]

计算数组 *Y* 关于数组 *X1*、*X2*、...、*X10* 的多元线性回归。结果摘要存储在 stat.results 变量中。（请参阅第 98 页。）

所有数组必须维数相同。

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：b0+b1·x1+b2·x2+ ...
stat.b0　stat.b1 ...	回归系数
stat.R^2	多元确定系数
stat.$\hat{y}$List	$\hat{y}$List = b0+b1·x1+ ...
stat.Resid	回归残差

MultRegIntervals *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]],*XValList*[,*CLevel*]

计算预测的 y 值 针对单次观察的 C 级预测区间和针对平均响应的 C 级置信区间。

结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

所有数组必须维数相同。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：b0+b1·x1+b2·x2+ ...
stat.ŷ	点估计：ŷ = b0 + b1 · xl + ... for *XValList*
stat.dfError	误差自由度
stat.CLower stat.CUpper	平均响应的置信区间
stat.ME	置信区间误差范围
stat.SE	平均响应的标准误差
stat.LowerPred stat.UpperrPred	单次观察的预测区间
stat.MEPred	预测区间误差范围
stat.SEPred	预测的标准误差
stat.bList	回归系数数组 {b0,b1,b2,...}
stat.Resid	回归残差

MultReg *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]]

多元线性回归检验计算给定数据的多元线性回归并提供系数的全局 *F* 检验统计和 *t* 检验统计。

结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出

输出变量	说明
stat.RegEqn	回归方程：b0+b1·x1+b2·x2+ ...
stat.F	全局 *F* 检验统计
stat.PVal	与全局 *F* 统计相关的 P 值
stat.R^2	多元确定系数

输出变量	说明
stat.AdjR^2	调整的多元确定系数
stat.s	误差的标准差
stat.DW	Durbin-Watson 统计　用于确定模型中是否存在一阶自动关联
stat.dfReg	回归自由度
stat.SSReg	回归平方和
stat.MSReg	回归均值平方
stat.dfError	误差自由度
stat.SSError	误差平方和
stat.MSError	误差均值平方
stat.bList	{b0,b1,...} 系数数组
stat.tList	t 统计数组　一个元素对应 bList 中的一个系数
stat.PList	每个 t 统计的 P 值数组
stat.SEList	bList 中系数的标准误差数组
stat.$\hat{y}$List	$\hat{y}$List = b0+b1·x1+ ...
stat.Resid	回归残差
stat.sResid	标准化残差　通过残差除以其标准差获得
stat.CookDist	Cook 距离　测量基于残差和杠杆值的观察带来的影响
stat.Leverage	测量因变量值与平均值之间的差值

N

nand

ctrl = 键

布尔表达式 1 **nand** *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 **nand** *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 **nand** *布尔矩阵 2* 返回 *布尔矩阵*

返回两个自变量的 **and** 逻辑运算的逻辑非。返回真　假或简化方程。

列表和矩阵则按元素返回对比。

整数 1 **nand** *整数 2* ⇒ *整数*

使用 **nand** 运算逐位比较实整数。在内部　两个整数都转化为带符号的 64 位二进制数。比较相应位时　若两位都是 1 则返回结果为 1　否则结果为 0。返回的值代表位结果　是根据数基模式显示的。

您可输入任意数基的整数。对于二进制或十六进制项　您必须分别使用 0b 或 0h 作为前缀。若没有前缀　则整数将被视为十进制（ 数基 10 ）。

$x\geq 3$ and $x\geq 4$	$x\geq 4$
$x\geq 3$ nand $x\geq 4$	$x<4$

3 and 4	0
3 nand 4	-1
$\{1,2,3\}$ and $\{3,2,1\}$	$\{1,2,1\}$
$\{1,2,3\}$ nand $\{3,2,1\}$	$\{-2,-3,-2\}$

nCr()

目录 >

nCr(*Value1***, *Value2***)** ⇒ *表达式*

对于 *Value1* 和 *Value2* 且 *Value1* ≥ *Value2* ≥ 0 **nCr()** 表示从 *Value1* 件东西中每次取出 *Value2* 件时可能的不同组合。（这也称为二项式系数。）

nCr(*Value***, 0)** ⇒ **1**

nCr(*Expr***, *negInteger*)** ⇒ **0**

nCr(*Expr***, *posInteger*)** ⇒ *值*（*值* − 1）**...**
（*值* − *正整数***+1)/** *正整数***!**

nCr(*Value***, *nonInteger*)** ⇒ *表达式***!/**
((*值* − *非整数***)!·** *非整数***!)**

nCr(*List1***, *List2*)** ⇒ *数组*

返回一个数组 其组成是基于两个数组中对应元素对的组合值。自变量必须是维数相同的数组。

nCr(*Matrix1***, *Matrix2*)** ⇒ *矩阵*

返回一个矩阵 其组成是基于两个矩阵中对应元素对的组合值。自变量必须是维数相同的矩阵。

$\mathrm{nCr}(z,3)|z=5$ → 10

$\mathrm{nCr}(z,3)|z=6$ → 20

$\mathrm{nCr}(\{5,4,3\},\{2,4,2\})$ → $\{10,1,3\}$

$\mathrm{nCr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$ → $\begin{bmatrix}15&10\\6&3\end{bmatrix}$

nDerivative()

目录 >

nDerivative(*Expr1***,***Var***=***Value***[,***Order***])** ⇒ *值*

nDerivative(*Expr1***,***Var***[,***Order***]) | ***Var***=***Value** ⇒ *值*

返回使用自动微分方法计算的数值导数。

指定*值*之后 该值会覆盖之前的所有变量分配或变量的所有当前 "|" 代入值。

如果变量 *Var* 不包含数值 您必须提供 *Value*。

*导数的阶数*必须为 **1** 或 **2**。

注意：nDerivative() 存在局限性：它会通过未简化的表达式进行递推运算 计算一阶导数（和二阶导数 如适用）的数值并计算每个子表达式 这可能会导致得到意外结果。

请注意右侧的示例。**x=0** 时 **x·(x^2+x)^(1/3)** 的一阶导数等于 **0**。但是 由于 **x=0** 时子表达式 **(x^2+x)^(1/3)** 的一阶导数未定义 且该值是用于计算整个表达式的导数 因此 **nDerivative()** 会将结果报告为未定义并显示警告信息。

如果您遇到此局限 请从图形上验证解。您还可以尝试使用 **centralDiff()**。

$\mathrm{nDerivative}(|x|,x=1)$ → 1

$\mathrm{nDerivative}(|x|,x)|x=0$ → undef

$\mathrm{nDerivative}(\sqrt{x-1},x)|x=1$ → undef

$\mathrm{nDerivative}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x,1\right)|x=0$ → undef

$\mathrm{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)|x=0$ → 0.000033

newList()

目录 >

newList(*numElements***)** ⇒ *数组*

返回一个维数为 *numElements* 的数组 其元素均为零。

$\mathrm{newList}(4)$ → $\{0,0,0,0\}$

newMat()

目录 >

newMat(*numRows***, *numColumns***)** ⇒ *矩阵*

返回一个全零矩阵 其行数为 *numRows* 列数为 *numColumns*。

$\mathrm{newMat}(2,3)$ → $\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}$

nfMax()

目录 >

nfMax(*Expr***, *Var*)** ⇒ *值*

nfMax(*Expr***, *Var*, *lowBound*)** ⇒ *值*

nfMax(*Expr***, *Var*, *lowBound*, *upBound*)** ⇒ *值*

nfMax(*Expr, Var***) | *lowBound*≤*Var*≤*upBound*** ⇒ *值*

返回 *Expr* 为局部最大值时 变量 *Var* 的候选数值。

如果提供了 *下界* 和 *上界* 则函数会在闭区间 [*下界*, *上界*] 寻找局部最大值。

$\text{nfMax}(-x^2-2\cdot x-1,x)$ $-1.$

$\text{nfMax}(0.5\cdot x^3-x-2,x,-5,5)$ -0.816497

nfMin()

目录 >

nfMin(*Expr***, *Var*)** ⇒ *值*

nfMin(*Expr***, *Var*, *lowBound*)** ⇒ *值*

nfMin(*Expr***, *Var*, *lowBound*, *upBound*)** ⇒ *值*

nfMin(*Expr, Var***) | *lowBound*≤*Var*≤*upBound*** ⇒ *值*

返回 *Expr* 为局部最小值时 变量 *Var* 的候选数值。

如果提供了 *下界* 和 *上界* 则函数会在闭区间 [*下界*, *上界*] 寻找局部最低限度值。

$\text{nfMin}(x^2+2\cdot x+5,x)$ $-1.$

$\text{nfMin}(0.5\cdot x^3-x-2,x,-5,5)$ 0.816497

nInt()

目录 >

nInt(*Expr1, Var, Lower, Upper***)** ⇒ *表达式*

如果被积函数 *Expr1* 未包含除 *Var* 以外的其他变量 且 *Lower* 和 *Upper* 为常数 正 ∞ 或负 ∞ 则 **nInt()** 会返回 ∫**(***Expr1*, *Var*, *Lower*, *Upper***)** 的近似值。此近似值是被积函数在区间 *Lower*<*Var*<*Upper* 上部分样本值的加权平均值。

运算目标是获得六位有效数字。如果目标实现或增加样本也不能对结果产生有意义的改善时 所采用的算法将会终止。

如果目标无法实现 将显示警告 (“Questionable accuracy”)。

嵌套 **nInt()** 可求多元数值积分。积分极限可能取决于积分函数外部的积分变量。

$\text{nInt}\left(e^{-x^2},x,-1,1\right)$ 1.49365

$\text{nInt}\left(\cos(x),x,-\pi,\pi+1.\text{E}-12\right)$ $-1.04144\text{E}-12$

$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right)$ 3.30423

nom()

目录 >

nom(*effectiveRate,CpY***)** ⇒ *值*

将年度有效利率 *effectiveRate* 转换为名义利率的财务函数 指定 *CpY* 作为每年复利期数的数量。

effectiveRate 必须为实数 *CpY* 必须为 > 0 的实数。

注意：另请参阅 **eff()**（ 第 32 页 ）。

$\text{nom}(5.90398,12)$ 5.75

nor ctrl = 键

布尔表达式 1 **nor** *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 **nor** *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 **nor** *布尔矩阵 2* 返回 *布尔矩阵*

返回两个自变量的 **or** 逻辑运算的逻辑非。返回真 假或简化方程。

列表和矩阵则按元素返回对比。

整数 1 **nor** *整数 2* ⇒ *整数*

使用 **nor** 运算逐位比较实整数。在内部 两个整数都转化为带符号的 64 位二进制数。比较相应位时 若两位都是 1 则返回结果为 1 否则结果为 0。返回的值代表位结果 是根据数基模式显示的。

您可输入任意数基的整数。对于二进制或十六进制项 您必须分别使用 0b 或 0h 作为前缀。若没有前缀 则整数将被视为十进制（ 数基 10 ）。

$x\geq 3 \text{ or } x\geq 4$	$x\geq 3$
$x\geq 3 \text{ nor } x\geq 4$	$x<3$
$3 \text{ or } 4$	7
$3 \text{ nor } 4$	-8
$\{1,2,3\} \text{ or } \{3,2,1\}$	$\{3,2,3\}$
$\{1,2,3\} \text{ nor } \{3,2,1\}$	$\{-4,-3,-4\}$

norm() 目录 >

norm(*Matrix***)** ⇒ *表达式*
norm(*Vector***)** ⇒ *表达式*

返回 Frobenius 范数。

$\text{norm}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$	5.47723
$\text{norm}\begin{pmatrix}1 & 2\end{pmatrix}$	2.23607
$\text{norm}\begin{pmatrix}1\\ 2\end{pmatrix}$	2.23607

normCdf() 目录 >

normCdf(*lowBound***,***upBound***[,μ[,σ]])** ⇒ 如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*，如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

计算在 *lowBound* 与 *upBound* 之间 指定 μ（ 默认值 =0 ）和 σ（ 默认值 =1 ）的正态分布概率。

对于 P(X ≤ *upBound*) 设置 *lowBound* = −9**E**999。

normPdf() 目录 >

normPdf(*XVal***[,μ[,σ]])** ⇒ 如果 *XVal* 是数值，则结果为 *数值*，如果 *XVal* 是数组，则结果为 *数组*

计算 *XVal* 为指定值时 正态分布在指定 μ 和 σ 范围内的概率密度函数。

not 目录 >

not *BooleanExpr* ⇒ *布尔表达式*

返回值为 true false 或自变量的简化形式。

$\text{not } (2\geq 3)$	true
not 0hB0▶Base16	0hFFFFFFFFFFFFFF4F
not not 2	2

not

not *Integer1* ⇒ *整数*

返回实整数的补数。在内部运算中 *Integer1* 被转换为带符号的 64 位二进制数值。各位上的数值进行反转（0 变成 1 反之亦然）从而得到其补数。结果根据进位制模式显示。

您可以输入任何数字进位制的整数。对于按二进制或十六进制输入的整数 您必须分别使用 0b 或 0h 前缀。不带前缀的整数都将被视为十进制 (base 10)。

如果您输入的十进制整数对于带符号的 64 位二进制形式来说过大 可使用对称的模数运算将该值纳入合理的范围。更多信息 请参阅 ▶Base2（第 12 页）。

在 Hex 模式下：

重要信息：零 非字母 O。

not 0h7AC36	0hFFFFFFFFFF853C9

在 Bin 模式下：

0b100101▶Base10	37
not 0b100101 0b1111111111111111111111111111111111▶	
not 0b100101▶Base10	−38

要查看整个结果 请按 ▲ 然后使用 ◀ 和 ▶ 移动光标。

注意：二进制输入最多可为 64 位（不包括 0b 前缀）。十六进制输入最多可为 16 位。

nPr()

nPr(*Value1***, *Value2***)** ⇒ *表达式*

nPr(*Expr***, *posInteger***)** ⇒ *表达式* **°§(** *表达式* **− 1)...**
(*表达式* **− 1)**

nPr(*Expr***, *nonInteger***)** ⇒ *表达式* **!/(** *表达式* **− 1)!**

对于 *Value1* 和 *Value2* 且 *Value1* ≥ *Value2* ≥ 0 **nPr()** 表示从 *Value1* 件东西中每次取出 *Value2* 件时可能的不同排列数。

nPr(*Value***, 0)** ⇒ **1**

nPr(*Value***, *negInteger***)** ⇒ **1/((** *值* **+1)·(** *值* **+2)...**
(*值* **− 1))**

nPr(*Value***, *posInteger***)** ⇒ *值* **°§(** *值* **− 1)...**
(*值* **− 1)**

nPr(*Value***, *nonInteger***)** ⇒ *值* **!/(** *值* **− 1)!**

nPr(*List1***, *List2***)** ⇒ *数组*

返回一个数组 其组成是基于两个数组中对应元素对的排列数。自变量必须是维数相同的数组。

nPr(*Matrix1***, *Matrix2***)** ⇒ *矩阵*

返回一个矩阵 其组成是基于两个矩阵中对应元素对的排列数。自变量必须是维数相同的矩阵。

$nPr(z,3)\|z=5$	60
$nPr(z,3)\|z=6$	120
$nPr(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$
$nPr\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$	$\begin{bmatrix}30&20\\12&6\end{bmatrix}$

$nPr(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$

$nPr\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$	$\begin{bmatrix}30&20\\12&6\end{bmatrix}$

npv()

目录 >

npv(*InterestRate*,*CFO*,*CFList*[,*CFFreq*]**)**

计算净现值的财务函数　现金流入和流出的现值之和。 npv 结果为正表示投资盈利。

InterestRate 是一段时间内现金流（ 资金成本 ）的折扣率。

CF0 是时间为 0 时的初始现金流　该值必须为实数。

CFList 是一个由初始现金流 *CF0* 之后的现金流金额组成的数组。

CFFreq 是一个数组　其中每个元素指定归组（ 连续 ）现金流金额（ 即 *CFList* 的对应元素 ）的出现频率。默认值为 1　如果您输入值　这些值必须为 < 10,000 的正整数。

$list1:=\{6000,-8000,2000,-3000\}$ $\{6000,-8000,2000,-3000\}$

$list2:=\{2,2,2,1\}$ $\{2,2,2,1\}$

$\text{npv}(10,5000,list1,list2)$ 4769.91

nSolve()

目录 >

nSolve(*Equation*,*Var*[**=***Guess*]**)** ⟹ *数值或错误 _ 字符串*

nSolve(*Equation*,*Var*[**=***Guess*]**,***lowBound***)**

⟹ *数值或错误 _ 字符串*

nSolve(*Equation*,*Var*[**=***Guess*]**,***lowBound*,*upBound***)**

⟹ *数值或错误 _ 字符串*

nSolve(*Equation*,*Var*[**=***Guess*]**) |** *lowBound*≤*Var*≤*upBound*

⟹ *数值或错误 _ 字符串*

对 *Equation* 的某个变量反复搜索其实数解的近似值。指定变量为：

变量
- 或 -
变量 = *实数*

例如　x 和 x=3 都是有效形式。

$\text{nSolve}(x^2+5\cdot x-25=9,x)$ 3.84429

$\text{nSolve}(x^2=4,x=-1)$ -2.

$\text{nSolve}(x^2=4,x=1)$ 2.

注意： 如果存在多个解　您可以使用估计值来帮助找到特解。

nSolve() 会尝试确定残差值为零的一点　或残差值符号相反　且大小不超过限值的相对接近的两点。如果使用样本点中的适当数值无法实现　则会返回字符串 **"no solution found"**。

$\text{nSolve}(x^2+5\cdot x-25=9,x)|x<0$ -8.84429

$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right)|r>0 \text{ and } r<0.25$ 0.006886

$\text{nSolve}(x^2=-1,x)$ "No solution found"

O

OneVar 目录 >

OneVar [**1,**]*X*[**,**[*Freq*][**,***Category*,*Include*]]
OneVar [*n*,]*X1*,*X2*[*X3*[**,**…[**,***X20*]]]

计算最多 20 个数组的单变量统计。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外　所有数组必须有相同维数。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是相应 *X* 数值的类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

数组 *X*　*Freq* 或 *Category* 中任意一个数组的空（ 空值 ）元素都会导致所有这些数组中对应元素为空值。数组 *X1* 到 *X20* 中任意一个数组的空元素都会导致所有这些数组中对应元素为空值。有关空元素的更多信息　请参阅第 134 页。

输出变量	说明
stat.$\bar{x}$	x 值的平均值
stat.Σx	x 值之和
stat.Σx^2	x^2 值之和
stat.sx	x 的样本标准差
stat.σx	x 的总体标准差
stat.n	数据点的数量
stat.MinX	x 值的最小值
stat.Q_1X	x 的第一个四分位数
stat.MedianX	x 的中位数
stat.Q_3X	x 的第三个四分位数
stat.MaxX	x 值的最大值
stat.SSX	x 平均值的方差和

布尔表达式 1 **or** *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 **or** *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 **or** *布尔矩阵 2* 返回 *布尔矩阵*

返回 true 或 false 或者原始输入的简化形式。

如果其中一个或两个表达式化简为 true 则返回 true。仅当两个表达式的计算结果均为 false 时 才返回 false。

注意：请参阅 **xor**。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中 您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

Integer1 **or** *Integer2* ⇒ *整数*

使用 or 运算逐位比较两个实整数。在内部运算中 两个整数都将转换为带符号的 64 位二进制数字。当相应位进行比较时 如果任何一个位值为 1 则结果为 1 仅当两个位值均为 0 时 结果才为 0。返回的值代表位结果 将根据 Base 模式显示。

您可以输入任何进位制的整数。对于按二进制或十六进制输入的整数 您必须分别使用 0b 或 0h 前缀。不带前缀的整数都将被视为十进制（ 基数为 10)。

如果您输入的十进制整数对于带符号的 64 位二进制形式来说过大 可使用对称的模数运算将该值纳入合理的范围。更多信息 请参阅 ▶Base2（ 第 12 页 ）。

注意：请参阅 **xor**。

Define $g(x)$=Func *Done*
If $x\le 0$ or $x\ge 5$
Goto *end*
Return $x\cdot 3$
Lbl *end*
EndFunc

$g(3)$ 9

$g(0)$ *A function did* not return *a value*

在 Hex 模式下：

0h7AC36 or 0h3D5F 0h7BD7F

重要信息：零 非字母 O。

在 Bin 模式下：

0b100101 or 0b100 0b100101

注意：二进制输入最多可为 64 位（ 不包括 0b 前缀 ）。十六进制输入最多可为 16 位。

ord() 目录 >

ord(*String***)** ⇒ *整数*
ord(*List1***)** ⇒ *数组*

返回字符串 *String* 中第一个字符的数值代码 或返回一个由 List1 中各元素的第一个字符所组成的数组。

ord("hello") 104
char(104) "h"
ord(char(24)) 24
ord({"alpha","beta"}) {97,98}

P

P▶Rx() 目录 >

P▶Rx(*rExpr***, θ***Expr***)** ⇒ *表达式*
P▶Rx(*rList***, θ***List***)** ⇒ *数组*
P▶Rx(*rMatrix***, θ***Matrix***)** ⇒ *矩阵*

返回
(r, θ) 对的等值 x 坐标值。

注意：θ 自变量可以是度 弧度或百分度 具体取决于当前的角度模式。如果自变量为表达式 您可以使用 ° G 或 r 临时更改角度模式。

注意：您可以通过在计算机键盘上键入 **P@>Rx(...)** 插入此函数。

在 Radian 角度模式下：

$P\blacktriangleright Rx(4,60°)$ 2.

$P\blacktriangleright Rx\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$
$\{-1.5,7.07107,1.3\}$

P▶Ry()

目录 >

P▶Ry(*rValue*, θ*Value*) ⇒ *值*

P▶Ry(*rList*, θ*List*) ⇒ *数组*

P▶Ry(*rMatrix*, θ*Matrix*) ⇒ *矩阵*

返回 (r, θ) 对的等值 y 坐标值。

注意： θ 自变量可以是度、弧度或百分度。具体取决于当前的角度模式。

注意： 您可以通过在计算机键盘上键入 **P@>Ry(...)** 插入此函数。

在 Radian 角度模式下：

$\mathrm{P}\blacktriangleright\mathrm{Ry}(4,60°)$ 3.4641

$\mathrm{P}\blacktriangleright\mathrm{Ry}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$ $\{-2.59808,-7.07107,0\}$

PassErr

目录 >

PassErr

将错误传递到下一级。

如果系统变量 *errCode* 为零，则 **PassErr** 不会进行任何操作。

Try...Else...EndTry 块的 **Else** 语句应使用 **ClrErr** 或 **PassErr**。如果要处理或忽略错误，请使用 **ClrErr**。如果不知道如何处理错误，请使用 **PassErr** 将其发送到下一个错误处理句柄。如果没有其他未完成的 **Try...Else...EndTry** 错误处理句柄，错误对话框将正常显示。

注意： 另请参见第 17 页的 **ClrErr** 和第 106 页的 **Try**。

输入样本的注意事项： 在手持设备的 " 计算器 " 应用程序中请按 ↵ 输入多行定义，而不要在各行末按 enter。在计算机键盘上按住 **Alt** 并按 **Enter**。

有关 **PassErr** 的示例，请参阅 **Try** 命令下的示例 2（第 106 页）。

piecewise()

目录 >

piecewise(*Expr1* [, *Cond1* [, *Expr2* [, *Cond2* [, ...]]]])

以数组形式返回分段函数的定义。您还可以使用模板创建分段函数。

注意： 另请参阅**分段模板**（第 2 页）。

Define $p(x)=\begin{cases}x, & x>0\\ \text{undef}, & x\le 0\end{cases}$ *Done*

$p(1)$ 1

$p(-1)$ undef

poissCdf()

目录 >

poissCdf(λ**,***lowBound*,*upBound***)** ⇒ 如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*；如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

poissCdf(λ**,***upBound***)**，P(0≤X≤*upBound*) ⇒ 如果 *upBound* 是数值，则结果为 *数值*；**如果** *upBound* 是数组，则结果为数组

计算具有指定平均值 λ 的离散泊松分布的累积概率。

对于 P(X ≤ *upBound*)，设置 *lowBound*=0

poissPdf()

目录 >

poissPdf(λ**,***XVal***)** ⇒ 如果 *XVal* 是数值，则结果为 *数值*，如果 *XVal* 是数组，则结果为 *数组*

计算具有指定平均值 λ 的离散泊松分布的概率。

▶Polar

目录 >

Vector **▶Polar**

注意：您可以通过在计算机键盘上键入 **@>Polar** 插入此运算符。

以极坐标形式 [r ∠θ] 显示*向量*。向量维数必须为 2 可以是行向量 也可以是列向量。

注意：**▶Polar** 是一条显示格式指令 不是转换函数。您只能在输入行结尾处使用该函数 并且 *ans* 不会得到更新。

注意：另请参阅 **▶Rect**（第 82 页）。

$[1 \quad 3.]\blacktriangleright\text{Polar} \qquad [3.16228 \quad \angle 71.5651]$

complexValue **▶Polar**

以极坐标形式显示 *complexVector*。

- Degree 角度模式下将返回 (r∠θ)。
- Radian 角度模式下将返回 $re^{i\theta}$。

complexValue 可为任意复数形式 不过 $re^{i\theta}$ 形式的输入会在 Degree 角度模式中产生错误。

注意：您必须对 (r∠θ) 形式的极坐标输入使用括号。

在 Radian 角度模式下：

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad e^{.927295\cdot i}\cdot 5$

$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Polar} \qquad e^{1.0472\cdot i}\cdot 4.$

在 Gradian 角度模式下：

$(4\cdot i)\blacktriangleright\text{Polar} \qquad (4\angle 100)$

在 Degree 角度模式下：

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad (5\angle 53.1301)$

polyEval()

目录 >

polyEval(*List1***, *Expr1*)** ⟹ *表达式*

polyEval(*List1***, *List2*)** ⟹ *表达式*

将第一个自变量看作一个降次多项式的系数 然后返回该多项式 用于计算第二个自变量的值计算。

$\text{polyEval}(\{1,2,3,4\},2) \qquad 26$

$\text{polyEval}(\{1,2,3,4\},\{2,-7\}) \qquad \{26,-262\}$

polyRoots()

目录 >

polyRoots(*Poly***, *Var*)** ⟹ *数组*

polyRoots(*ListOfCoeffs***)** ⟹ *数组*

第一种句法 **cPolyRoots(***Poly***, *Var*)** 返回一个数组 其元素为关于变量 *Var* 的多项式 *Poly* 的实数根。如果实数根不存在则返回一个空的数组：{}。

Poly 必须为扩展形式的单变量多项式。请勿使用非扩展形式 如 y^2·y+1 或 x·x+2·x+1

第二种句法 **cPolyRoots(***ListOfCoeffs***)** 返回一个数组 其元素为 *ListOfCoeffs 中系数的实数根*。

注意：另请参阅 **cPolyRoots()**（第 23 页）。

$\text{polyRoots}(y^3+1,y) \qquad \{-1\}$

$\text{cPolyRoots}(y^3+1,y) \qquad \{-1,0.5-0.866025\cdot i,0.5+0.866025\cdot i\}$

$\text{polyRoots}(x^2+2\cdot x+1,x) \qquad \{-1,-1\}$

$\text{polyRoots}(\{1,2,1\}) \qquad \{-1,-1\}$

PowerReg *X,Y* [*, Freq*] [*, Category, Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算幂回归 $y = (a \cdot (x)^b)$ 。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

除 *Include* 外，所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。*Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息，请参阅 "空（空值）元素"（第 134 页）。

输出变量	说明
stat.RegEqn	回归方程：$a \cdot (x)^b$
stat.a　stat.b	回归系数
stat.r^2	变换数据的线性确定系数
stat.r	变换数据 (ln(x), ln(y)) 的相关系数
stat.Resid	与幂模型相关的残差
stat.ResidTrans	与变换数据的线性拟合相关的残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* 限制的回归中
stat.YReg	被修改后的数组 *Y List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* 限制的回归中
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* 的频率所组成的数组

Prgm

目录 >

Prgm
Block
EndPrgm

创建用户定义程序的模板　必须与 **Define**　**Define L bPub** 或 **Define LibPriv** 命令一起使用。

Block 可以是一条语句　也可以是以 **":"** 字符分隔的或者单独行上的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

计算 GCD 并显示中间结果。

```
Define proggcd(a,b)=Prgm
                    Local d
                    While b≠0
                    d:=mod(a,b)
                    a:=b
                    b:=d
                    Disp a," ",b
                    EndWhile
                    Disp "GCD=",a
                    EndPrgm
                                        Done

proggcd(4560,450)
                                        450 60
                                        60 30
                                        30 0
                                        GCD=30
                                        Done
```

prodSeq()

???? '? Π0???? 127 "???° ?

Product (PI)

???? '? Π0???? 127 "???° ?

product()

目录 >

product(*List*[, *Start*[, *End*]]) ⇒ *表达式*

返回 *List* 所含元素的乘积。*Start* 和 *End* 为可选项。它们指定了元素的范围。

product(*Matrix1*[, *Start*[, *End*]]) ⇒ *矩阵*

返回由 *Matrix1* 中各列元素的乘积所组成的行向量。*Start* 和 *end* 为可选项。它们指定了行的范围。

空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

```
product({1,2,3,4})                          24
product({4,5,8,9},2,3)                      40

product([1 2 3; 4 5 6; 7 8 9])              [28 80 162]
product([1 2 3; 4 5 6; 7 8 9],1,2)          [4 10 18]
```

propFrac()

目录 >

propFrac(*Value1*[**,** *Var*]) ⟹ *值*

propFrac(*rational_number***)** 以整数与分数之和的形式返回 *rational_number*　其中分数与整数符号相同且分母大于分子。

$\text{propFrac}\left(\frac{4}{3}\right)$	$1+\frac{1}{3}$
$\text{propFrac}\left(\frac{-4}{3}\right)$	$-1-\frac{1}{3}$

propFrac(*rational_expression***,***Var***)** 返回适当比值及关于 *Var* 的多项式的和。在各个适当比值中　分母中 *Var* 的次数应大于分子中 *Var* 的次数。 *Var* 的同次幂将汇集在一起。各项及其因式将按主变量 *Var* 进行分类。

如果省略 *Var*　则得到一个关于主变量的适当分子展开形式。然后　先给出关于主变量的多项式部分的系数　以此类推。

您可以使用 **propFrac()** 函数表示带分数并演示带分数的加法和减法。

$\text{propFrac}\left(\frac{11}{7}\right)$	$1+\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8+\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2-\frac{29}{44}$

Q

QR

目录 >

QR *Matrix*, *qMatrix*, *rMatrix*[, *Tol*]

计算实数或复数矩阵的 Householder QR 因式分解。结果 Q 矩阵和 R 矩阵存储在指定的 *Matrix* 中。Q 矩阵为酉矩阵　R 矩阵为上三角矩阵。

作为可选项　如果矩阵中任何元素的绝对值小于 *Tol*　则将该元素将作为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时　使用此公差。否则　*Tol* 将被忽略。

- 如果您使用 ctrl enter 或将 **Auto or Approximate** 设定为 Approximate 模式　则运算会使用浮点算法完成。
- 如果 *Tol* 省略或未使用　则默认的公差计算方法为：5E⁻14 ·max(dim(*Matrix*)) ·rowNorm(*Matrix*)

QR 因式分解采用 Householder 变换进行数值运算。使用 Gram-Schmidt 进行符号运算。 *qMatName* 中的列向量是 *matrix* 所定义的空间上的规范正交基。

m1 中的浮点数值 (9.) 使得结果以浮点形式进行计算。

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}$
QR *m1*,*qm*,*rm*	*Done*
qm	$\begin{bmatrix}0.123091&0.904534&0.408248\\0.492366&0.301511&-0.816497\\0.86164&-0.301511&0.408248\end{bmatrix}$
rm	$\begin{bmatrix}8.12404&9.60114&11.0782\\0.&0.904534&1.80907\\0.&0.&0.\end{bmatrix}$
ClearAZ	*Done*

QuadReg *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

在数组 *X* 和 *Y* 上使用频率 *Freq* 计算二次多项式回归 y = a·x²+b·x+c 。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

除 *Include* 外，所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。*Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息，请参阅 "空（空值）元素"（第 134 页）。

输出变量	说明
stat.RegEqn	回归方程：a·x²+b·x+c
stat.a　stat.b stat.c	回归系数
stat.R²	确定系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

QuartReg *X*,*Y* [, *Freq*] [, *Category*, *Include*]

计算
在数组 *X* 和 *Y* 上使用频率 *Freq* 计算四次多项式回归 $y = a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$ 。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外 所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.RegEqn	回归方程：$a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$
stat.a stat.b stat.c stat.d stat.e	回归系数
stat.R^2	确定系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组 实际用在基于 *Freq* *Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

R

R▶Pθ()

目录 >

R▶Pθ (*xValue*, *yValue*) ⟹ *值*

R▶Pθ (*xList*, *yList*) ⟹ *数组*

R▶Pθ (*xMatrix*, *yMatrix*) ⟹ *矩阵*

返回 θ 坐标值
（ 与 **(*x*,*y*)** 自变量对等效的 ）。

注意： 返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

注意： 您可以通过在计算机键盘上键入 **R@>Ptheta(...)** 插入此函数。

在 Degree 角度模式下：

R▶Pθ(2,2) 45.

在 Gradian 角度模式下：

R▶Pθ(2,2) 50.

在 Radian 角度模式下：

R▶Pθ(3,2) 0.588003

R▶Pθ([3 −4 2],[0 π/4 1.5])

[0. 2.94771 0.643501]

R▶Pr()

目录 >

R▶Pr (*xValue*, *yValue*) ⟹ *值*

R▶Pr (*xList*, *yList*) ⟹ *数组*

R▶Pr (*xMatrix*, *yMatrix*) ⟹ *矩阵*

返回 **(*x*,*y*)** 自变量对等效的 r 坐标值。

注意： 您可以通过在计算机键盘上键入 **R@>Pr(...)** 插入此函数。

在 Radian 角度模式下：

R▶Pr(3,2) 3.60555

R▶Pr([3 −4 2],[0 π/4 1.5])

[3 4.07638 5/2]

▶Rad

目录 >

Value1▶**Rad** ⟹ *值*

将自变量转换为弧度角度测量值。

注意： 您可以通过在计算机键盘上键入 **@>Rad** 插入此运算符。

在 Degree 角度模式下：

(1.5)▶Rad (0.02618)ʳ

在 Gradian 角度模式下：

(1.5)▶Rad (0.023562)ʳ

rand()

目录 >

rand() ⟹ *表达式*

rand(#*Trials*) ⟹ *数组*

rand() 返回一个 0 到 1 之间的随机值。

rand(#*Trials*) 返回一个数组　其元素为 *#Trials* 个介于 0 到 1 之间的随机值。

设置随机数种。

RandSeed 1147 *Done*

rand(2) {0.158206,0.717917}

randBin()

目录 >

randBin(*n*, *p*) ⇒ *表达式*

randBin(*n*, *p*, *#Trials*) ⇒ *数组*

randBin(*n*, *p*) 从指定的二项式分布中返回一个随机实数。

randBin(*n*, *p*, *#Trials*) 返回一个数组 其元素为 *#Trials* 个指定二项式分布的随机实数。

randBin(80,.5)	34.
randBin(80,.5,3)	{47.,41.,46.}

randInt()

目录 >

randInt(*lowBound*,*upBound*) ⇒ *表达式*

randInt(*lowBound*,*upBound* ,*#Trials*) ⇒ *数组*

randInt(*lowBound*, *upBound*) 返回一个介于指定范围 *lowBound* 和 *upBound* 之间的随机整数。

randInt(*lowBound*, *upBound* , *#Trials*) 返回一个数组 其元素为指定范围内的 *#Trials* 个随机整数。

randInt(3,10)	7.
randInt(3,10,4)	{8.,9.,4.,4.}

randMat()

目录 >

randMat(*numRows*, *numColumns*) ⇒ *矩阵*

返回指定维数的 元素值为介于 -9 到 9 之间的整数的矩阵。两个自变量必须都化简为整数。

RandSeed 1147	*Done*
randMat(3,3)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

注意：您每次按下 enter 时 该矩阵中的数值都会改变。

randNorm()

目录 >

randNorm(μ, σ) ⇒ *表达式*

randNorm(μ, σ, *#Trials*) ⇒ *数组*

randNorm(μ, σ) 从指定的正态分布中返回一个十进制小数。该值可以为任意实数 但必须尽可能地落在区间 [μ−3·σ, μ+3·σ] 内。

randNorm(μ, σ, *#Trials*) 返回一个数组 其元素为 *#Trials* 个指定正态分布的十进制小数。

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	-3.54356

randPoly()

目录 >

randPoly(*Var*, *Order*) ⇒ *表达式*

返回一个关于变量 *Var* 的指定*阶数*的多项式。系数为介于 -9 到 9 之间范围的随机整数。首项系数不得为零。

*阶数*必须介于 0 到 99 之间。

RandSeed 1147	*Done*
randPoly(*x*,5)	$-2\cdot x^5+3\cdot x^4-6\cdot x^3+4\cdot x-6$

randSamp()

目录 >

randSamp(*List*,*#Trials*[,*noRepl*]) ⇒ *数组*

返回一个数组 其元素为 *#Trials* 个取自 *List* 的随机样本可附样本替代值 (*noRepl*=0) 也可不附样本替代值 (*noRepl*=1)。默认附样本替换值。

Define *list3*={1,2,3,4,5}	*Done*
Define *list4*=randSamp(*list3*,6)	*Done*
list4	{5.,1.,3.,3.,4.,4.}

RandSeed

目录 >

RandSeed *Number*

如果 *Number* = 0　将种子设置为随机数生成器的出厂默认值。如果 *Number* ≠ 0　则可使用该函数来生成两个种子分别存储在变量 seed1 和 seed2 中。

RandSeed 1147	*Done*
rand()	0.158206

real()

目录 >

real(*Value1***)** ⟹ *值*

返回自变量的实数部分。

real(2+3·*i*)	2

real(*List1***)** ⟹ *数组*

返回数组中各元素的实数部分。

real({1+3·*i*,3,*i*})	{1,3,0}

real(*Matrix1***)** ⟹ *矩阵*

返回矩阵中各元素的实数部分。

$$\text{real}\begin{bmatrix}1+3\cdot i & 3\\ 2 & i\end{bmatrix} \qquad \begin{bmatrix}1 & 3\\ 2 & 0\end{bmatrix}$$

▶Rect

目录 >

Vector **▶Rect**

注意：您可以通过在计算机键盘上键入 **@>Rect** 插入此运算符。

以直角坐标 [x, y, z] 的形式显示 *Vector*。该向量必须为 2 维或 3 维　可以是行向量或列向量。

$$\begin{bmatrix}3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6}\end{bmatrix}\blacktriangleright\text{Rect}$$
$$\begin{bmatrix}1.06066 & 1.06066 & 2.59808\end{bmatrix}$$

注意：**▶Rect** 是一条显示格式指令　不是转换函数。您只能在输入行结尾处使用该函数　并且 *ans* 不会得到更新。

注意：另请参阅 **▶Polar**（ 第 74 页 ）。

complexValue **▶Rect**

以直角坐标形式 a+bi 显示 *complexValue*。该 *complexValue* 可为任意复数形式。不过　$re^{i\theta}$ 形式的输入会在 Degree 角度模式中产生错误。

注意：您必须对 (r∠θ) 形式的极坐标输入使用括号。

在 Radian 角度模式下：

$$\left(4\cdot e^{\frac{\pi}{3}}\right)\blacktriangleright\text{Rect} \qquad 11.3986$$

$$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Rect} \qquad 2.+3.4641\cdot i$$

在 Gradian 角度模式下：

$$((1\angle 100))\blacktriangleright\text{Rect} \qquad i$$

在 Degree 角度模式下：

$$((4\angle 60))\blacktriangleright\text{Rect} \qquad 2.+3.4641\cdot i$$

注意：要输入 ∠　可从 Catalog 的符号列表中选择。

ref()

目录 >

ref(*Matrix1***[, *Tol*]) ⟹** *矩阵*

返回 *Matrix1* 的行梯矩阵。

作为可选项　如果矩阵中任何元素的绝对值小于 *Tol*　则将该元素作为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时　使用此公差。否则　*Tol* 将被忽略。

- 如果您使用 ctrl enter 或将 **Auto or Approximate** 设定为 Approximate 模式　则运算会使用浮点算法完成。
- 如果 *Tol* 省略或未使用　则默认的公差计算方法为：
  5E⁻14 ·max(dim(*Matrix1*)) ·rowNorm(*Matrix1*)

Matrix1 中不得出现未定义的元素　否则可能会得到意想不到的结果。

例如　如果以下表达式中的 *a* 未定义　将显示一则警告消息　结果将显示如下：

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

出现警告是因为通用元素 1/*a* 在 *a*=0 时无效。

您可通过事先在 *a* 中存储一个值或使用约束运算符 ("|") 代换一个值来避免此项操作　如下例所示。

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \mid a=0 \quad \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

注意：另请参阅 **rref()**（ 第 88 页 ）。

$$\text{ref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

remain()

目录 >

remain(*Value1***, *Value2*) ⟹** *值*

remain(*List1***, *List2*) ⟹** *数组*

remain(*Matrix1***, *Matrix2*) ⟹** *矩阵*

根据如下恒等式所定义　返回第一个自变量关于第二个自变量的余数：

remain(x,0)　x

remain(x,y)　x−y·iPart(x/y)

作为结果　注意 **remain(**−x,y**)**　**−remain(**x,y**)**。结果要么为零　要么与第一个自变量有相同的正负号。

注意：另请参阅 **mod()**（ 第 63 页 ）。

$\text{remain}(7,0)$	7
$\text{remain}(7,3)$	1
$\text{remain}(-7,3)$	-1
$\text{remain}(7,-3)$	1
$\text{remain}(-7,-3)$	-1
$\text{remain}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,1\}$

$$\text{remain}\left(\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix},\begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Request *提示字符串*, *变量*[, *显示标记* [, *状态变量*]]
Request *提示字符串*, *函数*(*自变量 1*, ... *自变量 n*)
[, *显示标记* [, *状态变量*]]

编程命令：暂停程序　并显示包含消息 *promptString* 的对话框和填写用户响应的输入框。

当用户键入响应并单击 **OK** 后　输入框的内容将赋值给变量 *var*。

如果用户单击 **Cancel**　则程序将继续而不接受任何输入。如果 *var* 已定义　该程序会使用 *var* 以前的值。

可选的 *DispFlag* 自变量可以是任意表达式。

- 如果 *DispFlag* 已省略或计算为 **1**　则提示消息和用户响应将在 Calculator 历史记录中显示。
- 如果 *DispFlag* 计算为 **0**　则提示消息和响应不会在历史记录中显示。

可选的*状态变量*自变量为程序提供了一种方式　用于确定用户如何取消对话框。请注意　*状态变量*需要*显示标记*自变量。

- 如果用户单击**确定**或按下 **Enter** 或 **Ctrl+Enter**　则变量 *状态变量*设置为值 **1**。
- 否则　变量*状态变量*设置为值 **0**。

func() 自变量使程序能够将用户响应存储为函数定义。此句法的运算等同于用户执行以下命令：

Define *func*(*arg1*, *...argn*) = *user's response*

随后　程序就可以使用定义的函数 *func*()。*promptString* 应指导用户输入恰当的*用户响应*　完成函数定义。

注意：您可以在用户定义的程序内使用 **Request** 命令　但不能在函数内使用该命令。

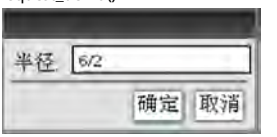
停止在无限循环内包含 **Request** 命令的程序：

- **Windows®：**按住 **F12** 键并反复按 **Enter** 键。
- **Macintosh®:** 按住 **F5** 键并反复按 **Enter** 键。
- **手持设备：**按住 on 键并反复按 enter 。

注意：另请参阅 **RequestStr**（ 第 85 页 ）。

定义程序：
```
Define request_demo()=Prgm
  Request " 半径：",r
  Disp " 区域 = ",pi*r^2
EndPrgm
```

运行该程序　然后键入响应：
```
request_demo()
```



选择 **OK** 后结果显示为：
半径：6/2
区域 = 28.2743

定义程序：
```
Define polynomial()=Prgm
  Request " 输入关于 x 的多项式：",p(x)
  Disp " 实数根为：",polyRoots(p(x),x)
EndPrgm
```

运行该程序　然后键入响应：
```
polynomial()
```

选择 **OK** 后结果显示为：
输入关于 x 的多项式：x^3+3x+1
实数根为：{-0.322185}

RequestStr

目录 >

RequestStr *promptString*, *var*[, *DispFlag*]

编程命令：除了用户的响应理解为字符串之外　其余完全按照 `Request` 命令的第一种句法进行运算。而 **Request** 命令将响应理解为表达式　除非用户将响应包含在引号 (“”) 内。

注意：您可以在用户定义的程序内使用 `RequestStr` 命令　但不能在函数内使用该命令。

停止在无限循环内包含 **RequestStr** 命令的程序：

- **Windows®：**按住 **F12** 键并反复按 **Enter** 键。
- **Macintosh®:** 按住 **F5** 键并反复按 **Enter** 键。
- **手持设备：**按住 [on] 键并反复按 [enter] 。

注意：另请参阅 **Request**（ 第 84 页 ）。

定义程序：

```
Define requestStr_demo()=Prgm
  RequestStr “Your name:”,name,0
  Disp “Response has “,dim(name),” characters.”
EndPrgm
```

运行该程序　然后键入响应：

```
requestStr_demo()
```

选择 **OK** 后的结果（ 注意　*DispFlag* 自变量为 **0** 时提示消息和响应不会在历史记录中显示 ）：

```
requestStr_demo()
响应有 5 个字符。
```

Return

目录 >

Return [*Expr*]

返回作为函数结果的 *Expr*。在 **Func**...**EndFunc** 块内使用。

注意：在 **Prgm**...**EndPrgm** 块内使用不带自变量的 **Return** 指令可退出程序。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 [↵]（ 而不是 [enter] ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

```
Define factoral(nn)=Func
                      Local answer,count
                      1→answer
                      For count,1,nn
                      answer·count→answer
                      EndFor
                      Return answer
                      EndFunc
                                        Done
factoral(3)                                6
```

right()

目录 >

right(*List1*[, *Num*]**)** ⟹ *数组*

返回 *List1* 中最右边的 *Num* 个元素。

如果您省略 *Num*　则会返回整个 *List1*。

right(*sourceString*[, *Num*]**)** ⟹ *字符串*

返回字符串 *sourceString* 中最右边的 *Num* 个字符。

如果您省略 *Num*　则会返回整个 *sourceString*。

right(*Comparison***)** ⟹ *表达式*

返回方程或不等式右侧的内容。

```
right({1,3,-2,4},3)                {3,-2,4}

right("Hello",2)                       "lo"
```

rk23()

目录 >

rk23(*表达式*, *变量*, *因变量*, { *变量0*, *变量最大值* }, *因变量0*, *变量步长*
[, *容差*]) ⟹ *矩阵*

rk23(*表达式方程组*, *变量*, *因变量数组*, { *变量0*, *变量最大值* }, *因变量数组0*, *变量步长* [, *容差*]) ⟹ *矩阵*

rk23(*表达式数组*, *变量*, *因变量数组*, { *变量0*, *变量最大值* }, *因变量数组0*, *变量步长* [, *容差*]) ⟹ *矩阵*

使用龙格 - 库塔方法求解方程组

$$\frac{d\,depVar}{d\,Var} = Expr(\text{变量}, \text{因变量})$$

其中 *depVar*(*变量0*)= *因变量0* 位于区间 [*变量0*, *变量最大值*] 中。返回一个矩阵　其第一行定义 *变量*输出值（ 通过 *变量步长*定义 ）。第二行定义相应的 *变量*值处第一个求解分量的值　依此类推。

*表达式*是定义常微分方程 (ODE) 的右侧内容。

*表达式方程组*是定义 ODE 方程组的右侧方程组（ 对应*因变量数组*中因变量的阶数 ）。

*表达式数组*是定义 ODE 方程组的右侧数组（ 对应*因变量数组*中因变量的阶数 ）。

*变量*是自变量。

*因变量数组*是因变量的数组。

{ *变量0*, *变量最大值* } 是两个元素的数组　告知函数从 *变量0* 到 *变量最大值*为一个整体。

因变量数组0 是因变量初始值的数组。

如果 *变量步长*计算为非零数字：sign(*变量步长*) = sign(*变量最大值* - *变量0*) 而解在 *变量0*+i* *变量步长*处返回（ 对于所有满足 *变量0*+i* *变量步长*位于 [*变量0*, *变量最大值*] 区间的 i=0,1,2,…　*变量最大值*处可能没有解值 ）。

如果 *变量步长*计算为零　则在 " 龙格 - 库塔 " *变量*值处返回解。

*容差*即误差容限（ 默认设为 0.001 ）。

微分方程：
y'=0.001*y*(100-y) 和 y(0)=10

$$\text{rk23}\left(0.001\cdot y\cdot\left(100-y\right),t,y,\left\{0,100\right\},10,1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9493 & 13.042 & 14.2 \end{bmatrix}$$

要查看完整结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

*容差*设置为 1.E•6 的同一方程

$$\text{rk23}\left(0.001\cdot y\cdot\left(100-y\right),t,y,\left\{0,100\right\},10,1,1.\text{E}{-6}\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix}$$

方程组：

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

其中 ***y1***(0)=2 并且 ***y2***(0)=5

$$\text{rk23}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases},t,\left\{y1,y2\right\},\left\{0,5\right\},\left\{2,5\right\},1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

root()

目录 >

root(*Value***)** ⟹　*根*

root(*Value1***, *Value2***)** ⟹　*根*

root(*Value*) 返回 *Value* 的平方根。

root(*Value1*, *Value2*)　返回 *Value1* 的 *Value2* 次方根。*Value1* 可以是实数或复数浮点常数　也可以是整数或复数有理数常数。

注意：另请参阅 **N 次方根模板**（ 第 1 页 ）。

$$\sqrt[3]{8} \qquad 2$$

$$\sqrt[3]{3} \qquad 1.44225$$

rotate()

rotate(*Integer1*[**,***#ofRotations*]**)** ⟹ *整数*

对一个二进制整数进行循环移位。您可以输入任意进位制的 *Integer1*　该整数将自动转换为带符号的 64 位二进制形式。如果 *Integer1* 的大小超出二进制整数的表示范围　可使用对称的模数运算将该值纳入合理的范围。更多信息　请参阅 ▶Base2（ 第 12 页 ）。

如果 *#ofRotations* 为正　则向左循环移位。如果 *#ofRotations* 为负　则向右循环移位。默认值为 −1（ 右移一位 ）。

例如　在向右循环移位的情况下：

各数位均右移。

0b00000000000000111101011000011010 1

最右的数位移动到最左。

结果为：

0b10000000000000011110101100001101 0

结果根据 Base 模式显示。

在 Bin 模式下：

rotate(0b11111111111111111111111111111111)	0b10000000000000000000000000000001▶
rotate(256,1)	0b1000000000

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

在 Hex 模式下：

rotate(0h78E)	0h3C7
rotate(0h78E,−2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

重要信息：要输入二进制或十六进制数值　始终使用 0b 或 0h 前缀（ 零　非字母 O ）。

rotate(*List1*[**,***#ofRotations*]**)** ⟹ *数组*

返回向右或向左循环移位 *#of Rotations* 个元素后的 *List1* 的副本。此运算不会更改 *List1*。

如果 *#ofRotations* 为正　则向左循环移位。如果 *#of Rotations* 为负　则向右循环移位。默认值为 −1（ 右移一个元素 ）。

在 Dec 模式下：

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(*String1*[**,***#ofRotations*]**)** ⟹ *字符串*

返回向右或向左循环移位 *#ofRotations* 个字符后的 *String1* 的副本。此运算不会更改 *String1*。

如果 *#ofRotations* 为正　则向左循环移位。如果 *#ofRotations* 为负　则向右循环移位。默认值为 −1（ 右移一个字符 ）。

rotate("abcd")	"dabc"
rotate("abcd",−2)	"cdab"
rotate("abcd",1)	"bcda"

round()

round(*Value1*[**,** *digits*]**)** ⟹ *值*

按四舍五入返回小数点后保留指定位数的自变量的值。

digits 必须为介于 0 到 12 之间的整数。如果不含 *digits*　则返回四舍五入后完整为 12 位有效数字的自变量的值。

注意：数字的显示模式可能对显示结果有影响。

round(1.234567,3)	1.235

round(*List1*[**,** *digits*]**)** ⟹ *数组*

返回一个数组　其组成为 List1 中的元素按四舍五入保留小数点后指定位数的值。

round(*Matrix1*[**,** *digits*]**)** ⟹ *矩阵*

返回一个矩阵　其组成为 Matrix1 中的元素按四舍五入保留小数点后指定位数的值。

round({π,√(2),ln(2)},4)	{3.1416,1.4142,0.6931}
round([ln(5) ln(3); π e^1],1)	[1.6 1.1; 3.1 2.7]

rowAdd()

目录 >

rowAdd(*Matrix1***, *rIndex1*, *rIndex2*)** ⇒ *矩阵*

返回 *Matrix1* 的副本　其中第 *rIndex2* 行被第 *rIndex1* 行与第 *rIndex2* 行的和替代。

$$\text{rowAdd}\left(\begin{bmatrix}3 & 4\\ -3 & -2\end{bmatrix},1,2\right) \qquad \begin{bmatrix}3 & 4\\ 0 & 2\end{bmatrix}$$

rowDim()

目录 >

rowDim(*Matrix***)** ⇒ *表达式*

返回 *Matrix* 的行数。

注意：另请参阅 **colDim()**（ 第 17 页 ）。

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

$$\text{rowDim}(m1) \qquad 3$$

rowNorm()

目录 >

rowNorm(*Matrix***)** ⇒ *表达式*

返回 *Matrix* 中各行元素的绝对值之和的最大值。

注意：所有矩阵元素必须化简为数值。另请参阅 **colNorm()**（ 第 17 页 ）。

$$\text{rowNorm}\left(\begin{bmatrix}-5 & 6 & -7\\ 3 & 4 & 9\\ 9 & -9 & -7\end{bmatrix}\right) \qquad 25$$

rowSwap()

目录 >

rowSwap(*Matrix1***, *rIndex1*, *rIndex2*)** ⇒ *矩阵*

返回 *Matrix1*　将其第 *rIndex1* 行与第 *rIndex2* 行进行交换。

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\rightarrow mat \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

$$\text{rowSwap}(mat,1,3) \qquad \begin{bmatrix}5 & 6\\ 3 & 4\\ 1 & 2\end{bmatrix}$$

rref()

目录 >

rref(*Matrix1*[**,** *Tol*]**)** ⇒ *矩阵*

返回 *Matrix1* 的递减行梯形式。

$$\text{rref}\left(\begin{bmatrix}-2 & -2 & 0 & -6\\ 1 & -1 & 9 & -9\\ -5 & 2 & 4 & -4\end{bmatrix}\right) \qquad \begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\ 0 & 1 & 0 & \frac{147}{71}\\ 0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$$

作为可选项　如果矩阵中任何元素的绝对值小于 *Tol*　则将该元素作为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时　使用此公差。否则　*Tol* 将被忽略。

- 如果您使用 ctrl enter 或将 **Auto or Approximate** 设定为 Approximate 模式　则运算会使用浮点算法完成。
- 如果 *Tol* 省略或未使用　则默认的公差计算方法为：5E⁻14 ·max(dim(*Matrix1*)) ·rowNorm(*Matrix1*)

注意：另请参阅 **ref()**（ 第 83 页 ）。

S

sec()

trig 键

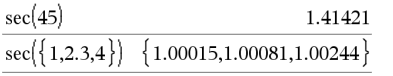
sec(*Value1*) ⇒ *值*

sec(*List1*) ⇒ *数组*

返回 *Value1* 的正割值　或返回一个数组　其元素为 *List1* 中所对应元素的正割值。

注意：自变量可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。您可以使用 ° G 或 r 临时更改角度模式。

在 Degree 角度模式下：



sec^{-1}()

trig 键

sec^{-1}(*Value1*) ⇒ *值*

sec^{-1}(*List1*) ⇒ *数组*

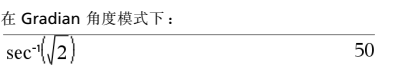
返回正割值为 *Value1* 的角度　或返回一个数组　其元素为 *List1* 所对应元素的反正割值。

注意：返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

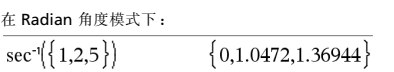
注意：您可以通过在计算机键盘上键入 **arcsec(...)** 插入此函数。

在 Degree 角度模式下：

在 Gradian 角度模式下：



在 Radian 角度模式下：



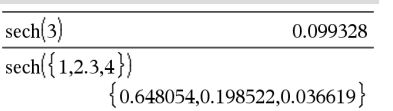
sech()

目录 >

sech(*Value1*) ⇒ *值*

sech(*List1*) ⇒ *数组*

返回 *Value1* 的双曲正割值　或返回一个数组　其元素为 *List1* 所对应元素的双曲正割值。



sech^{-1}()

目录 >

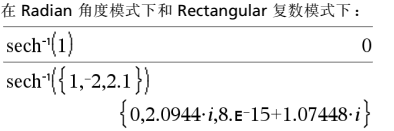
sech^{-1}(*Value1*) ⇒ *值*

sech^{-1} (*List1*) ⇒ *数组*

返回 *Value1* 的反双曲正割值或返回一个数组　其元素为 *List1* 所对应元素的反双曲正割值。

注意：您可以通过在计算机键盘上键入 **arcsech(...)** 插入此函数。

在 Radian 角度模式下和 Rectangular 复数模式下：



seq()

目录 >

seq(*Expr***, *Var*, *Low*, *High*[, *Step*]) ⟹** *数组*

从 *下限* 到 *上限* 以 *步长* 为增量增加 *变量* 计算 *表达式* 并返回结果数组。*变量* 的初始内容在 **seq()** 执行完毕后保持不变。

步长 的默认值 = 1。

$\text{seq}\left(n^2,n,1,6\right)$ $\{1,4,9,16,25,36\}$

$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$ $\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ $\frac{1968329}{1270080}$

按 **Ctrl+Enter** ctrl enter (Macintosh®: ⌘+Enter) 进行计算：

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ 1.54977

seqGen()

目录 >

seqGen(*表达式*, *变量*, *因变量*, { *变量0*, *变量最大值* }[, *初始项数组*

[, *变量步长* [, *上限值*]]]) ⟹ *数组*

生成序列 *depVar*(*变量*)= *表达式* 的项数组如下：从 *变量0* 到 *变量最大值* 以 *变量步长* 为增量增加自变量 *变量* 使用 *表达式* 公式和 *初始项数组* 计算对应 *变量* 值的 *depVar*(*变量*) 然后返回结果数组。

seqGen(*表达式数组或表达式方程组*, *变量*, *因变量数组*, { *变量0*, *变量最大值* }

[, *初始项矩阵* [, *变量步长* [, *上限值*]]]) ⟹ *矩阵*

生成序列 *ListOfDepVars*(*变量*)= *表达式数组或表达式方程组* 的方程组（或数组）项矩阵如下：从 *变量0* 到 *变量最大值* 以 *变量步长* 为增量增加自变量 *变量* 使用 *表达式数组或表达式方程组* 公式和 *初始项矩阵* 计算对应 *变量* 值的 *ListOfDepVars*(*变量*) 然后返回结果矩阵。

变量 的初始内容在 **seqGen()** 执行完毕后保持不变。

变量步长 的默认值 = **1**。

生成序列 $u(n) = u(n\text{-}1)^2/2$ 的前 5 项 其中 $u(1)=2$ 并且 *变量步长* =**1**。

$\text{seqGen}\left(\frac{\left(u\left(n-1\right)\right)^2}{n},n,u,\{1,5\},\{2\}\right)$

$\left\{2,2,\frac{4}{3},\frac{4}{9},\frac{16}{405}\right\}$

变量 0=2 的示例：

$\text{seqGen}\left(\frac{u\left(n-1\right)+1}{n},n,u,\{2,5\},\{3\}\right)$

$\left\{3,\frac{4}{3},\frac{7}{12},\frac{19}{60}\right\}$

两个序列的方程组：

$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2\left(n-1\right)}{2}+u1\left(n-1\right)\right\},n,\{u1,u2\},\{1,5\},\left[\begin{matrix}\_\\2\end{matrix}\right]\right)$

$\begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24}\end{bmatrix}$

注意：上述初始项矩阵中的空值 (_) 用于表示 u1(n) 的初始项使用显式序列公式 u1(n)=1/n 计算。

seqn()

目录 >

seqn(*Expr***(***u***, *n* [, *初始项数组* [, *n 最大值* [, *上限值*]]]) ⟹ *数组***

生成序列 *u*(*n*)=*Expr*(*u*, *n*) 的项数组如下：从 1 到 *n 最大值* 以 1 为增量增加 *n*。使用 *Expr*(*u*, *n*) 公式和*初始项数组*计算对应值 *n* 的 *u*(*n*)，然后返回结果数组。

seqn(*Expr***(***n* [, *n 最大值* [, *上限值*]]) ⟹ *数组***

生成非递归序列 *u*(*n*)=*Expr*(*n*) 的项数组如下：从 1 到 *n 最大值*以 1 为增量增加 *n*。使用 *Expr*(*u*, n) 公式计算对应值 *n* 的 *u*(*n*)，然后返回结果数组。

如果缺少 *n 最大值*，则 *n 最大值*设置为 2500

如果 *n 最大值*=0，则 *n 最大值*设置为 2500

注意：seqn() 通过 *n0*=**1** 和 *n 步长* =**1** 调用 **seqGen( )**

生成序列 *u*(*n*) = *u*(*n*-1)/2 的前 6 项，其中 *u*(1)=**2**。

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right) \qquad \left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

setMode()

目录 >

setMode(*modeNameInteger***, *settingInteger*)**
⟹ *整数*

setMode(*list***) ⟹** *整数数组*

仅在函数或程序内有效。

setMode(*modeNameInteger***, *settingInteger*)** 可临时将模式 *modeNameInteger* 设置为新设置 *settingInteger*，并返回一个对应于该模式原始设置的整数。此更改仅可在程序 / 函数的执行过程中进行。

modeNameInteger 指定您要设置的模式的名称。它必须为下表中的模式整数之一。

settingInteger 指定模式的新设置名称。它必须为下列特定模式设置整数之一。

setMode(*list***)** 可以更改多个设置。*list* 包含模式整数和设置整数对。**setMode(***list***)** 返回一个类似数组，其中整数对表示原始模式和设置。

如果您使用 **getMode(0) ⟶** *var* 保存所有模式设置，则可以使用 **setMode(***var***)** 还原这些设置，直到函数或程序退出。另请参阅 **getMode()**（第 42 页）。

注意：此时将传递当前模式设置以调用子例程。如果任何子例程更改了模式设置，则控制返回到调用例程时模式更改将丢失。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中，您可以通过在每行结尾处按 ↵（而不是 enter）输入多行定义。在计算机键盘上，按住 **Alt** 然后按 **Enter**。

使用 Display Digits 的默认设置显示 π 的近似值，然后使用 Fix2 的设置显示 π。检查程序执行后默认值是否还原。

```
Define prog1()=Prgm                    Done
                Disp π
                setMode(1,16)
                Disp π
                EndPrgm
prog1()
                                       3.14159
                                       3.14
                                       Done
```

模式 名称	模式 整数	设置整数
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

模式 名称	模式 整数	设置整数
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
Vector Format	**6**	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base	**7**	**1**=Decimal, **2**=Hex, **3**=Binary

shift()

目录 >

shift(*Integer1*[**,***#ofShifts*]**)** ⟹ *整数*

对一个二进制整数进行平移。您可以输入任意进位制的 *Integer1*。该整数将自动转换为带符号的 64 位二进制形式。如果 *Integer1* 的大小超出二进制整数的表示范围，可使用对称的模数运算将该值纳入合理的范围。更多信息，请参阅 ▶Base2（第 12 页）。

如果 *#ofShifts* 为正，将向左平移。如果 *#ofShifts* 为负，将向右平移。默认值为 −1（向右平移一位）。

向右平移时，去掉最右边的数位，同时在最左边的数位上插入 0 或 1。向左平移时，去掉最左边的数位，同时在最右边的数位上插入 0。

例如，在向右平移时：

各数位向右平移。

0b00000000000000011110101011000011010

如果最左侧的数位为 0 则插入 0
如果最左侧的数位为 1 则插入 1。

结果为：

0b00000000000000001111010110000011010

结果根据 Base 模式显示。首尾的零不显示。

在 Bin 模式下：

shift(0b1111010110000110101)	0b111101011000011010
shift(256,1)	0b1000000000

在 Hex 模式下：

shift(0h78E)	0h3C7
shift(0h78E,−2)	0h1E3
shift(0h78E,2)	0h1E38

重要信息：要输入二进制或十六进制数值，始终使用 0b 或 0h 前缀（零，非字母 O）。

shift(*List1* [**,***#ofShifts*]**)** ⟹ *数组*

返回向右或向左平移 *#ofShifts* 个元素后的 *List1* 的副本。此运算不会更改 *List1*。

如果 *#ofShifts* 为正，将向左平移。如果 *#ofShifts* 为负，将向右平移。默认值为 −1（向右平移一个元素）。

通过平移引入到*数组*首位或末位的元素被设置为符号 "undef"。

在 Dec 模式下：

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(*String1* [**,***#ofShifts*]**)** ⟹ *字符串*

返回向右或向左平移 *#ofShifts* 个字符后的 *String1* 的副本。此运算不会更改 *String1*。

如果 *#ofShifts* 为正，将向左平移。如果 *#ofShifts* 为负，将向右平移。默认值为 −1（向右平移一个字符）。

通过平移引入到*字符串*首位或末位的元素被设置为空格。

shift("abcd")	" abc"
shift("abcd",−2)	" ab"
shift("abcd",1)	"bcd "

sign()

目录 >

sign(*Value1***)** ⟹ *值*

sign(*List1***)** ⟹ *数组*

sign(*Matrix1***)** ⟹ *矩阵*

对于实数和复数 *Value1*　*Value1*≠ 0 时返回 *Value1* / **abs(***Value1***)**。

如果 *Value1* 为正则返回 1。

如果 *Value1* 为负则返回 −1。

如果复数格式模式为 Real，则 sign(0) 返回 ±1　否则返回自身的值。

sign(0) 表示复数域中的单位圆。

对于数组或矩阵　返回所有元素的符号。

$\text{sign}(-3.2)$ → -1

$\text{sign}(\{2,3,4,-5\})$ → $\{1,1,1,-1\}$

如果复数格式模式为 Real：

$\text{sign}([-3 \quad 0 \quad 3])$ → $[-1 \quad \text{undef} \quad 1]$

simult()

目录 >

simult(*coeffMatrix***,** *constVector***[,** *Tol***])** ⟹ *矩阵*

返回包含线性方程组的解的列向量。

注意：另请参阅 **linSolve()**（ 第 54 页 ）。

coeffMatrix 必须为包含方程系数的方阵。

constVector 必须与 *coeffMatrix* 有相同的行数（ 相同的维数 ）且包含常数项。

作为可选项　如果矩阵中任何元素的绝对值小于 *Tol*　则将该元素作为零值处理。仅当矩阵有浮点输入项且不含任何未赋值的符号变量时　使用此公差。否则　*Tol* 将被忽略。

- 如果您将 **Auto or Approximate** 模式设置为 Approximate　运算将使用浮点计算完成。
- 如果 *Tol* 省略或未使用　则默认的公差计算方法为：
  5E−14 ·max(dim(*coeffMatrix*)) ·rowNorm(*coeffMatrix*)

求 x 和 y 的解：
x + 2y = 1
3x + 4y = −1

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\begin{bmatrix}1\\ -1\end{bmatrix}\right)$ → $\begin{bmatrix}-3\\ 2\end{bmatrix}$

解为 x=−3 且 y=2。

求解：
ax + by = 1
cx + dy = 2

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \rightarrow matx1$ → $\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$

$\text{simult}\left(matx1,\begin{bmatrix}1\\ 2\end{bmatrix}\right)$ → $\begin{bmatrix}0\\ \frac{1}{2}\end{bmatrix}$

simult(*coeffMatrix***,** *constMatrix***[,** *Tol***])** ⟹ *矩阵*

求解多个系数相同但常数项不同的线性方程组。

constMatrix 的各列必须包含方程组的常数项。结果矩阵的各列包含相应方程组的解。

求解：
x + 2y = 1
3x + 4y = −1

x + 2y = 2
3x + 4y = −3

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\begin{bmatrix}1 & 2\\ -1 & -3\end{bmatrix}\right)$ → $\begin{bmatrix}-3 & -7\\ 2 & \frac{9}{2}\end{bmatrix}$

对于第一个方程组　x=−3 且 y=2。对于第二个方程组　x=−7 且 y=9/2。

sin()

trig 键

sin(*Value1*) ⇒ *值*

sin(*List1*) ⇒ *数组*

sin(*Value1*) 返回自变量的正弦值。

sin(*List1*) 返回一个数组　其元素为 *List1* 中所有元素的正弦值。

注意：自变量可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。您可以使用 ° G 或 r 临时更改角度模式。

在 Degree 角度模式下：

$$\sin\left(\left(\frac{\pi}{4}\right)^r\right) \qquad 0.707107$$

$$\sin(45) \qquad 0.707107$$

$$\sin(\{0,60,90\}) \qquad \{0.,0.866025,1.\}$$

在 Gradian 角度模式下：

$$\sin(50) \qquad 0.707107$$

在 Radian 角度模式下：

$$\sin\left(\frac{\pi}{4}\right) \qquad 0.707107$$

$$\sin(45°) \qquad 0.707107$$

sin(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵正弦值。此运算不同于计算每个元素的正弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Radian 角度模式下：

$$\sin\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

$\sin^{-1}()$

trig 键

$\sin^{-1}$(*Value1*) ⇒ *值*

$\sin^{-1}$(*List1*) ⇒ *数组*

$\sin^{-1}$(*Value1*) 返回一个角度值　其正弦值为 *Value1*。

$\sin^{-1}$(*List1*) 返回一个数组　其元素为 *List1* 中所对应元素的反正弦值。

注意：返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

注意：您可以通过在计算机键盘上键入 **arcsin(...)** 插入此函数。

$\sin^{-1}$(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵反正弦值。此运算不同于计算每个元素的反正弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Degree 角度模式下：

$$\sin^{-1}(1) \qquad 90$$

在 Gradian 角度模式下：

$$\sin^{-1}(1) \qquad 100$$

在 Radian 角度模式下：

$$\sin^{-1}(\{0,0.2,0.5\}) \qquad \{0,0.201358,0.523599\}$$

在 Radian 角度模式下和 Rectangular 复数格式模式下：

$$\sin^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}-0.164058-0.064606\cdot i & 1.49086-2.1051\\ 0.725533-1.51594\cdot i & 0.947305-0.7783\\ 2.08316-2.63205\cdot i & -1.79018+1.2718\end{bmatrix}$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

sinh()

目录 >

sinh(*Numver1***)** ⇒ *值*

sinh(*List1***)** ⇒ *数组*

sinh (*Value1***)** 返回自变量的双曲正弦值。

sinh (*List1***)** 返回一个数组　其元素为 *List1* 中所对应元素的双曲正弦值。

sinh(*squareMatrix1***)** ⇒ *方阵*

返回 *squareMatrix1* 的矩阵双曲正弦值。此运算不同于计算每个元素的双曲正弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

$\sinh(1.2)$ → 1.50946

$\sinh(\{0,1.2,3.\})$ → $\{0,1.50946,10.0179\}$

在 Radian 角度模式下：

$$\sinh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

$\sinh^{-1}()$

目录 >

sinh^{-1}(*Value1***)** ⇒ *值*

sinh^{-1}(*List1***)** ⇒ *数组*

sinh^{-1}(*Value1***)** 返回自变量的反双曲正弦值。

sinh^{-1}(*List1***)** 返回一个数组　其元素为 *List1* 中所对应元素的反双曲正弦值。

注意：您可以通过在计算机键盘上键入 **arcsinh(...)** 插入此函数。

sinh^{-1}(*squareMatrix1***)** ⇒ *方阵*

返回 *squareMatrix1* 的矩阵反双曲正弦值。此运算不同于计算每个元素的反双曲正弦值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,1.81845\}$

在 Radian 角度模式下：

$$\sinh^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

SinReg

目录 >

SinReg *X*, *Y* [, [*Iterations*] ,[*Period*] [, *Category*, *Include*]]

计算基于数组 *X* 和 *Y* 的正弦回归。结果摘要存储在 *stat.results* 变量中。（请参阅第 98 页。）

除 *Include* 外，所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Iterations 指定了求解的最大尝试次数（1 到 16）。如果省略，则尝试 8 次。通常，该值越大，则结果越精确，但执行时间也越长，反之亦然。

Period 指定了预计周期。如果省略，则 *X* 中各元素之间的差值应相等并且按顺序排列。如果指定了 *Period*，则 x 各元素之间的差值可不相等。

Category 是由相应 *X* 和 *Y* 数据的数值或字符串类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

不论角度模式设置如何，**SinReg** 的输出始终为弧度。

有关数组中空元素结果的信息，请参阅 "空（空值）元素"（第 134 页）。

输出变量	说明
stat.RegEqn	回归方程：a•sin(bx+c)+d
stat.a　stat.b stat.c　stat.d	回归系数
stat.Resid	回归残差
stat.XReg	被修改后的数组 *X List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* *限制的回归中*
stat.YReg	被修改后的数组 *Y List* 中的数据点数组，实际用在基于 *Freq*、*Category List* 和 *Include Categories* *限制的回归中*
stat.FreqReg	由对应于 *stat.XReg* 和 *stat.YReg* *的频率所组成的数组*

SortA

目录 >

SortA *List1*[, *List2*] [, *List3*] ...
SortA *Vector1*[, *Vector2*] [, *Vector3*] ...

将第一自变量的元素按升序排列。

如果您加入了其他自变量，那么这些自变量的元素也将跟随第一自变量重新排列，以保持与第一自变量元素的相对位置不变。

所有自变量必须为数组或向量。所有自变量必须维数相等。

第一个自变量中的空（空值）元素将移至底部。有关空元素的更多信息，请参阅第 134 页。

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA *list1*	*Done*
list1	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA *list2*,*list1*	*Done*
list2	$\{1,2,3,4\}$
list1	$\{4,3,2,1\}$

SortD

目录 >

SortD *List1*[, *List2*] [, *List3*] ...
SortD *Vector1*[,*Vector2*] [,*Vector3*] ...

与 **SortA** 类似　只是 **SortD** 以降序排列元素。

第一个自变量中的空（ 空值 ）元素将移至底部。有关空元素的更多信息　请参阅第 134 页。

$\{2,1,4,3\} \rightarrow list1$ → $\{2,1,4,3\}$

$\{1,2,3,4\} \rightarrow list2$ → $\{1,2,3,4\}$

SortD *list1*,*list2* → *Done*

list1 → $\{4,3,2,1\}$

list2 → $\{3,4,1,2\}$

▶Sphere

目录 >

Vector **▶Sphere**

注意：您可以通过在计算机键盘上键入 **@>Sphere** 插入此运算符。

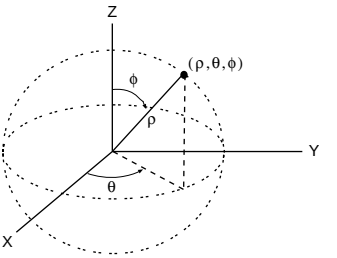
以球坐标形式 [ρ ∠θ ∠ϕ] 显示行向量或列向量。

Vector 必须为 3 维　可以是行向量或列向量。

注意：**▶Sphere** 是一条显示格式指令　不是转换函数。您只能在输入行结尾处使用。

$[1 \quad 2 \quad 3] \blacktriangleright \text{Sphere}$
$[3.74166 \quad \angle 1.10715 \quad \angle 0.640522]$

$\left[2 \quad \angle\frac{\pi}{4} \quad 3\right] \blacktriangleright \text{Sphere}$
$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$



sqrt()

目录 >

sqrt(*Value1***)** ⇒ *值*
sqrt(*List1***)** ⇒ *数组*

返回自变量的平方根。

对于数组　返回 *List1* 中所有元素的平方根。

注意：另请参阅**平方根模板**（ 第 1 页 ）。

$\sqrt{4}$ → 2

$\sqrt{\{9,2,4\}}$ → $\{3,1.41421,2\}$

stat.results

显示统计计算的结果。

结果以名值对集合的形式显示。显示的特定名称取决于最近计算的统计函数或命令。

您可以复制名称或值并将其粘贴到其他位置。

注意：用于定义变量的名称避免与统计分析中的变量名称相同。某些情况下 可能会出现错误。用于统计分析的变量名称将在下表中列出。

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*

"Title"	"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.AdjR2
stat.b
stat.b0
stat.b1
stat.b2
stat.b3
stat.b4
stat.b5
stat.b6
stat.b7
stat.b8
stat.b9
stat.b10
stat.bList
stat.χ2
stat.c
stat.CLower
stat.CLowerList
stat.CompList
stat.CompMatrix
stat.CookDist
stat.CUpper
stat.CUpperList
stat.d
stat.dfDenom
stat.dfBlock
stat.dfCol
stat.dfError
stat.dfInteract
stat.dfReg
stat.dfNumer
stat.dfRow
stat.DW
stat.e
stat.ExpMatrix
stat.F
stat.FBlock
stat.Fcol
stat.FInteract
stat.FreqReg
stat.Frow
stat.Leverage
stat.LowerPred
stat.LowerVal
stat.m
stat.MaxX
stat.MaxY
stat.ME
stat.MedianX
stat.MedianY
stat.MEPred
stat.MinX
stat.MinY
stat.MS
stat.MSBlock
stat.MSCol
stat.MSError
stat.MSInteract
stat.MSReg
stat.MSRow
stat.n
stat.$\hat{p}$
stat.$\hat{p}_1$
stat.$\hat{p}_2$
stat.$\hat{p}$Diff
stat.PList
stat.PVal
stat.PValBlock
stat.PValCol
stat.PValInteract
stat.PValRow
stat.Q1X
stat.Q1Y
stat.Q3X
stat.Q3Y
stat.r
stat.r2
stat.RegEqn
stat.Resid
stat.ResidTrans
stat.σ_x
stat.σ_y
stat.σ_{x1}
stat.σ_{x2}
stat.Σ_x
stat.Σ_{x2}
stat.Σ_{xy}
stat.Σ_y
stat.Σ_{y2}
stat.s
stat.SE
stat.SEList
stat.SEPred
stat.sResid
stat.SEslope
stat.sp
stat.SS
stat.SSBlock
stat.SSCol
stat.SSX
stat.SSY
stat.SSError
stat.SSInteract
stat.SSReg
stat.SSRow
stat.tList
stat.UpperPred
stat.UpperVal
stat.$\bar{X}$
stat.$\bar{X}_1$
stat.$\bar{X}_2$
stat.$\bar{X}$Diff
stat.$\bar{X}$List
stat.XReg
stat.XVal
stat.XValList

注意：每次 Lists & Spreadsheet 应用程序计算统计结果时 都会将 **.”** 组变量复制到 **“stat#.”** 组 其中 **#** 是自动增加的数值。这样可让您在进行多个计算时保留原来的结果。

stat.values

目录 >

stat.values

显示一个矩阵　其元素为最近计算的统计函数或命令的计算值。

与 **stat.results** 不同的是　**stat.values** 会省略与这些值相关的名称。

您可以复制值并将其粘贴到其他位置。

请参阅 **stat.results** 示例。

stDevPop()

目录 >

stDevPop(*List***[,** *freqList***])** ⇒ *表达式*

返回 *List* 中元素的总体标准差。

freqList 中的元素为 *List* 中各对应元素出现的次数。

注意：*List* 必须包含至少两个元素。空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

stDevPop(*Matrix1***[,** *freqMatrix***])** ⇒ *矩阵*

返回 *Matrix1* 中各列的总体标准差组成的行向量。

freqMatrix 中的元素为 *Matrix1* 中各对应元素出现的次数。

注意：*Matrix1* 必须至少有两行。空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

在 Radian 角度模式和自动模式下：

$\text{stDevPop}(\{1,2,5,-6,3,-2\})$ 3.59398

$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\})$ 4.11107

$$\text{stDevPop}\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \quad \begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix},\begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

stDevSamp()

目录 >

stDevSamp(*List***[,** *freqList***])** ⇒ *表达式*

返回 *List* 中元素的样本标准差。

freqList 中的元素为 *List* 中各对应元素出现的次数。

注意：*List* 必须包含至少两个元素。空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

stDevSamp(*Matrix1***[,** *freqMatrix***])** ⇒ *矩阵*

返回 *Matrix1* 中各列的样本标准差的行向量。

freqMatrix 中的元素为 *Matrix1* 中各对应元素出现的次数。

注意：*Matrix1* 必须至少有两行。空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

$\text{stDevSamp}(\{1,2,5,-6,3,-2\})$ 3.937

$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\})$ 4.33345

$$\text{stDevPop}\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \quad \begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix},\begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

Stop

目录 >

Stop

编程命令：终止程序。

Stop 不能在函数中使用。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

$i:=0$	0
Define $prog1()$=Prgm For i,1,10,1 If i=5 Stop EndFor EndPrgm	*Done*
$prog1()$	*Done*
i	5

Store

请参阅 → **(store)**（ 第 132 页 ）。

string()

目录 >

string(*Expr***)** ⇒ *字符串*

简化 *Expr* 并以字符串形式返回结果。

string(1.2345)	"1.2345"
string(1+2)	"3"

subMat()

目录 >

subMat(*Matrix1***[, *startRow*] [, *startCol*] [, *endRow*] [, *endCol*])**
⇒ *矩阵*

返回 *Matrix1* 的指定子矩阵。

默认值：*startRow*=1　*startCol*=1　*endRow*=last row *endCol*=last column。

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$
subMat($m1$,2,1,3,2)	$\begin{bmatrix}4&5\\7&8\end{bmatrix}$
subMat($m1$,2,2)	$\begin{bmatrix}5&6\\8&9\end{bmatrix}$

Sum (Sigma)

请参阅 Σ()（ 第 127 页 ）。

sum()

目录 >

sum(*List*[,　*Start*[,　*End*]]) ⇒ *表达式*

返回 *List* 所有元素的和。

Start 和 *End* 为可选项。它们指定了元素的范围。

任何空值自变量都会生成空值结果。*List* 中的空（ 空值 ）元素将被忽略。有关空元素的更多信息　请参阅第 134 页。

sum({1,2,3,4,5})	15
sum({a,2·a,3·a})	"Error: Variable is not defined"
sum(seq(n,n,1,10))	55
sum({1,3,5,7,9},3)	21

sum()

目录 >

sum(*Matrix1*[, *Start*[, *End*]]) ⟹ *矩阵*

返回由 *Matrix1* 中各列的元素和组成的行向量。

Start 和 *End* 为可选项。它们指定了行的范围。

任何空值自变量都会生成空值结果。*Matrix1* 中的空（空值）元素将被忽略。有关空元素的更多信息，请参阅第 134 页。

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$ $\quad\begin{bmatrix}5&7&9\end{bmatrix}$

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ $\quad\begin{bmatrix}12&15&18\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$ $\quad\begin{bmatrix}11&13&15\end{bmatrix}$

sumIf()

目录 >

sumIf(*List*,*Criteria*[, *SumList*]) ⟹ *值*

返回 *List* 中符合指定 *Criteria* 的所有元素的和。作为可选项，您可以指定候选数组 *sumList* 提供要累加的元素。

List 可以是表达式、数组或矩阵。*SumList*（如指定）必须与 *List* 维数相同。

Criteria 可以是：

- 值、表达式或字符串。例如，如指定标准为 **34**，则仅累加 *List* 中化简值等于 34 的元素。
- 布尔表达式，使用符号 **?** 作为各元素的占位符。例如，如指定标准为 **?<10**，则仅累加 *List* 中小于 10 的元素。

List 中符合 *Criteria* 的元素将累加到和中。如果您添加了 *sumList*，则会累加 *sumList* 中的相应元素。

在 Lists & Spreadsheet 应用程序中，您可以使用单元格范围代替 *List* 和 *sumList*。

空（空值）元素将被忽略。有关空元素的更多信息，请参阅第 134 页。

注意：另请参阅 **countIf()**（第 23 页）。

$\text{sumIf}\left(\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5\right)$

12.859874482

$\text{sumIf}\left(\{1,2,3,4\},2<?<5,\{10,20,30,40\}\right)$

70

sumSeq()

请参阅 Σ()（第 127 页）。

system()

目录 >

system(*Value1* [, *Value2* [, *Value3* [, ...]]])

以数组形式返回一个方程组。您也可以使用模板创建方程组。

T

T（转置）

目录 >

Matrix1^{T} ⟹ *矩阵*

返回 *Matrix1* 的复共轭转置矩阵。

注意：您可以通过在计算机键盘上键入 **@t** 插入此运算符。

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}^{T}$ $\quad\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}$

tan()

trig 键

tan(*Value1***)** ⇒ *值*

tan(*List1***)** ⇒ *数组*

tan(*Value1***)** 返回自变量的正切值。

tan(*List1***)** 返回一个数组　其元素为 *List1* 中所有元素的正切值。

注意：自变量可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。您可以使用 ° G 或 r 临时更改角度模式设置。

在 Degree 角度模式下：

$\tan\left(\left(\frac{\pi}{4}\right)^{r}\right)$ 1.

$\tan(45)$ 1.

$\tan(\{0,60,90\})$ $\{0.,1.73205,undef\}$

在 Gradian 角度模式下：

$\tan\left(\left(\frac{\pi}{4}\right)^{r}\right)$ 1.

$\tan(50)$ 1.

$\tan(\{0,50,100\})$ $\{0.,1.,undef\}$

在 Radian 角度模式下：

$\tan\left(\frac{\pi}{4}\right)$ 1.

$\tan(45°)$ 1.

$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right)$ $\{0.,1.73205,0.,1.\}$

tan(*squareMatrix1***)** ⇒ *方阵*

返回 *squareMatrix1* 的矩阵正切。此运算不同于计算每个元素的正切值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Radian 角度模式下：

$$\tan\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

$\tan^{-1}()$

trig 键

$\tan^{-1}$(*Value1***)** ⇒ *值*

$\tan^{-1}$(*List1***)** ⇒ *数组*

$\tan^{-1}$(*Value1***)** 返回一个角度值　其正切值为 *Value1*。

$\tan^{-1}$(*List1***)** 返回一个数组　其元素为 *List1* 中所对应元素的反正切值。

注意：返回的结果可以是度　弧度或百分度形式　具体取决于当前的角度模式设置。

注意：您可以通过在计算机键盘上键入 **arctan(...)** 插入此函数。

在 Degree 角度模式下：

$\tan^{-1}(1)$ 45

在 Gradian 角度模式下：

$\tan^{-1}(1)$ 50

在 Radian 角度模式下：

$\tan^{-1}(\{0,0.2,0.5\})$ $\{0,0.197396,0.463648\}$

tan^{-1}()

tan^{-1}(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵反正切值　此运算不同于计算每个元素的反正切值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Radian 角度模式下：

$$\tan^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

tanh()

tanh(*Value1*) ⇒ *值*

tanh(*List1*) ⇒ *数组*

tanh(*Value1*) 返回自变量的正切值。

tanh(*List1*) 返回一个数组　其元素为 *List1* 中所对应元素的双曲正切值。

tanh(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵双曲正切值　此运算不同于计算每个元素的双曲正切值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

$$\tanh(1.2) \qquad 0.833655$$

$$\tanh(\{0,1\}) \qquad \{0.,0.761594\}$$

在 Radian 角度模式下：

$$\tanh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

tanh^{-1}()

tanh^{-1}(*Value1*) ⇒ *值*

tanh^{-1}(*List1*) ⇒ *数组*

tanh^{-1}(*Value1*) 返回自变量的反双曲正切值。

tanh^{-1}(*List1*) 返回一个数组　其元素为 *List1* 中所对应元素的反双曲正切值。

注意：您可以通过在计算机键盘上键入 **arctanh(...)** 插入此函数。

tanh^{-1}(*squareMatrix1*) ⇒ *方阵*

返回 *squareMatrix1* 的矩阵反双曲正切值　此运算不同于计算每个元素的反双曲正切值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

在 Rectangular 复数格式下：

$$\tanh^{-1}(0) \qquad 0.$$

$$\tanh^{-1}(\{1,2.1,3\})$$

$$\{\text{undef},0.518046-1.5708\cdot i,0.346574-1.570\blacktriangleright$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

在 Radian 角度模式和 Rectangular 复数格式下：

$$\tanh^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908\\ -0.087596-0.725533\cdot i & 0.479679-0.9473\blacktriangleright\\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

tCdf()

目录 >

tCdf(*lowBound*,*upBound*,*df***)** ⇒ 如果 *lowBound* 和 *upBound* 是数值，则结果为 *数值*，如果 *lowBound* 和 *upBound* 是数组，则结果为 *数组*

计算在 *lowBound* 和 *upBound* 之间 指定自由度为 *df* 的学生 *t* 分布概率。

对于 P(X ≤ *upBound*) 设置 *lowBound* = **−9E999**。

Text

目录 >

Text *promptString* [, *DispFlag*]

编程命令：暂停程序并在对话框中显示字符串 *promptString*。

用户选择 **OK** 后 程序将继续执行。选择 **Cancel** 将停止程序。

可选的 *flag* 自变量可以是任意表达式。

- 如果 *DispFlag* 已省略或计算为 **1** 则文本消息将添加到 Calculator 历史记录中。
- 如果 *DispFlag* 计算为 **0** 则文本消息不会添加到历史记录。

如果程序需要用户输入响应 请参阅 **Request**（ 第 84 页 ）或 **RequestStr**（ 第 85 页 ）。

注意：此命令可以在用户定义的程序内使用 但不能在函数内使用。

定义一个程序 暂停可在对话框中显示五个随机数值 每次显示一个。

在 Prgm...EndPrgm 模板内 通过按 ↵（ 而不是 enter)完成每行的输入。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

```
Define text_demo()=Prgm
 For i,1,5
  strinfo:=" 随机数 " & string(rand(i))
  Text strinfo
   EndFor
EndPrgm
```

运行该程序：

```
text_demo()
```

一个对话框示例：

Then

请参阅 **If（** 第 45 页 ）。

tInterval

目录 >

tInterval *List*[,*Freq*[,*CLevel*]]

（ 数据数组输入 ）

tInterval $\bar{x}$,*sx*,*n*[,*CLevel*]

（ 摘要统计输入 ）

计算 *t* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	未知总体平均值的置信区间
stat.$\bar{x}$	正态随机分布的数据序列样本平均值

输出变量	说明
stat.ME	误差范围
stat.df	自由度
stat.σx	样本标准差
stat.n	带样本平均值的数据序列长度

tInterval_2Samp

目录 >

tInterval_2Samp
List1,*List2*[,*Freq1*[,*Freq2*[,*CLevel*[,*Pooled*]]]]

（ 数据数组输入 ）

tInterval_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*CLevel*[,*Pooled*]]

（ 摘要统计输入 ）

计算双样本 *t* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

Pooled=**1** 时合并方差　*Pooled*=**0** 时不合并方差。

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	包含置信水平分布概率的置信区间
stat.$\bar{x}$1-$\bar{x}$2	正态随机分布的数据序列样本平均值
stat.ME	误差范围
stat.df	自由度
stat.$\bar{x}$1　stat.$\bar{x}$2	正态随机分布的数据序列样本平均值
stat.σx1　stat.σx2	*List 1* 和 *List 2* *的样本标准差*
stat.n1　stat.n2	数据序列中的样本数
stat.sp	合并的标准差。 *Pooled* = YES 时的计算结果

tPdf()

目录 >

tPdf(*XVal*,*df***)** ⟹　如果 *XVal* 是数值，则结果为 *数值*，如果 *XVal* 是数组，则结果为 *数组*。

计算 *x* 为指定值时　指定自由度 *df* 的学生 *t* 分布概率密度函数 (pdf)。

trace()

目录 >

trace(*squareMatrix***)** ⇒ *值*

返回 *squareMatrix* 的跟踪值（ 主对角线上所有元素之和 ）。

$\text{trace}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right)$ 15

$a:=12$ 12

$\text{trace}\left(\begin{bmatrix}a&0\\1&a\end{bmatrix}\right)$ 24

Try

目录 >

Try
  block1
Else
  block2
EndTry

如果无错误产生　执行 *block1*。如果 *block1* 出错　则程序转而执行 *block2*。系统变量 *errCode* 包含允许程序进行错误恢复的错误代码。有关错误代码的列表　请参阅 **"** 错误代码和消息 **"**（ 第 140 页 ）。

block1 和 *block2* 可以是一条语句　也可以是以 **":"** 字符分隔的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

```
Define prog1()=Prgm
  Try
  z:=z+1
  Disp "z incremented."
  Else
  Disp "Sorry, z undefined."
  EndTry
  EndPrgm
```

Done

```
z:=1:prog1()
```

z incremented.

Done

```
DelVar z:prog1()
```

Sorry, z undefined.

Done

示例 2

要在运算中查看 **Try**　**ClrErr** 和 **PassErr** 命令　请如右侧所示输入 eigenvals() 程序。通过执行以下各表达式来运行程序。

$eigenvals\left(\begin{bmatrix}-3\\-41\\5\end{bmatrix},\begin{bmatrix}-1&2&-3.1\end{bmatrix}\right)$

注意：另请参阅第 17 页的 **ClrErr** 和第 73 页的 **PassErr**。

```
Define eigenvals(a,b)=Prgm
© Program 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)
Else
  If errCode=230 Then
    Disp "Error:Product of A·B must be a square matrix"
    ClrErr
  Else
    PassErr
  EndIf
EndTry
EndPrgm
```

tTest 目录 >

tTest μ*0*,*List*[,*Freq*[,*Hypoth*]]

（ 数据数组输入 ）

tTest μ*0*,$\overline{x}$,*sx*,*n*,[*Hypoth*]

（ 摘要统计输入 ）

当总体标准差 σ 未知时对单一未知总体平均值 μ 进行假设检验。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

依据以下规则之一检验 H_0：μ = μ0：

对于 H_a：μ < μ0　设置 *Hypoth*<0
对于 H_a：μ ≠ μ0（ 默认值 ） 设置 *Hypoth*0
对于 H_a：μ > μ0　设置 *Hypoth*>0

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.t	($\overline{x}$ − μ0) / (stdev / sqrt(n))
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	自由度
stat.$\overline{x}$	*List 中数据序列的样本平均值*
stat.sx	数据序列的样本标准差
stat.n	样本的大小

tTest_2Samp 目录 >

tTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*[,*Pooled*]]]]

（ 数据数组输入 ）

tTest_2Samp $\overline{x}$*1*,*sx1*,*n1*,$\overline{x}$*2*,*sx2*,*n2*[,*Hypoth*[,*Pooled*]]

（ 摘要统计输入 ）

计算双样本 *t* 检验。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

依据以下规则之一检验 H_0：μ = μ2：

对于 H_a：μ < μ2　设置 *Hypoth*<0
对于 H_a：μ ≠ μ2（ 默认值 ） 设置 *Hypoth*0
对于 H_a：μ > μ2　设置 *Hypoth*>0

Pooled=**1** 时合并方差
Pooled=**0** 时不合并方差

有关数组中空元素结果的信息　请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.t	计算的平均值差值的标准正规值
stat.PVal	可拒绝零假设的最小显著性水平
stat.df	t 统计的自由度
stat.$\bar{x}$1 stat.$\bar{x}$2	*List1* 和 *List2* *中数据序列的样本平均值*
stat.sx1 stat.sx2	*List1* 和 *List2* *中数据序列的样本标准差*
stat.n1 stat.n2	样本的大小
stat.sp	合并的标准差。 *Pooled*=1. 时的计算结果。

tvmFV() 目录 >

tvmFV(*N,I,PV,Pmt,[PpY],[CpY],[PmtAt]***)** ⇒ *值*

计算货币终值的财务函数。

注意： TVM 函数中使用的自变量已在 TVM 自变量表格中列出（ 第 109 页 ）。另请参阅 **amortTbl()**（ 第 6 页 ）。

tvmFV(120,5,0,-500,12,12) 77641.1

tvmI() 目录 >

tvmI(*N,PV,Pmt,FV,[PpY],[CpY],[PmtAt]***)** ⇒ *值*

计算年利率的财务函数。

注意： TVM 函数中使用的自变量已在 TVM 自变量表格中列出（ 第 109 页 ）。另请参阅 **amortTbl()**（ 第 6 页 ）。

tvmI(240,100000,-1000,0,12,12) 10.5241

tvmN() 目录 >

tvmN(*I,PV,Pmt,FV,[PpY],[CpY],[PmtAt]***)** ⇒ *值*

计算支付期数量的财务函数。

注意： TVM 函数中使用的自变量已在 TVM 自变量表格中列出（ 第 109 页 ）。另请参阅 **amortTbl()**（ 第 6 页 ）。

tvmN(5,0,-500,77641,12,12) 120.

tvmPmt() 目录 >

tvmPmt(*N,I,PV,FV,[PpY],[CpY],[PmtAt]***)** ⇒ *值*

计算每次支付金额的财务函数。

注意： TVM 函数中使用的自变量已在 TVM 自变量表格中列出（ 第 109 页 ）。另请参阅 **amortTbl()**（ 第 6 页 ）。

tvmPmt(60,4,30000,0,12,12) -552.496

tvmPV() 目录 >

tvmPV(*N,I,Pmt,FV,[PpY],[CpY],[PmtAt]***)** ⇒ *值*

计算现值的财务函数。

注意： TVM 函数中使用的自变量已在 TVM 自变量表格中列出（ 第 109 页 ）。另请参阅 **amortTbl()**（ 第 6 页 ）。

tvmPV(48,4,-500,30000,12,12) -3426.7

TVM 自变量 *	说明	数据类型
N	支付期数量	实数
I	年利率	实数
PV	现值	实数
Pmt	支付金额	实数
FV	终值	实数
PpY	每年支付次数　默认值 =1	> 0 的整数
CpY	每年的复利期数　默认值 =1	> 0 的整数
PmtAt	每个支付期结束或开始时的应付账款　默认值 = 结束时	整数（ 0= 结束时 1= 开始时 ）

***** 这些货币时间价值自变量名称类似于 Calculator 应用程序的财务求解器所用的 TVM 变量名称（ 例如 **tvm.pv** 和 **tvm.pmt** ）。不过　财务函数不会将其自变量值或结果保存到 TVM 变量。

TwoVar

目录 >

TwoVar *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

计算 TwoVar 统计值。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

除 *Include* 外　所有数组必须有相同维数。

X 和 *Y* 分别是自变量和因变量的数组。

Freq 是由频率值组成的可选数组。 *Freq* 中的每个元素指定各相应 *X* 和 *Y* 数据点的出现频率。默认值为 1。所有元素必须为 ≥0 的整数。

Category 是相应 *X* 和 *Y* 数据类别代码组成的数组。

Include 是由一个或多个类别代码组成的数组。计算值仅包括类别代码包含在此数组中的数据项。

数组 *X*　*Freq* 或 *Category* 中任意一个数组的空（ 空值 ）元素都会导致所有这些数组中对应元素为空值。数组 *X1* 到 *X20* 中任意一个数组的空元素都会导致所有这些数组中对应元素为空值。有关空元素的更多信息　请参阅第 134 页。

输出变量	说明
stat.$\bar{x}$	x 值的平均值
stat.Σx	x 值之和
stat.Σx2	x2 值之和
stat.sx	x 的样本标准差
stat.σx	x 的总体标准差
stat.n	数据点的数量

输出变量	说明
stat.$\bar{y}$	y 值的平均值
stat.Σy	y 值之和
stat.Σy^2	y2 值之和
stat.sy	y 的样本标准差
stat.σy	y 的总体标准差
stat.Σxy	x·y 值的和
stat.r	相关系数
stat.MinX	x 值的最小值
stat.Q_1X	x 的第一个四分位数
stat.MedianX	x 的中位数
stat.Q_3X	x 的第三个四分位数
stat.MaxX	x 值的最大值
stat.MinY	y 值的最小值
stat.Q_1Y	y 的第一个四分位数
stat.MedY	y 的中位数
stat.Q_3Y	y 的第三个四分位数
stat.MaxY	y 值的最大值
stat.$\Sigma(x-\bar{x})^2$	x 平均值的方差和
stat.$\Sigma(y-\bar{y})^2$	y 平均值的方差和

U

unitV()

目录 >

unitV(*Vector1***)** ⇒ *向量*

根据 *Vector1* 的格式返回单位行向量或列向量。

Vector1 必须是单行矩阵或单列矩阵。

$$\text{unitV}\left(\begin{bmatrix}1 & 2 & 1\end{bmatrix}\right) \quad \begin{bmatrix}0.408248 & 0.816497 & 0.408248\end{bmatrix}$$

$$\text{unitV}\left(\begin{bmatrix}1\\2\\3\end{bmatrix}\right) \quad \begin{bmatrix}0.267261\\0.534522\\0.801784\end{bmatrix}$$

unLock

目录 >

unLock *Var1*[, *Var2*] [, *Var3*] ...
unLock *Var*.

给指定的变量或变量组解锁。锁定的变量无法修改或删除。

请参阅 **Lock**（ 第 56 页 ）和 **getLockInfo()**（ 第 42 页 ）。

$a:=65$	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*

V

varPop()

目录 >

varPop(*List*[, *freqList*]**)** ⇒ *表达式*

返回 *List* 的总体方差。

freqList 中的元素为 *List* 中各对应元素出现的次数。

注意：*List* 必须至少包含两个元素。

如果任一数组中的元素为空（ 空值 ） 则该元素将被忽略并且另一数组中的对应元素也将被忽略。有关空元素的更多信息 请参阅第 134 页。

varPop$(\{5,10,15,20,25,30\})$	72.9167

varSamp()

目录 >

varSamp(*List*[, *freqList*]**)** ⇒ *表达式*

返回 *List* 的样本方差。

freqList 中的元素为 *List* 中各对应元素出现的次数。

注意：*List* 必须至少包含两个元素。

如果任一数组中的元素为空（ 空值 ） 则该元素将被忽略并且另一数组中的对应元素也将被忽略。有关空元素的更多信息 请参阅第 134 页。

varSamp$(\{1,2,5,-6,3,-2\})$	$\frac{31}{2}$
varSamp$(\{1,3,5\},\{4,6,2\})$	$\frac{68}{33}$

varSamp(*Matrix1*[, *freqMatrix*]**)** ⇒ *矩阵*

返回一个由 *Matrix1* 中各列样本方差组成的行向量。

freqMatrix 中的元素为 *Matrix1* 中各对应元素出现的次数。

如果任一矩阵中的元素为空（ 空值 ） 则该元素将被忽略并且另一矩阵中的对应元素也将被忽略。有关空元素的更多信息 请参阅第 134 页。

注意：*Matrix1* 必须至少包含两行。

varSamp$\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}$	$\begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$
varSamp$\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix},\begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$	$\begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$

warnCodes()

目录 >

warnCodes(*表达式 1*, *状态变量* **)** ⇒ *表达式*

计算表达式 *表达式 1* 返回结果 并在*状态变量*数组变量中存储任何生成的警告的代码。如果没有生成任何警告 则此函数会为*状态变量*赋值一个空数组。

表达式 1 可以是任何有效的 TI-Nspire™ 或 TI-Nspire™ CAS 数学表达式。您不能使用命令或赋值作为 *表达式 1*。

*状态变量*必须是有效的变量名称。

有关警告代码的列表和相关消息 请参阅第 145 页。

$\text{warnCodes}\left(\text{solve}\left(\sin(10\cdot x)=\frac{x^2}{x},x\right),warn\right)$

$x=-0.84232 \text{ or } x=-0.706817 \text{ or } x=-0.285234 \text{ or } x=$

$warn \qquad \{10007,10009\}$

要查看完整结果 请按 ▲ 然后使用 ◀ 和 ▶ 移动光标。

when()

目录 >

when(*Condition***,** *trueResult* [**,** *falseResult*][**,** *unknownResult*]**)**
⇒ *表达式*

根据 *Condition* 的取值是 true false 还是 unknown 返回 *trueResult* *falseResult* 或 *unknownResult*。如果自变量不足以得出合理的结果 则返回输入值。

省略 *falseResult* 和 *unknownResult* 可仅在 *Condition* 的值为 true 的区域中定义表达式。

使用 **undef** *falseResult* 可定义仅在某个区间内作图的表达式。

when() 对于定义递归函数非常有用。

$\text{when}(x<0,x+3)\,|\,x=5 \qquad \text{undef}$

$\text{when}(n>0,n\cdot factoral(n-1),1)\rightarrow factoral(n)$ *Done*

$factoral(3) \qquad 6$

$3! \qquad 6$

While

目录 >

While *Condition*
Block
EndWhile

只要 *Condition* 为 true 就执行 *Block* 中的语句。

Block 可以是一条语句 也可以是以 **":"** 字符分隔的一系列语句。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中 您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

```
Define sum_of_recip(n)=Func
                         Local i,tempsum
                         1→i
                         0→tempsum
                         While i≤n
                         tempsum+1/i→tempsum
                         i+1→i
                         EndWhile
                         Return tempsum
                         EndFunc
                                                Done
sum_of_recip(3)                                 11/6
```

X

xor 目录 >

布尔表达式 1 **xor** *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 **xor** *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 **xor** *布尔矩阵 2* 返回 *布尔矩阵*

如果 *BooleanExpr1* 为 true *BooleanExpr2* 为 false 则返回 true 反之亦然。

如果两个自变量均为 true 或均为 false 则返回 false。如果两个自变量中的任何一个都无法确定为 true 或 false 则返回简化的布尔表达式。

注意：请参阅 **or**（ 第 72 页 ）。

Integer1 **xor** *Integer2* ⟹ *整数*

使用 **xor** 运算逐位比较两个实整数。在内部运算中 两个整数都将转换为带符号的 64 位二进制数字。比较对应的位时 如果任何一位（ 但不是两位同时 ）为 1 则结果为 1 如果两位均为 0 或两位均为 1 则结果为 0。返回的值代表位结果 将根据 Base 模式显示。

您可以输入任何进位制的整数。对于按二进制或十六进制输入的整数 您必须分别使用 0b 或 0h 前缀。不带前缀的整数都将被视为十进制（ 基数为 10)。

如果您输入的十进制整数对于带符号的 64 位二进制形式来说过大 可使用对称的模数运算将该值纳入合理的范围。更多信息 请参阅 ▶Base2（ 第 12 页 ）。

注意：请参阅 **or**（ 第 72 页 ）。

true xor true	false
5>3 xor 3>5	true

在 Hex 模式下：
重要信息： ??? ¨ ???÷?? 0° ?

0h7AC36 xor 0h3D5F	0h79169

在 Bin 模式下：

0b100101 xor 0b100	0b100001

注意：二进制输入最多可为 64 位（ 不包括 0b 前缀 ）。十六进制输入最多可为 16 位。

Z

zInterval 目录 >

zInterval σ,*List*[,*Freq*[,*CLevel*]]

（ 数据数组输入 ）

zInterval σ,$\bar{x}$,*n* [,*CLevel*]

（ 摘要统计输入 ）

计算 *z* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	未知总体平均值的置信区间
stat.$\bar{x}$	正态随机分布的数据序列样本平均值
stat.ME	误差范围
stat.sx	样本标准差

输出变量	说明
stat.n	带样本平均值的数据序列长度
stat.σ	数据序列 *List* 的已知总体标准差

zInterval_1Prop

目录 >

zInterval_1Prop *x,n* [*,CLevel*]

计算单比例 *z* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

x 为非负整数。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 " （ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	包含置信水平分布概率的置信区间
stat.$\hat{p}$	计算的成功比例
stat.ME	误差范围
stat.n	数据序列中的样本数

zInterval_2Prop

目录 >

zInterval_2Prop *x1,n1,x2,n2*[*,CLevel*]

计算双比例 *z* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

x1 和 *x2* 为非负整数。

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 " （ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	包含置信水平分布概率的置信区间
stat.$\hat{p}$Diff	计算的两个比例间差值
stat.ME	误差范围
stat.$\hat{p}$1	第一个样本比例估算
stat.$\hat{p}$2	第二个样本比例估算
stat.n1	数据序列一中的样本大小
stat.n2	数据序列二中的样本大小

zInterval_2Samp σ_1,σ_2 *,List1,List2*[*,Freq1*[*,Freq2,*[*CLevel*]]]

（ 数据数组输入 ）

zInterval_2Samp $\sigma_1,\sigma_2\bar{x}1,n1,\bar{x}2,n2$[*,CLevel*]

（ 摘要统计输入 ）

计算双样本 *z* 置信区间。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.CLower stat.CUpper	包含置信水平分布概率的置信区间
stat.$\bar{x}1-\bar{x}2$	正态随机分布的数据序列样本平均值
stat.ME	误差范围
stat.$\bar{x}1$ stat.$\bar{x}2$	正态随机分布的数据序列样本平均值
stat.$\sigma x1$ stat.$\sigma x2$	*List 1* 和 *List 2 的样本标准差*
stat.n1 stat.n2	数据序列中的样本数
stat.r1 stat.r2	数据序列 *List 1* 和 *List 2 的已知总体标准差*

zTest

zTest $\mu 0,\sigma$*,List,*[*Freq*[*,Hypoth*]]

（ 数据数组输入 ）

zTest $\mu 0,\sigma,\bar{x}$*,n*[*,Hypoth*]

（ 摘要统计输入 ）

使用频率 *freqlist* 执行 *z* 检验。结果摘要存储在 *stat.results* 变量中。（ 请参阅第 98 页。 ）

依据以下规则之一检验 H_0：$\mu = \mu 0$：

对于 H_a：$\mu < \mu 0$ 设置 *Hypoth*<0

对于 H_a：$\mu \neq \mu 0$（ 默认值 ） 设置 *Hypoth*0

对于 H_a：$\mu > \mu 0$ 设置 *Hypoth*>0

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

输出变量	说明
stat.z	$(\bar{x} - \mu 0) / (\sigma / \text{sqrt}(n))$

输出变量	说明
stat.P Value	可拒绝零假设的最小概率
stat.$\overline{x}$	*List 中数据序列的样本平均值*
stat.sx	数据序列的样本标准差。仅返回 *Data* 输入值。
stat.n	样本的大小

zTest_1Prop 目录 >

zTest_1Prop *p0*,*x*,*n*[,*Hypoth*]

计算单比例 *z* 检验。结果摘要存储在 *stat.results* 变量中。
（ 请参阅第 98 页。 ）

x 为非负整数。

依据以下规则之一检验 H_0：*p = p0*：

对于 H_a：*p > p0* 设置 *Hypoth*>0
对于 H_a：*p ≠ p0*（ *默认值* ） 设置 *Hypoth*=0
对于 H_a：*p < p0* 设置 *Hypoth*<0

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "
（ 第 134 页 ）。

输出变量	说明
stat.p0	假设的总体比例
stat.z	计算的比例标准正规值
stat.PVal	可拒绝零假设的最小显著性水平
stat.$\hat{p}$	估算的样本比例
stat.n	样本的大小

zTest_2Prop 目录 >

zTest_2Prop *x1*,*n1*,*x2*,*n2*[,*Hypoth*]

计算双比例 *z* 检验。结果摘要存储在 *stat.results* 变量中。
（ 请参阅第 98 页。 ）

x1 和 *x2* 为非负整数。

依据以下规则之一检验 H_0：*p1 = p2*：

对于 H_a：*p1 > p2* 设置 *Hypoth*>0
对于 H_a：*p1 ≠ p2*（ 默认值 ） 设置 *Hypoth*0
对于 H_a：*p < p0* 设置 *Hypoth*<0

有关数组中空元素结果的信息 请参阅 " 空（ 空值 ）元素 "
（ 第 134 页 ）。

输出变量	说明
stat.z	计算的比例差值标准正规值
stat.PVal	可拒绝零假设的最小显著性水平
stat.$\hat{p}1$	第一个样本比例估算
stat.$\hat{p}2$	第二个样本比例估算
stat.$\hat{p}$	合并样本比例估算
stat.n1 stat.n2	取自尝试 1 和 2 的样本数

zTest_2Samp 目录 >

zTest_2Samp σ_1,σ_2 ,*List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

（ 数据数组输入 ）

zTest_2Samp $\sigma_1,\sigma_2\bar{x}1,n1,\bar{x}2,n2$[,*Hypoth*]

（ 摘要统计输入 ）

计算双样本 *z* 检验。结果摘要存储在 *stat.results* 变量中。
（ 请参阅第 98 页。 ）

依据以下规则之一检验 H_0：$\mu = \mu 2$：

对于 H_a：$\mu 1 < \mu 2$ 设置 *Hypoth*<0

对于 H_a：$\mu 1 \neq \mu 2$ 设置 *Hypoth*0

对于 H_a：$\mu 1 > \mu 2$ 设置 *Hypoth*>0

有关数组中空元素结果的信息 请参阅 “ 空（ 空值 ）元素 ”
（ 第 134 页 ）。

输出变量	说明
stat.z	计算的平均值差值的标准正规值
stat.PVal	可拒绝零假设的最小显著性水平
stat.$\bar{x}1$ stat.$\bar{x}2$	*List1* 和 *List2* *中数据序列的样本平均值*
stat.sx1 stat.sx2	*List1* 和 *List2* *中数据序列的样本标准差*
stat.n1 stat.n2	样本的大小

+（加） ⊞ 键

Value1 + *Value2* ⇒ *值*

返回两个自变量之和。

56	56
$56+4$	60
$60+4$	64
$64+4$	68
$68+4$	72

List1 + *List2* ⇒ *数组*

Matrix1 + *Matrix2* ⇒ *矩阵*

返回一个数组（或矩阵）其元素为 *List1* 和 *List2*（或 *Matrix1* 和 *Matrix2*）中对应元素之和。

两个自变量的维数必须相等。

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\{22,3.14159,1.5708\}$
$\left\{10,5,\frac{\pi}{2}\right\}\rightarrow l2$	$\{10,5,1.5708\}$
$l1+l2$	$\{32,8.14159,3.14159\}$

返回一个数组　其元素为 *Expr* 与 *List1* 中每个元素的和。

$15+\{10,15,20\}$	$\{25,30,35\}$
$\{10,15,20\}+15$	$\{25,30,35\}$

Value + *List1* ⇒ *数组*

List1 + *Value* ⇒ *数组*

返回一个数组　其元素为 *Value* 与 *List1* 中每个元素的和。

返回一个矩阵　其对角线上的元素为 *Expr* 与 *Matrix1* 对角线上的各元素相加的和。*Matrix1* 必须为方阵。

$20+\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\ 3 & 24\end{bmatrix}$

Value + *Matrix1* ⇒ *矩阵*

Matrix1 + *Value* ⇒ *矩阵*

返回一个矩阵　其对角线上的元素为 *Value* 与 *Matrix1* 对角线上的各元素相加的和。*Matrix1* 必须为方阵。

注意：使用 **.+**（点和 +）可将表达式分别与每个元素相加。

−（减） ⊟ 键

Value1 − *Value2* ⇒ *值*

返回 *Value1* 减去 *Value2* 的差值。

$6-2$	4
$\pi-\frac{\pi}{6}$	2.61799

List1 − *List2* ⇒ *数组*

Matrix1 − *Matrix2* ⇒ *矩阵*

返回一个数组（或矩阵）其元素为 *List1*（或 *Matrix1*）中的元素减去 *List2*（或 *Matrix2*）中对应元素的差值。

两个自变量的维数必须相等。

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,-1.85841,0.\}$
$\begin{bmatrix}3 & 4\end{bmatrix}-\begin{bmatrix}1 & 2\end{bmatrix}$	$\begin{bmatrix}2 & 2\end{bmatrix}$

−（减）

− 键

返回一个数组　其元素为 *Expr* 减去 *List1* 各元素的差值或 *List1* 各元素减去 *Expr* 的差值。

Value − *List1* ⇒ *数组*

List1 − *Value* ⇒ *数组*

返回一个数组　其元素为 *Value* 减去 *List1* 各元素的差值或 *List1* 各元素减去 *Value* 的差值。

15−{10,15,20}　{5,0,−5}

{10,15,20}−15　{−5,0,5}

ExprValue − *Matrix1* ⇒ *矩阵*

Matrix1 − *Value* ⇒ *矩阵*

Value − *Matrix1* 返回一个矩阵　其元素为 *Value* 乘以单位矩阵再减去 *Matrix1* 得到的值。 *Matrix1* 必须为方阵。

Matrix1 − *Value* 返回一个矩阵　其元素为 *Matrix1* 减去 *Value* 与单位矩阵的乘积后得到的值。 *Matrix1* 必须为方阵。

20−$\begin{bmatrix}1&2\\3&4\end{bmatrix}$　$\begin{bmatrix}19&-2\\-3&16\end{bmatrix}$

注意：使用 **.−**（点差）可从各元素分别减去表达式。

·（乘）

× 键

Value1 ·*Value2* ⇒ *值*

返回两个自变量的乘积。

2·3.45　6.9

List1°§*List2* ⇒ *数组*

返回一个数组　其元素为 *List1* 和 *List2* 中各对应元素的乘积。

两个数组的维数必须相等。

{1.,2,3}·{4,5,6}　{4,10,18}

Matrix1 ·*Matrix2* ⇒ *矩阵*

返回 *Matrix1* 和 *Matrix2* 的矩阵乘积。

Matrix1 的列数必须与 *Matrix2* 的行数相等。

$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\cdot\begin{bmatrix}7&8\\7&8\\7&8\end{bmatrix}$　$\begin{bmatrix}42&48\\105&120\end{bmatrix}$

返回一个数组　其元素为 *Expr* 与 *List1* 中各元素的乘积。

π·{4,5,6}　{12.5664,15.708,18.8496}

Value ·*List1* ⇒ *数组*

List1 ·*Value* ⇒ *数组*

返回一个数组　其元素为 *Value* 与 *List1* 中各元素的乘积。

返回一个矩阵　其元素为 *Expr* 与 *Matrix1* 中各元素的乘积。

$\begin{bmatrix}1&2\\3&4\end{bmatrix}\cdot 0.01$　$\begin{bmatrix}0.01&0.02\\0.03&0.04\end{bmatrix}$

Value ·*Matrix1* ⇒ *矩阵*

Matrix1 ·*Value* ⇒ *矩阵*

返回一个矩阵　其元素为 *Value* 与 *Matrix1* 中各元素的乘积。

6·identity(3)　$\begin{bmatrix}6&0&0\\0&6&0\\0&0&6\end{bmatrix}$

注意：使用 **.·**（点积）可将表达式分别与每个元素相乘。

／（除） ÷ 键

Value1 ／ *Value2* ⇒ *值*

返回 *Value1* 除以 *Value2* 的商。

注意：另请参阅**分数模板**（第 1 页）。

$\frac{2}{3.45}$ .57971

List1 ／ *List2* ⇒ *数组*

返回一个由 *List1* 除以 *List2* 的商组成的数组。

两个数组的维数必须相等。

$\frac{\{1.,2,3\}}{\{4,5,6\}}$ $\left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$

返回一个数组　其元素为 *Expr* 除以 *List1* 中各元素的商或 *List1* 中的各元素除以 *Expr* 的商。

$\frac{6}{\{3,6,\sqrt{6}\}}$ $\{2,1,2.44949\}$

Value ／ *List1* ⇒ *数组*

List1 ／ *Value* ⇒ *数组*

返回一个数组　其元素为 *Value* 除以 *List1* 中各元素的商或 *List1* 中的各元素除以 *Value* 的商。

$\frac{\{7,9,2\}}{7\cdot9\cdot2}$ $\left\{\frac{1}{18},\frac{1}{14},\frac{1}{63}\right\}$

Value ／ *Matrix1* ⇒ *矩阵*

Matrix1 ／ *Value* ⇒ *矩阵*

返回一个矩阵　其元素为 *Matrix1*／*Value* 的商。

$\frac{[7\ \ 9\ \ 2]}{7\cdot9\cdot2}$ $\begin{bmatrix}\frac{1}{18} & \frac{1}{14} & \frac{1}{63}\end{bmatrix}$

注意：使用 **.**／（点商）可使每个元素分别除以表达式。

^（乘方） ^ 键

Value1 ^ *Value2* ⇒ *值*

List1 ^ *List2* ⇒ *数组*

4^2 16

返回以第一个自变量为底　第二个自变量为乘方的结果。

$\{2,4,6\}^{\{1,2,3\}}$ $\{2,16,216\}$

注意：另请参阅**指数模板**（第 1 页）。

对于数组　返回以 *List1* 中各元素为底　*List2* 中对应元素为乘方的结果。

在实数域中　化简的奇分母分数乘方使用实数支　而在复数模式下使用主支。

Value ^ *List1* ⇒ *数组*

$\pi^{\{1,2,-3\}}$ $\{3.14159,9.8696,0.032252\}$

返回以 *Value* 为底　以 *List1* 各元素为乘方的计算结果。

List1 ^ *Value* ⇒ *数组*

$\{1,2,3,4\}^{-2}$ $\left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$

返回以 *List1* 中各元素为底　以 *Value* 为乘方的计算结果。

squareMatrix1 ^ *integer* ⇒ *矩阵*

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2$ $\begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$

返回以 *squareMatrix1* 为底　以 *integer* 为冥的计算结果。

squareMatrix1 必须为方阵。

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1}$ $\begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

如果 *integer* = −1　计算逆矩阵。

如果 *integer* < −1　以合适的正数乘方计算逆矩阵。

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2}$ $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x^2（ 平方 ）

x^2 键

Value1$^2 \Rightarrow$ *值*

返回自变量的平方。

List1$^2 \Rightarrow$ *数组*

返回一个数组　其元素为 *List1* 中各元素的平方。

squareMatrix1$^2 \Rightarrow$ *方阵*

返回 *squareMatrix1* 的矩阵平方　此运算不同于计算每个元素的平方。使用 .^2 可计算每个元素的平方。

.+（ 点加 ）

. + 键

Matrix1 .+ *Matrix2* $\Rightarrow$ *矩阵*

Value .+ *Matrix1* $\Rightarrow$ *矩阵*

Matrix1 .+ *Matrix2* 返回一个矩阵　其元素为 *Matrix1* 和 *Matrix2* 中各对应元素对的和。

Value .+ *Matrix1* 返回一个矩阵　其元素为 *Value* 与 *Matrix1* 中各元素的和。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} .+ \begin{bmatrix}10 & 30\\ 20 & 40\end{bmatrix} \qquad \begin{bmatrix}11 & 32\\ 23 & 44\end{bmatrix}$$

$$5 .+ \begin{bmatrix}10 & 30\\ 20 & 40\end{bmatrix} \qquad \begin{bmatrix}15 & 35\\ 25 & 45\end{bmatrix}$$

.−（ 点差 ）

. − 键

Matrix1 .− *Matrix2* $\Rightarrow$ *矩阵*

Value .− *Matrix1* $\Rightarrow$ *矩阵*

Matrix1 .− *Matrix2* 返回一个矩阵　其元素为 *Matrix1* 与 *Matrix2* 中各对应元素对的差。

Value .− *Matrix1* 返回一个矩阵　其元素为 *Value* 与 *Matrix1* 中各元素的差。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} .- \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}-9 & -18\\ -27 & -36\end{bmatrix}$$

$$5 .- \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}-5 & -15\\ -25 & -35\end{bmatrix}$$

.·（ 点积 ）

. × 键

Matrix1 .· *Matrix2* $\Rightarrow$ *矩阵*

Value .· *Matrix1* $\Rightarrow$ *矩阵*

Matrix1 .· *Matrix2* 返回一个矩阵　其元素为 *Matrix1* 和 *Matrix2* 中各对应元素对的乘积。

Value .· *Matrix1* 返回一个矩阵　其元素为 *Value* 与 *Matrix1* 中各元素的乘积。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} .\cdot \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}10 & 40\\ 90 & 160\end{bmatrix}$$

$$5 .\cdot \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}50 & 100\\ 150 & 200\end{bmatrix}$$

./（ 点商 ）

. ÷ 键

Matrix1 ./ *Matrix2* $\Rightarrow$ *矩阵*

Value ./ *Matrix1* $\Rightarrow$ *矩阵*

Matrix1 ./ *Matrix2* 返回一个矩阵　其元素为 *Matrix1* 和 *Matrix2* 中各对应元素对的商。

Value ./ *Matrix1* 返回一个矩阵　其元素为 *Value* 与 *Matrix1* 中各元素的商。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} ./ \left(\begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{10} & \frac{1}{10}\\ \frac{1}{10} & \frac{1}{10}\end{bmatrix}$$

$$5 ./ \left(\begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{2} & \frac{1}{4}\\ \frac{1}{6} & \frac{1}{8}\end{bmatrix}$$

.^（点乘方）

. ^ 键

Matrix1 **.^** *Matrix2* ⇒ *矩阵*

Value **.^** *Matrix1* ⇒ *矩阵*

Matrix1 **.^** *Matrix2* 返回一个矩阵　其中 *Matrix2* 中的元素是 *Matrix1* 中各对应元素的指数。

Value **.^** *Matrix1* 返回一个矩阵　其中 *Matrix1* 中的各元素是 *Value* 的指数。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}.^{\wedge}\begin{bmatrix}0 & 2\\ 3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 4\\ 27 & \frac{1}{4}\end{bmatrix}$$

$$5.^{\wedge}\begin{bmatrix}0 & 2\\ 3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 25\\ 125 & \frac{1}{5}\end{bmatrix}$$

-（求负）

(-) 键

-*Value1* ⇒ *值*

-*List1* ⇒ *数组*

-*Matrix1* ⇒ *矩阵*

返回自变量的负值。

对于数组或矩阵　返回所有元素的负值。

如果自变量为二进制或十六进制整数　则求负将计算出它们的补数。

$$-2.43 \qquad -2.43$$

$$-\{-1,0.4,1.2\text{E}19\} \qquad \{1.,-0.4,-1.2\text{E}19\}$$

在 Bin 模式下：

重要信息：零　非字母 O

$$0b100101\blacktriangleright\text{Dec} \qquad 37$$

$$-0b100101$$
$$0b1111111111111111111111111111111\blacktriangleright$$

$$Ans\blacktriangleright\text{Dec} \qquad -37$$

要查看整个结果　请按 ▲　然后使用 ◀ 和 ▶ 移动光标。

%（百分比）

ctrl 键

Value1 **%** ⇒ *值*

List1 **%** ⇒ *数组*

Matrix1 **%** ⇒ *矩阵*

返回 $\frac{argument}{100}$

对于数组或矩阵　返回由各元素分别除以 100 组成的数组或矩阵。

按 **Ctrl+Enter** ctrl enter (Macintosh®:⌘+**Enter**) 进行计算：

$$13\% \qquad 0.13$$

按 **Ctrl+Enter** ctrl enter (Macintosh®:⌘+**Enter**) 进行计算：

$$\{\{1,10,100\}\}\% \qquad \{0.01,0.1,1.\}$$

= （ 等于 ）

= 键

Expr1 = *Expr2* ⟹ *布尔表达式*

List1 = *List2* ⟹ *布尔数组*

Matrix1 = *Matrix2* ⟹ *布尔矩阵*

如果确定 *Expr1* 等于 *Expr2* 则返回 true。

如果确定 *Expr1* 不等于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

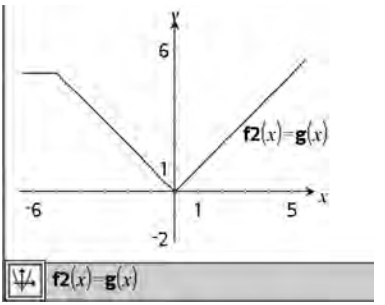
输入示例时需注意的事项：在手持设备的 Calculator 应用程序中 您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上 按住 **Alt** 然后按 **Enter**。

示例函数给出了使用数学测试符号的结果：=, ≠, <, ≤, >, ≥

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

绘制 g(x) 的结果



≠ （ 不等于 ）

ctrl = 键

Expr1 ≠ *Expr2* ⟹ *布尔表达式*

List1 ≠ *List2* ⟹ *布尔数组*

Matrix1 ≠ *Matrix2* ⟹ *布尔矩阵*

如果确定 *Expr1* 不等于 *Expr2* 则返回 true。

如果确定 *Expr1* 等于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

注意：您可以通过在计算机键盘上键入 **/=**
ðÂ»Î¥À'ÀÀ„²°°£

请参阅 **"="**（ 等于 ）示例。

< （ 小于 ） ctrl = 键

Expr1 < *Expr2* ⇒ *布尔表达式*

List1 < *List2* ⇒ *布尔数组*

Matrix1 < *Matrix2* ⇒ *布尔矩阵*

如果确定 *Expr1* 小于 *Expr2* 则返回 true。

如果确定 *Expr1* 大于或等于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

请参阅 **"="**（ 等于 ）示例。

≤（ 小于或等于 ） ctrl = 键

Expr1 ≤ *Expr2* ⇒ *布尔表达式*

List1 ≤ *List2* ⇒ *布尔数组*

Matrix1 ≤ *Matrix2* ⇒ *布尔矩阵*

如果确定 *Expr1* 小于或等于 *Expr2* 则返回 true。

如果确定 *Expr1* 大于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

注意： 您可以通过在计算机键盘上键入 **<=**
õÂ»Î¥À'ÀÀ„²°°£

请参阅 **"="**（ 等于 ）示例。

> （ 大于 ） ctrl = 键

Expr1 > *Expr2* ⇒ *布尔表达式*

List1 > *List2* ⇒ *布尔数组*

Matrix1 > *Matrix2* ⇒ *布尔矩阵*

如果确定 *Expr1* 大于 *Expr2* 则返回 true。

如果确定 *Expr1* 小于或等于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

请参阅 **"="**（ 等于 ）示例。

≥（ 大于或等于 ） ctrl = 键

Expr1 ≥ *Expr2* ⇒ *布尔表达式*

List1 ≥ *List2* ⇒ *布尔数组*

Matrix1 ≥ *Matrix2* ⇒ *布尔矩阵*

如果确定 *Expr1* 大于或等于 *Expr2* 则返回 true。

如果确定 *Expr1* 小于 *Expr2* 则返回 false。

其他情况则返回等式的简化形式。

对于数组和矩阵 返回各对应元素的比较结果。

注意： 您可以通过在计算机键盘上键入 **>=**
õÂ»Î¥À'ÀÀ„²°°£

请参阅 **"="**（ 等于 ）示例。

⇒（ 逻辑隐含式 ）

ctrl = 键

布尔表达式 1 ⇒ *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 ⇒ *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 ⇒ *布尔矩阵 2* 返回 *布尔矩阵*
整数 1 ⇒ *整数 2* 返回 *整数*

计算表达式 **not** < 自变量 1> **or** < 自变量 2> 并返回真 假或方程的简化形式。

列表和矩阵则按元素返回对比。

注意：您可以通过键盘输入 **=>** 来插入此运算符

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}
{1,2,3} ⇒ {3,2,1}	{-1,-1,-3}

⇔（ 逻辑双隐含式，XNOR ）

ctrl = 键

布尔表达式 1 ⇔ *布尔表达式 2* 返回 *布尔表达式*
布尔列表 1 ⇔ *布尔列表 2* 返回 *布尔列表*
布尔矩阵 1 ⇔ *布尔矩阵 2* 返回 *布尔矩阵*
整数 1 ⇔ *整数 2* 返回 *整数*

返回两个自变量 **XOR** 布尔运算的逻辑非。返回真 假或简化方程。

列表和矩阵则按元素返回对比。

注意：您可以通过键盘输入 **<=>** 来插入此运算符

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} ⇔ {3,2,1}	{-3,-1,-3}

!（ 阶乘 ）

?! 键

Value1!⇒ *值*
List1!⇒ *数组*
Matrix1!⇒ *矩阵*

返回自变量的阶乘。

对于数组或矩阵 返回由各元素阶乘组成的数组或矩阵。

5!	120
{{5,4,3}}!	{120,24,6}
$\left\{\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right\}!$	$\begin{bmatrix}1 & 2\\ 6 & 24\end{bmatrix}$

& 添加

ctrl 键

String1 & *String2* ⇒ *字符串*

返回将 *String2* 添加到 *String1* 之后的文本字符串。

"Hello "&"Nick"	"Hello Nick"

d()（ 导数 ）

目录 >

d(*Expr1*, *Var*[, *Order*]) | *Var*=*Value* ⇒ *值*

d(*Expr1*, *Var*[, *Order*]) ⇒ *值*

d(*List1*, *Var*[, *Order*]) ⇒ *数组*

d(*Matrix1*, *Var*[, *Order*]) ⇒ *矩阵*

除了使用第一个句法时以外　您必须在变量 *Var* 中存储一个数值　然后才能计算 **d()**。请参阅示例。

d() 可用于计算某一点的一阶导数和二阶导数（ 使用自动微分方法 ）。

Order（ 如包括 ）必须 =**1** 或 **2**。默认值为 **1**。

注意：您可以通过在计算机键盘上键入 **derivative(...)** 插入此函数。

$\frac{d}{dx}(|x|)|x=0$ undef

$x:=0:\frac{d}{dx}(|x|)$ undef

$x:=3:\frac{d}{dx}(\{x^2,x^3,x^4\})$ $\{6,27,108\}$

注意：另请参阅 **First derivative**（ 第 4 页 ）或 **Second derivative**（ 第 5 页 ）。

注意：d() 存在局限性：它会通过未简化的表达式进行递推运算　计算一阶导数（ 和二阶导数　如适用 ）的数值并计算每个子表达式　这可能会导致得到意外结果。

请注意右侧的示例。x=0 时　x·(x^2+x)^(1/3) 的一阶导数等于 0。但是　由于 x=0 时子表达式 (x^2+x)^(1/3) 的一阶导数未定义　且该值是用于计算整个表达式的导数　因此 **d()** 会将结果报告为未定义并显示警告信息。

如果您遇到此局限　请从图形上验证解。您还可以尝试使用 **centralDiff()**。

$\frac{d}{dx}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}}\right)|x=0$ undef

$\text{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)|x=0$ 0.000033

∫()（ 积分 ）

目录 >

∫(*Expr1*, *Var*, *Lower*, *Upper*) ⇒ *值*

返回 *Expr1* 关于变量 *Var* 从 *Lower* 到 *Upper* 的积分。可用于计算定积分数值（ 使用与 nInt() 相同的方法 ）。

注意：您可以通过在计算机键盘上键入 **integral(...)** 插入此函数。

注意：另请参阅 **nInt()**（ 第 67 页 ）和 **定 积分模板**（ 第 5 页 ）。

$\int_0^1 x^2\,dx$ 0.333333

√()（ 平方根 ）

ctrl x² 键

√ (*Value1*) ⇒ *值*

√ (*List1*) ⇒ *数组*

返回自变量的平方根。

对于数组　返回 *List1* 中所有元素的平方根。

注意：您可以通过在计算机键盘上键入 **sqrt(...)**ðÂ»Î¥À½Øþ˜£

注意：另请参阅**平方根模板**（ 第 1 页 ）。

$\sqrt{4}$ 2

$\sqrt{\{9,2,4\}}$ $\{3,1.41421,2\}$

Π() (prodSeq)

目录 >

Π(*Expr1***, *Var*, *Low*, *High*)** ⇒ *表达式*

注意：您可以通过在计算机键盘上键入 **prodSeq(...)** 插入此函数。

计算 *Expr1* 在变量 *Var* 从 *Low* 到 *High* 取值时所对应的结果 并返回这些结果的乘积。

注意：另请参阅**乘积模板 (Π)**（ 第 4 页 ）。

$\prod_{n=1}^{5}\left(\frac{1}{n}\right)$ → $\frac{1}{120}$

$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right)$ → $\left\{\frac{1}{120},120,32\right\}$

Π(*Expr1***, *Var*, *Low*, *Low*−1)** ⇒ 1

Π(*Expr1***, *Var*, *Low*, *High*)**
⇒ **1/Π(***Expr1***, *Var*, *High*+1, *Low*−1)** if *High* < *Low*−1

$\prod_{k=4}^{3}(k)$ → 1

使用的乘积公式引自以下参考资料：

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik.*Concrete Mathematics:A Foundation for Computer Science*.Reading, Massachusetts:Addison-Wesley, 1994.

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)$ → 6

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right)$ → $\frac{1}{4}$

Σ() (sumSeq)

目录 >

Σ(*Expr1***, *Var*, *Low*, *High*)** ⇒ *表达式*

注意：您可以通过在计算机键盘上键入 **sumSeq(...)** 插入此函数。

计算 *Expr1* 在变量 *Var* 从 *Low* 到 *High* 取值时所对应的结果 并返回这些结果的和。

注意：另请参阅**求和模板**（ 第 4 页 ）。

$\sum_{n=1}^{5}\left(\frac{1}{n}\right)$ → $\frac{137}{60}$

Σ(*Expr1***, *Var*, *Low*, *Low*−1)** ⇒ 0

Σ(*Expr1***, *Var*, *Low*, *High*)**
⇒ **−Σ(***Expr1***, *Var*, *High*+1, *Low*−1)** if *High* < *Low*−1

$\sum_{k=4}^{3}(k)$ → 0

使用的求和公式引自以下参考资料：

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik.*Concrete Mathematics:A Foundation for Computer Science*.Reading, Massachusetts:Addison-Wesley, 1994.

$\sum_{k=4}^{1}(k)$ → −5

$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k)$ → 4

ΣInt()

目录 >

ΣInt(*NPmt1***, *NPmt2*, *N*, *I*, *PV* ,[*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*])** ⇒ *值*

ΣInt(*NPmt1***,***NPmt2***,***amortTable***)** ⇒ *值*

计算指定支付范围内需支付的利息之和的分期偿还函数。

NPmt1 和 *NPmt2* 定义支付范围的起始和结束日期。

N、*I*、*PV*、*Pmt*、*FV*、*PpY*、*CpY* 和 *PmtAt* 在 TVM 自变量表中有介绍（第 109 页）。

- 如果您省略 *Pmt*，则使用其默认值 *Pmt*=**tvmPmt(***N*,*I*,*PV*,*FV*,*PpY*,*CpY*,*PmtAt***)**。
- 如果您省略 *FV*，则使用其默认值 *FV*=0。
- *PpY*、*CpY* 和 *PmtAt* 的默认值与用于 TVM 函数的值相同。

roundValue 指定四舍五入的小数位数。默认保留两位小数。

ΣInt(*NPmt1***,***NPmt2***,***amortTable***)** 计算基于分期偿还表 *amortTable* 的利息之和。*amortTable* 自变量必须为 **amortTbl()**（第 6 页）下所介绍形式的矩阵。

注意：另请参阅下文的 ΣPrn() 和第 12 页的 **Bal()**。

ΣInt(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

ΣInt(1,3,*tbl*) -213.48

ΣPrn()

目录 >

ΣPrn(*NPmt1***, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*])** ⇒ *值*

ΣPrn(*NPmt1***,***NPmt2***,***amortTable***)** ⇒ *值*

计算指定支付范围内需支付的本金之和的分期偿还函数。

NPmt1 和 *NPmt2* 定义支付范围的起始和结束日期。

N、*I*、*PV*、*Pmt*、*FV*、*PpY*、*CpY* 和 *PmtAt* 在 TVM 自变量表中有介绍（第 109 页）。

- 如果您省略 *Pmt*，则使用其默认值 *Pmt*=**tvmPmt(***N*,*I*,*PV*,*FV*,*PpY*,*CpY*,*PmtAt***)**。
- 如果您省略 *FV*，则使用其默认值 *FV*=0。
- *PpY*、*CpY* 和 *PmtAt* 的默认值与用于 TVM 函数的值相同。

roundValue 指定四舍五入的小数位数。默认保留两位小数。

ΣPrn(*NPmt1***,***NPmt2***,***amortTable***)** 计算基于分期偿还表 *amortTable* 的本金之和。*amortTable* 自变量必须为 **amortTbl()**（第 6 页）下所介绍形式的矩阵。

注意：另请参阅上文的 ΣInt() 和第 12 页的 **Bal()**。

ΣPrn(1,3,12,4.75,20000,,12,12) -4916.28

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
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

ΣPrn(1,3,*tbl*) -4916.28

（ 间接引用 ）

ctrl 键

 # varNameString

调用名称为 *varNameString* 的变量。借助此功能 您可以在函数中使用字符串创建变量名称。

$xyz:=12$	12
$\#("x"\&"y"\&"z")$	12

创建或调用变量 xyz。

$10\rightarrow r$	10
$"r"\rightarrow s1$	"r"
$\#s1$	10

返回名称存储在变量 **s1** 中的变量 **(r)** 的值。

E （ 科学计数法 ）

EE 键

*mantissa***E***exponent*

输入一个科学记数法的数值。数值将表示为 *mantissa* × $10^{exponent}$。

提示：如果您要输入 10 的乘方而不引入十进制数值结果结果 请使用 **10^** *整数*。

注意： 您可以通过在计算机键盘上键入 **@E** 插入此运算符。例如 键入 **2.3@E4** 便可输入 2.3**E**4。

23000.	23000.
2300000000.+4.1E15	4.1E15
$3\cdot 10^4$	30000

g （ 百分度 ）

π▸ 键

*Expr1*g ⇒ *表达式*

*List1*g ⇒ *数组*

*Matrix1*g ⇒ *矩阵*

此函数让您能够在 Degree 或 Radian 模式下使用百分度角度。

在 Radian 角度模式下 用 *Expr1* 乘以 π/200。

在 Degree 角度模式下 用 *Expr1* 乘以 g/100。

在 Gradian 模式下 原样返回 *Expr1*。

注意： 您可以通过在计算机键盘上键入 **@g** 插入此符号。

在 Degree Gradian 或 Radian 模式下：

$\cos(50^g)$	0.707107
$\cos(\{0,100^g,200^g\})$	{1.,0.,-1.}

r （ 弧度 ）

π▸ 键

*Value1*r ⇒ *值*

*List1*r ⇒ *数组*

*Matrix1*r ⇒ *矩阵*

此函数让您能够在 Degree 或 Gradian 模式下使用弧度角。

在 Degree 角度模式下 用自变量乘以 180/π。

在 Radian 模式下 原样返回自变量。

在 Gradian 模式下 用自变量乘以 200/π。

提示：如果您希望在使用函数时无论采用何种模式 均强制使用弧度角 可使用 r。

注意： 您可以通过在计算机键盘上键入 **@r** 插入此符号。

在 Degree Gradian 或 Radian 角度模式下：

$\cos\left(\frac{\pi}{4^r}\right)$	0.707107
$\cos\left(\left\{0^r,\left(\frac{\pi}{12}\right)^r,-(\pi)^r\right\}\right)$	{1.,0.965926,-1.}

° （ 度 ）

π 键

Value1° ⇒ *值*

List1° ⇒ *数组*

Matrix1° ⇒ *矩阵*

此函数让您能够在 Gradian 或 Radian 模式下使用度数角。

在 Radian 角度模式下　用自变量乘以 π/180。

在 Degree 模式下　原样返回自变量。

在 Gradian 角度模式下　用自变量乘以 10/9。

注意：您可以通过在计算机键盘上键入 **@d** 插入此符号。

在 Degree　Gradian 或 Radian 角度模式下：

$\cos(45°)$ → 0.707107

在 Radian 角度模式下：

$\cos\left\{\left\{0,\frac{\pi}{4},90°,30.12°\right\}\right\}$

$\{1.,0.707107,0.,0.864976\}$

°, ', ''（ 度 / 分 / 秒 ）

ctrl 键

dd°*mm*'*ss.ss*'' ⇒ *表达式*

dd　正数或负数
mm　非负数
ss.ss　非负数

返回 *dd*+(*mm*/60)+(*ss.ss*/3600)。

使用 -60 进制的输入格式　您可以：

- 以度 / 分 / 秒格式输入角度　而无需考虑当前角度模式。
- 以时 / 分 / 秒格式输入时间。

注意：ss.ss 后跟两个撇号 (' ') 而不是引号 (")。

在 Degree 角度模式下：

25°13'17.5" → 25.2215

25°30' → $\frac{51}{2}$

∠（ 角度 ）

ctrl 键

[*Radius*,∠θ*_Angle*] ⇒ *向量*
（ 极坐标输入 ）

[*Radius*,∠θ*_Angle*,*Z_Coordinate*] ⇒ *向量*
（ 圆柱坐标输入 ）

[*Radius*,∠θ*_Angle*,∠θ*_Angle*] ⇒ *向量*
（ 球坐标输入 ）

根据 Vector Format 模式设置以向量形式返回坐标：直角坐标　圆柱坐标　球坐标。

注意：您可以通过在计算机键盘上键入 **@<** 插入此符号。

在 Radian 模式和向量格式下设置为：
直角坐标

$[5 \quad \angle 60° \quad \angle 45°]$ → $[1.76777 \quad 3.06186 \quad 3.53553]$

圆柱坐标

$[5 \quad \angle 60° \quad \angle 45°]$ → $[3.53553 \quad \angle 1.0472 \quad 3.53553]$

球坐标

$[5 \quad \angle 60° \quad \angle 45°]$ → $[5. \quad \angle 1.0472 \quad \angle 0.785398]$

(*Magnitude* ∠ *Angle*) ⇒ *复数值*
（ 极坐标输入 ）

以 (r∠θ) 极坐标形式输入复数值。 *Angle* 将根据当前 Angle 模式设置显示。

在 Radian 角度模式和 Rectangular 复数格式下：

$5+3\cdot i-\left(10 \angle \frac{\pi}{4}\right)$ → $-2.07107-4.07107\cdot i$

_（ 下划线作为空元素 ）

请参阅 " 空（ 空值 ）元素 "（ 第 134 页 ）。

10^()

目录 >

10^ (*Value1*) ⟹ *数值*

10^ (*List1*) ⟹ *数组*

返回以 10 为底　自变量为乘方的计算结果。

对于数组　返回以 10 为底　以 *List1* 中各元素为乘方的计算结果。

$10^{1.5}$ → 31.6228

10^(*squareMatrix1*) ⟹ *方阵*

返回以 10 为底　*squareMatrix1* 为乘方的计算结果。此运算不同于计算以 10 为底　以方阵中各元素为乘方的值。有关计算方法的信息　请参阅 **cos()**。

squareMatrix1 必须可对角化　结果始终包含浮点数。

$$10^{\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}} \rightarrow \begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

^-1（ 倒数 ）

目录 >

Value1 **^-1** ⟹ *值*

List1 **^-1** ⟹ *数组*

返回自变量的倒数。

对于数组　返回 *List1* 中所有元素的倒数。

$(3.1)^{-1}$ → 0.322581

squareMatrix1 **^-1** ⟹ *方阵*

返回 *squareMatrix1* 的逆矩阵。

squareMatrix1 必须为非退化方阵。

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \rightarrow \begin{bmatrix}-2 & 1\\ \frac{3}{2} & -\frac{1}{2}\end{bmatrix}$$

|（ 约束运算符 ）

ctrl 键

表达式 **|** *布尔表达式 1* [**and** *布尔表达式 2*]...

表达式 **|** *布尔表达式 1* [**or** *布尔表达式 2*]...

约束符号 ("**|**") 表示二进制运算符。| 左侧的运算数是一个表达式。| 右侧的运算数指定了一个或多个影响表达式简化的关系。| 后的多个关系必须使用 "**and**" 或 "**or**" 逻辑运算符进行连接。

$x+1|x=3$ → 4

$x+55|x=\sin(55)$ → 54.0002

约束运算符有三种基本功能：

- 代换
- 区间约束
- 排除

代换是用等式的形式表示的　如 x=3 或 y=sin(x)。为有效起见　左侧应该是一个简单变量。*表达式* **|** *变量* **=** *值* 将代换 *表达式*中所有 *变量*的 *值*。

$x^3-2\cdot x+7\rightarrow f(x)$ → *Done*

$f(x)|x=\sqrt{3}$ → 8.73205

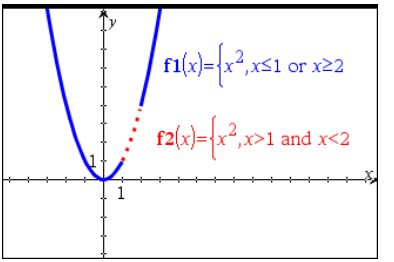
| (约束运算符)

ctrl 键

区间约束是用 "**and**" 或 "**or**" 逻辑运算符连接的一个或多个不等式。区间约束还允许简化　而在其他情况下简化可能无效或不可计算。

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x)$ 0.

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x) \mid x>0 \text{ and } x<5$ 3.



排除是使用 " 不等于 " (/= 或 ≠)关系运算符从对象中排除特定值。

→(存储)

ctrl var 键

Value → *Var*
List → *Var*
Matrix → *Var*
Expr → *Function*(*Param1*,...)
List → *Function*(*Param1*,...)
Matrix → *Function*(*Param1*,...)

如果变量 *Var* 不存在　则创建变量并将其赋值为 *Value* *List* 或 *Matrix*。

如果变量 *Var* 已存在且未被锁定或保护　则用 *Value*　*List* 或 *Matrix* 替换其值。

注意： 您可以通过在计算机键盘上键入 **=:** 来插入此运算符以作为快捷方式。例如　键入 **pi/4 =: myvar**。

$\frac{\pi}{4} \rightarrow myvar$ 0.785398

$2\cdot\cos(x) \rightarrow y1(x)$ *Done*

$\{1,2,3,4\} \rightarrow lst5$ $\{1,2,3,4\}$

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \rightarrow matg$ $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

"Hello" → *str1* "Hello"

:=(赋值)

ctrl 键

Var := *Value*
Var := *List*
Var := *Matrix*
Function(*Param1*,...):= *Expr*
Function(*Param1*,...):= *List*
Function(*Param1*,...):= *Matrix*

如果变量 *Var* 不存在　则创建 *Var* 并将其赋值为 *Value* *List* 或 *Matrix*。

如果变量 *Var* 已存在且未被锁定或保护　则用 *Value*　*List* 或 *Matrix* 替换其值。

$myvar := \frac{\pi}{4}$ .785398

$y1(x) := 2\cdot\cos(x)$ *Done*

$lst5 := \{1,2,3,4\}$ $\{1,2,3,4\}$

$matg := \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$ $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

str1 := "Hello" "Hello"

© (注释)

ctrl 键

© [*text*]

© 将*文本*作为注释行处理　可用于对所创建的函数和程序进行注释。

© 可位于行首或行间的任意位置。© 右侧直到该行结尾的所有内容均为注释。

输入示例时需注意的事项：在手持设备的 Calculator 应用程序中　您可以通过在每行结尾处按 ↵（ 而不是 enter ）输入多行定义。在计算机键盘上　按住 **Alt** 然后按 **Enter**。

Define $g(n)$=Func
© *Declare variables*
Local *i*,*result*
result:=0
For *i*,1,*n*,1 ©Loop *n times*
result:=*result*+i^2
EndFor
Return *result*
EndFunc

Done

$g(3)$ 14

0b，0h

0 B 键，0 H 键

0b *二进制数字*
0h *十六进制数字*

分别表示二进制或十六进制数值。要输入二进制或十六进制数值　在任何进位制模式下　您都必须输入 **0b** 或 **0h** 前缀。不带前缀的数值都将视为十进制（ 基数为 **10** ）处理。

结果根据进位制模式显示。

在 Dec 模式下：

0b10+0hF+10 27

在 Bin 模式下：

0b10+0hF+10 0b11011

在 Hex 模式下：

0b10+0hF+10 0h1B

空（ 空值 ）元素

分析真实数据时，您可能无法始终拥有完整的数据集。 TI-Nspire™ 软件允许空（ 或空值 ）数据元素，因此您可以处理近乎完整的数据，而不必重新开始或放弃不完整的情况。

您可以在 Lists & Spreadsheet 章节的 "*Graphing spreadsheet data*." 下找到涉及空元素的数据示例。

您可以通过 **delVoid()** 函数从列表中删除空元素。 **isVoid()** 函数让您能够检验空元素。有关详细信息，请参阅 **delVoid()**（ 第 29 页 ）和 **isVoid()**（ 第 49 页 ）。

注意：要在数学表达式中手动输入空元素，请键入 "_" 或关键字 **void**。计算表达式时，关键字 **void** 将自动转换为 "_" 符号。要在手持设备上键入 "_"，请按 ctrl ␣。

涉及空元素的计算

大多数涉及空值输入的计算都将生成空值结果。请参阅下面的特殊情况。

$\lvert\_\rvert$	_
$\gcd(100,\_)$	_
$3+\_$	_
$\{5,\_,10\}-\{3,6,9\}$	$\{2,\_,1\}$

包含空值元素的数组自变量

以下函数和命令会忽略（ 跳过 ）数组自变量中找到的空值元素。

count, countIf cumulativeSum freqTable▶list frequency max mean median product stDevPop stDevSamp sum sumIf varPop 和 **varSamp** 以及回归计算 **OneVar, TwoVar** 和 **FiveNumSummary** 统计 置信区间和统计检验

$\text{sum}(\{2,\_,3,5,6.6\})$	16.6
$\text{median}(\{1,2,\_,\_,\_,3\})$	2
$\text{cumulativeSum}(\{1,2,\_,4,5\})$	$\{1,3,\_,7,12\}$
$\text{cumulativeSum}\begin{bmatrix}1&2\\3&\_\\5&6\end{bmatrix}$	$\begin{bmatrix}1&2\\4&\_\\9&8\end{bmatrix}$

SortA 和 **SortD** 会将第一个自变量中的所有空值元素移至底部。

$\{5,4,3,\_,1\}\rightarrow list1$	$\{5,4,3,\_,1\}$
$\{5,4,3,2,1\}\rightarrow list2$	$\{5,4,3,2,1\}$
SortA *list1*,*list2*	*Done*
list1	$\{1,3,4,5,\_\}$
list2	$\{1,3,4,5,2\}$

$\{1,2,3,\_,5\}\rightarrow list1$	$\{1,2,3,\_,5\}$
$\{1,2,3,4,5\}\rightarrow list2$	$\{1,2,3,4,5\}$
SortD *list1*,*list2*	*Done*
list1	$\{5,3,2,1,\_\}$
list2	$\{5,3,2,1,4\}$

包含空值元素的数组自变量 (continued)

在回归中 **X** 或 **Y** 数组中的空值会引入残差对应元素的空值。

l1:={1,2,3,4,5}: *l2*:={2,_,3,5,6.6}	{2,_,3,5,6.6}
LinRegMx *l1*,*l2*	*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.}

回归中省略的类别会引入残差对应元素的空值。

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 *l1*,*l2*,1,*cat*,*incl*	*Done*
stat.Resid	{_,_,0.,0.}
stat.XReg	{_,_,4.,5.}
stat.YReg	{_,_,5.,6.6}
stat.FreqReg	{_,_,1.,1.}

回归中频率为 **0** 时会引入残差对应元素的空值。

l1:={1,3,4,5}: *l2*:={2,3,5,6.6}	{2,3,5,6.6}
LinRegMx *l1*,*l2*,{1,0,1,1}	*Done*
stat.Resid	{0.069231,_,-0.276923,0.207692}
stat.XReg	{1.,_,4.,5.}
stat.YReg	{2.,_,5.,6.6}
stat.FreqReg	{1.,_,1.,1.}

输入数学表达式的快捷方式

借助快捷方式，您可以通过输入数学表达式的元素，而无需使用 Catalog 或 Symbol Palette。例如，要输入表达式 √6，您可以在输入行中键入 **sqrt(6)**。按下 enter 时，表达式 **sqrt(6)** 将更改为 √6。一些快捷方式从手持设备和计算机键盘均可使用。另一些则主要从计算机键盘使用。

从手持设备或计算机键盘

要输入的内容：	键入的快捷方式：
π	**pi**
θ	**theta**
∞	**infinity**
≤	**<=**
≥	**>=**
≠	**/=**
⇒（ 逻辑隐含式 ）	**=>**
⇔（ 逻辑双隐含式，XNOR ）	**<=>**
→（ 存储运算符 ）	**=:**
||（ 绝对值 ）	**abs(…)**
√()	**sqrt(…)**
Σ()（ 求和模板 ）	**sumSeq(…)**
Π()（ 乘积模板 ）	**prodSeq(…)**
$\sin^{-1}()$, $\cos^{-1}()$, …	**arcsin(…), arccos(…), …**
ΔList()	**deltaList(…)**

从计算机键盘

要输入的内容：	键入的快捷方式：
i（ 虚数常数 ）	**@i**
e（ 以 e 为底的自然对数 ）	**@e**

要输入的内容：	键入的快捷方式：
E（ 科学计数法 ）	@E
T（ 转置 ）	@t
r（ 弧度 ）	@r
°（ 度 ）	@d
g（ 百分度 ）	@g
∠（ 角度 ）	@<
▶（ 转换 ）	@>
▶Decimal、 ▶approxFraction() 等。	@>Decimal、 @>approxFraction() 等。

EOS™ (Equation Operating System) 层次结构

本节介绍 TI-Nspire™ 数学及科学学习技术所采用的 Equation Operating System (EOS™)。数值、变量和函数均以简单、直接的顺序输入。EOS™ 软件可使用括号归组并根据下面介绍的属性计算表达式和方程。

计算顺序

级别	运算符
1	圆括号 **( )**、方括号 **[ ]**、花括号 **{ }**
2	间接 **(#)**
3	函数调用
4	后置运算符：度 - 分 - 秒 **(°,',")**、阶乘 **(!)**、百分比 **(%)**、弧度 **(r)**、下标 **([ ])**、转置 **(T)**
5	求幂、乘方运算符 **(^)**
6	求负 **(⁻)**
7	字符串联结 **(&)**
8	乘 **(•)**、除 **(/)**
9	加 **(+)**、减 **(-)**
10	相等关系：等于 **(=)**、不等于 **(≠ 或 /=)**， 小于 **(<)**、小于或等于（ ≤ 或 **<=** ）、大于 **(>)**、大于或等于（ ≥ 或 **>=** ）
11	逻辑 **not**
12	逻辑 **and**
13	逻辑 **or**
14	**xor**、**nor**、**nand**
15	逻辑隐含式 **(⇒)**
16	逻辑双隐含式，XNOR **(⇔)**
17	约束运算符 **("|")**
18	存储 **(→)**

圆括号、方括号和花括号

首先计算包含在圆括号、方括号或花括号内的所有计算。例如，在表达式 4(1+2) 中，EOS™ 软件首先计算表达式在圆括号内的部分（ 即 1+2 ），然后将结果 3 乘以 4.

表达式或方程内的左右圆括号、方括号和花括号数必须相同。否则会显示说明缺少元素的错误消息。例如， (1+2)/(3+4 将显示错误消息 “Missing).”

注意：由于 TI-Nspire™ 软件允许您定义自己的函数，因此如果变量名后跟包含在括号内的表达式，将被视为 “ 函数调用 ” 而不是隐含的乘法。例如，*a*(b+c) 是通过 b+c 求函数 *a* 的值。如果要将表达式 b+c 与变量 *a* 相乘，可使用显式乘法：*a*∗(b+c)。

间接运算

间接运算符 (#) 可将字符串转换为变量或函数名称。例如， #(“x”&”y”&”z”) 创建变量名称 xyz。间接运算还可以创建和修改程序内部的变量。例如，如果 10→r 且 “r”→s1，则 #s1=10。

后置运算符

后置运算符是直接跟在自变量之后的运算符，例如， 5!、 25% 或 60°15' 45"。后跟后置运算符的自变量以第四优先级进行计算。例如，在表达式 4^3! 中，首先计算 3!。结果 6，然后计算以 4 为底，以 6 为指数的值，得出 4096。

求幂

求幂 (^) 和逐个元素求幂 (.^) 均为自右至左进行计算。例如，表达式 2^3^2 与 2^(3^2) 计算得到的结果相同，都为 512。而 (2^3)^2 得到的结果则不同，是 64。

求负

要输入负数，请先按 ⁻ 然后输入数值。后置运算和求幂将在求负之前进行。例如， $-x^2$ 的结果为负数， $-9^2 = -81$。使用括号对负数求平方，例如 $(-9)^2$ 得到的结果为 81。

约束 （“|”）

约束运算符 (“|”) 后的自变量对运算符前的自变量计算有一系列的影响。

错误代码和消息

出现错误消息时，其代码将赋值给变量 *errCode*。用户定义的程序和函数可以检查 *errCode* 以确定出错的原因。有关使用 *errCode* 的示例，请参阅 **Try** 命令下的示例 2（ 第 106 页 ）。

注意：某些错误情况仅适用于 TI-Nspire™ CAS 产品，而另一些则适用于 TI-Nspire™ 产品。

错误代码	说明
10	函数未返回值
20	检验未解析为 TRUE 或 FALSE。 通常 未定义的变量无法进行比较。例如 如果 a 或 b 未定义 则在执行 If 语句时检验 If a<b 将导致此错误。
30	自变量不能是文件夹名称。
40	自变量错误
50	自变量不匹配 两个或多个自变量必须属于同一类型。
60	自变量必须是布尔表达式或整数
70	自变量必须是十进制数值
90	自变量必须是数组
100	自变量必须是矩阵
130	自变量必须是字符串
140	自变量必须是变量名称。 请确定名称满足以下要求： • 不以数字开头 • 不包含空格或特殊字符 • 不以无效方式使用下划线或句点 • 不超出长度限制 请参见文档中的 Calculator 一节 了解更多信息。
160	自变量必须是表达式
165	电池电量不足 无法发送或接收 发送或接收前安装新电池。
170	限值 下限必须小于上限才能定义搜索区间。
180	中断 在进行长时间运算或执行程序期间按了 esc 或 on 键。
190	循环定义 显示此消息可避免化简时无限替换变量值用尽内存。例如 a+1->a（ 其中 a 是未定义变量 ）将导致此错误。
200	限制条件表达式无效 例如 solve(3x^2-4=0,x) | x<0 或 x>5 将产生此错误消息 原因是限制条件以 “or” 分隔 而不是以 “and” 分隔。
210	数据类型无效 自变量的数据类型错误。

错误代码	说明
220	因变量受限
230	维数 数组或矩阵指数无效。例如　如果数组 {1,2,3,4} 存储在 L1 中　则 L1[5] 为维数错误　因为 L1 只包含四个元素。
235	维数错误。数组中元素数量不足。
240	维数不匹配 两个或多个自变量的维数必须相同。例如　[1,2]+[1,2,3] 的维数不匹配　因为两个矩阵包含的元素个数不同。
250	除数为零
260	域错误 自变量必须在指定域内。例如　**rand(0)** 无效。
270	变量名称重复
280	Else 和 ElseIf 在 If...EndIf 块外部无效
290	EndTry 缺少匹配的 Else 语句
295	迭代过度
300	数组或矩阵由预期的 2 个或 3 个元素组成
310	**nSolve** 的第一自变量必须是一元方程。不能包含除利率变量外的其他无值变量。
320	solve 或 cSolve 的第一自变量必须是方程或不等式 例如　solve(3x^2-4,x) 无效　因为第一自变量不是一个方程。
345	单位不一致
350	指数超出范围
360	间接字符串不是有效的变量名称
380	未定义 Ans 要么上一次计算未创建 Ans　要么之前未输入任何计算。
390	分配无效
400	赋值无效
410	命令无效
430	当前模式设置无效
435	估计值无效
440	隐含乘法无效 例如　x(x+1) 无效　而 x*(x+1) 是正确的句法。这样是为了避免混淆隐含乘法与函数调用。
450	在函数或当前表达式中无效 只有特定命令在用户定义的函数中有效。
490	在 Try..EndTry 块中无效
510	数组或矩阵无效
550	在函数或程序外部无效 有些命令在函数或程序外部无效。例如　**Local** 只能在函数或程序中使用。
560	在 Loop..EndLoop　For..EndFor 或 While..EndWhile 块外部无效 例如　Exit 命令仅在这些循环块内部有效。

错误代码	说明
565	在程序外部无效
570	路径名无效 例如　\var 无效。
575	复数极坐标无效
580	程序引用无效 程序不能在函数或表达式内引用（ 如 1+p(x)　其中 p 为程序 ）。
600	表格无效
605	单位使用无效
610	Local 语句中的变量名称无效
620	变量或函数名称无效
630	变量引用无效
640	向量句法无效
650	链接传输 两个设备之间的传输未完成。请确认连接电缆两端均已牢固连接。
665	矩阵不可对角化
670	内存不足 1. 删除本文档中的部分数据 2. 保存并关闭此文档 如果步骤 1 和 2 都无法完成　请取出电池然后重新插入
672	资源耗尽
673	资源耗尽
680	(缺失
690	) 缺失
700	" 缺失
710	] 缺失
720	} 缺失
730	块句法的开始或结束部分缺失
740	If..EndIf 块中缺少 Then
750	名称不是函数或程序
765	没有选择函数
780	找不到解
800	非实数结果 例如　如果软件使用 Real 设置　则 $\sqrt{(-1)}$ 无效。 要允许复数结果　请将 "Real or Complex" 模式设置更改为 RECTANGULAR 或 POLAR。
830	上溢
850	找不到程序 执行期间无法在提供的路径找到一个程序对于另一程序的引用。
855	绘图中不允许使用 Rand 类型函数

错误代码	说明
860	递归太深
870	名称或系统变量已保留
900	自变量错误 中位数 - 中位数模型无法应用到数据集。
910	句法错误
920	文本未找到
930	自变量过少 函数或命令缺少一个或多个自变量。
940	自变量过多 表达式或方程包含过多自变量且无法计算。
950	下标过多
955	未定义的变量过多
960	变量未定义 未向变量赋值。请使用以下命令之一： • sto → • := • **Define** 给变量赋值。
965	操作系统未获得许可
970	正在使用变量　因此不能被引用或更改
980	变量受保护
990	变量名称无效 请确定名称未超出长度限制
1000	窗口变量域
1010	缩放
1020	内部错误
1030	内存保护违规
1040	不支持的函数。此函数需要计算机代数系统。尝试使用 TI-Nspire™ CAS。
1045	不支持的运算符。此运算符需要计算机代数系统。尝试使用 TI-Nspire™ CAS。
1050	不支持的功能。此运算符需要计算机代数系统。尝试使用 TI-Nspire™ CAS。
1060	输入自变量必须是数值。仅允许输入数值。
1070	三角函数自变量过大　无法精确化简
1080	不支持使用 Ans。此应用程序不支持 Ans。
1090	函数未定义。请使用以下命令之一： • **Define** • := • sto → 以定义函数。
1100	非实数计算 例如　如果软件使用 Real 设置　则 $\sqrt{(-1)}$ 无效。 要允许复数结果　请将 "Real or Complex" 模式设置更改为 RECTANGULAR 或 POLAR。

错误代码	说明
1110	限值无效
1120	符号无变化
1130	自变量不能是数组或矩阵
1140	自变量错误 第一自变量必须是关于第二自变量的多项式表达式。如果缺少第二自变量　软件将尝试选择默认值。
1150	自变量错误 前两个自变量必须是关于第三个自变量的多项式表达式。如果缺少第三个自变量　软件将尝试选择默认值。
1160	库路径名称无效 路径名称必须使用 *xxx\yyy* 形式　其中： • *xxx* 部分可以是 1 到 16 个字符。 • *yyy* 部分可以是 1 到 15 个字符。 更多信息请参见文档中的 " 库 " 一节。
1170	库路径名称使用无效 • 不能使用 **Define**　**:=** 或 sto → 向路径名称赋值。 • 路径名称不能为 Local 变量　也不能作为参数在函数或程序定义中使用。
1180	库变量名称无效。 请确定名称满足以下要求： • 不包含句点 • 不以下划线开始 • 不超过 15 个字符 更多信息请参见文档中的 " 库 " 一节。
1190	未找到库文档： • 验证库位于 MyLib 文件夹中。 • 刷新库。 更多信息请参见文档中的 " 库 " 一节。
1200	未找到库变量： • 验证库变量位于库的第一个问题中。 • 请确定库变量定义为 LibPub 或 LibPriv。 • 刷新库。 更多信息请参见文档中的 " 库 " 一节。
1210	库快捷方式名称无效。 请确定名称满足以下要求： • 不包含句点 • 不以下划线开始 • 不超过 16 个字符 • 不是保留名称 更多信息请参见文档中的 " 库 " 一节。
1220	域错误： tangentLine 和 normalLine 函数仅支持实值函数。
1230	域错误。 Degree 或 Gradian 角度模式不支持三角转换运算符。
1250	自变量错误 使用线性方程组。 带变量 x 和 y 的二元线性方程组示例： 3x+7y=5 2y-5x=-1
1260	自变量错误： **nfMin** 或 **nfMax** 的第一自变量必须是单变量表达式。不能包含除利率变量外的其他无值变量。

错误代码	说明
1270	自变量错误 导数必须为 1 阶或 2 阶。
1280	自变量错误 请使用扩展式单变量多项式。
1290	自变量错误 请使用单变量多项式。
1300	自变量错误 多项式系数必须计算成数值。
1310	自变量错误： 函数的一个或多个自变量无法计算。
1380	自变量错误： 不允许嵌套调用 domain() 函数。

警告代码和消息

您可以使用 **warnCodes()** 函数存储通过计算表达式生成的警告代码。此表格列出每个数字警告代码及其关联的消息。

有关存储警告代码的示例，请参阅 **warnCodes()**（ 第 112 页 ）。

警告代码	消息
10000	进行运算可能会得到假解。
10001	求方程的微分可能会得到假方程。
10002	解可疑
10003	精确度可疑
10004	进行运算可能得不到解。
10005	cSolve 可能会指定更多零点。
10006	Solve 可能会指定更多零点。
10007	可能存在更多解。尝试指定合适的上下限和 / 或估计值。 使用 solve() 的示例： • solve（ 方程　变量 = 估计值 ）| 下界 < 变量 < 上界 • solve（ 方程　变量 ）| 下界 < 变量 < 上界 • solve（ 方程　变量 = 估计值 ）
10008	结果域可能比输入域小。
10009	结果域可能比输入域大。
10012	非实数计算
10013	∞^0 或 undef^0 被 1 取代
10014	undef^0 被 1 取代
10015	1^∞ 或 1^undef 被 1 取代
10016	1^undef 被 1 取代
10017	溢出被 ∞ 或 −∞ 取代
10018	运算需要 64 位值且返回 64 位值。
10019	资源耗尽　简化可能未完成。
10020	三角函数自变量过大　无法精确约简。
10021	输入中包含未定义参数。 结果可能并非对所有可能的参数值都有效。

警告代码	消息
10022	指定合适的上下边界可能可得到解。
10023	标量已乘以单位矩阵。
10024	使用近似计算获得结果。
10025	精确模式下无法验证是否相等。
10026	必须忽略限制条件。以 "\" ' 变量数学测试符号限制条件 ' 形式或这些形式的组合指定限制条件　例如　'x<3 and x>-12'

Texas Instruments 支持与服务

美国与加拿大：

一般信息

主页：	education.ti.com
基础知识与电子邮件咨询：	education.ti.com/support
电话：	(800) TI-CARES / (800) 842-2737 仅适用于美国、加拿大、墨西哥、波多黎各以及维京岛
国际信息：	education.ti.com/international

技术支持

基础知识及电子邮件支持：-	education.ti.com/support
电话（收费）：	(972) 917-8324

产品（硬件）服务

美国、加拿大、墨西哥、波多黎各以及维京岛的客户 请在返修产品前联系 Texas Instruments 客户支持部门。

其他国家：

一般信息

有关 TI 产品和服务的更多信息，请通过电子邮件与 TI 联系或访问 TI 网址。

电子邮件咨询：	ti-cares@ti.com
主页：	education.ti.com

维修和保修信息

关于保修期限和条款，及产品维修的信息，请参阅本产品附带的保修声明，或者联系当地的 Texas Instruments 零售商 / 分销商。

索引

Symbols

Numerics

A

B

C

D

E

F

G

H

J

K

L

M

S

T

W

X

Y

Z