TI-Nspire™
Referenshandbok

Mer information om TI-teknologi hittar du i online-hjälpen på education.ti.com/eguide.

Viktigt information

Med undantag för vad som uttryckligen anges i den licens som medföljer ett program lämnar Texas Instruments inga garantier, vare sig uttryckliga eller underförstådda, inklusive garantier avseende säljbarhet eller lämplighet för visst ändamål beträffande något program- eller bokmaterial, och tillhandahåller sådant material "i befintligt skick". Under inga omständigheter skall Texas Instruments hållas ansvarigt för några speciella, indirekta eller tillfälliga skador eller följdskador i samband med inköpet eller användningen av materialet, och Texas Instruments:s enda och uteslutande skadeståndsskyldighet, oberoende av anspråkets form, skall inte överstiga det belopp som anges i licensen för programmet. Inte heller skall Texas Instruments hållas ansvarigt för anspråk av något som helst slag beträffande användningen av materialet av annan part.

© 2006 - 2019 Texas Instruments Incorporated

Innehåll

Mallar för uttryck **1**

Alfabetisk lista **7**

A 7
B 15
C 19
D 35
E 44
F 52
G 59
I 69
L 77
M 91
N 99
O 108
P 110
Q 117
R 120
S 134
T 153
U 166
V 166
W 167
X 170
Z 171

Symboler **177**

Tomma element **200**

Kortkommandon för att mata in matematiska uttryck **202**

EOS™-hierarki (Equation Operating System) **204**

Konstanter och värden **206**

Felkoder och meddelanden **207**

Varningskoder och meddelanden **216**

Allmän information **218**

Hjälp-funktion online 218

Kontakta TI support 218
Service- och garanti-information 218

Innehållsförteckning 219

Mallar för uttryck

Mallar för uttryck erbjuder ett enkelt sätt att mata skriva in uttryck med matematiska standardtecken. När du matar in en mall visas den på inmatningsraden med små block i positioner där du kan skriva in element. En markör visar vilket element du kan skriva in.

Använd piltangenterna eller tryck på [tab] för att flytta till varje elements position, och skriv ett värde eller uttryck för det aktuella elementet. Tryck på [enter] eller [ctrl] [enter] för att utvärdera uttrycket.

Mall för Bråk

[ctrl] [÷] tangenter

$\frac{\square}{\square}$

Obs: Se även **/ (dela)**, på sidan 179.

Exempel:

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

Mall för Exponent

[^] tangent

$\square^{\square}$

Obs: Skriv in det första värdet, tryck på [^] och skriv sedan in exponenten. För att återföra markören till basraden, tryck på högerpilen (▶).

Obs: Se även **^ (potens)**, på sidan 180.

Exempel:

2^3	8

Mall för Kvadratrot

[ctrl] [x^2] tangenter

$\sqrt{\square}$ **Obs:** Se även **√() (kvadratrot)**, på sidan 189.

Exempel:

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

Mall för N:te rot

[ctrl] [^] tangenter

$\sqrt[\square]{\square}$ **Obs:** Se även **root()**, på sidan 131.

Exempel:

Mall för N:te rot

ctrl ^ tangenter

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

e exponent mall

e^x tangent

$e^{\square}$

Basen för den naturliga logaritmen *e* upphöjd till

Obs: Se även **e^()**, på sidan 44.

Exempel:

e^{1}	2.71828182846

Mall för Log

ctrl 10^x tangenter

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

Beräknar logaritmen till en specificerad bas. För en förinställning av bas 10, utelämna basen.

Obs: Se även **log()**, på sidan 87.

Exempel:

$\log_{4}(2.)$	0.5

Stegvis mall (2 steg)

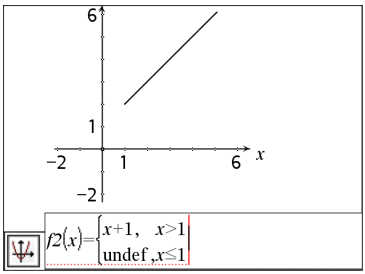
Katalog >

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

Låter dig skapa uttryck och villkor för en stegvis funktion med två steg.- För att lägga till ett steg, klicka i mallen och upprepa mallen.

Obs: Se även **stegvis()**, på sidan 112.

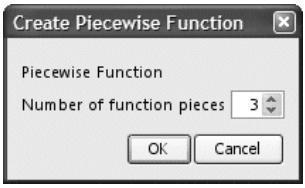
Exempel:



Stegvis mall (N steg)

Katalog >

Låter dig skapa uttryck och villkor för en stegvis funktion med N steg.- Promptar för N.



Obs: Se även **stegvis()**, på sidan 112.

Exempel:

Se exemplet på Stegvis mall (2 steg).

Mall för System med 2 ekvationer

Katalog >

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

Skapar ett ekvationssystem med två linjära ekvationer. För att lägga till en rad i ett befintligt system, klicka i mallen och upprepa mallen.

Obs: Se även **system()**, på sidan 153.

Exempel:

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

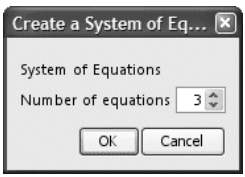
$$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$$

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

Mall för System med N ekvationer

Katalog >

Låter dig skapa ett ekvationssystem med Nlinjära ekvationer. Promptar för N.



Obs: Se även **system()**, på sidan 153.

Exempel:

Se exemplet på mall för Ekvationssystem (2 ekvationer).

Mall för Absolutbelopp

Katalog >

$|\square|$ **Obs:** Se även **abs()**, på sidan 7.

Exempel:

Mall för Absolutbelopp

Katalog >

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

Mall för dd°mm'ss.ss''

Katalog >

□°□'□"

Låter dig skriva in vinklar i formatet **dd°mm'ss.ss''**, där **dd** är antalet decimala grader, **mm** är antalet minuter och **ss.ss** är antalet sekunder.

Exempel:

$$30°15'10" \qquad 0.528011$$

Matrismall (2 x 2)

Katalog >

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

Skapar en 2 x 2-matris.

Exempel:

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

Matrismall (1 x 2)

Katalog >

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

Exempel:

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

Matrismall (2 x 1)

Katalog >

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

Exempel:

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

Matrismall (m x n)

Katalog >

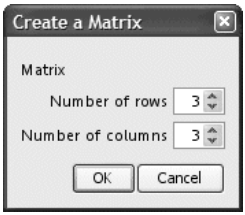
Mallen visas när du har uppmanats att specificera antalet rader och kolumner.

Exempel:

$$\text{diag}\left(\begin{bmatrix} 4 & 2 & 6 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}\right) \qquad \begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$$

Matrismall (m x n)

Katalog >



Obs: Om du skapar en matris med många rader och kolumner kan det ta några sekunder innan den visas.

Mall för Summa (Σ)

Katalog >

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

Obs: Se även **Σ() (sumSeq)**, på sidan 190.

Exempel:

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

Mall för Produkt (Π)

Katalog >

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

Obs: Se även **Π() (prodSeq)**, på sidan 189.

Exempel:

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

Mall för förstaderivata

Katalog >

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

Mallen för förstaderivata kan användas för att numeriskt beräkna förstaderivatan i en punkt med automatiska deriveringsmetoder.

Obs: Se även **d() (derivata)**, på sidan 188.

Exempel:

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

Andraderivata, mall

Katalog >

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

Mallen för andraderivata kan användas för att numeriskt beräkna andraderivatan i en punkt med automatiska deriveringsmetoder.

Obs: Se även **d() (derivata)**, på sidan 188.

Exempel:

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

Mall för Bestämd integral

Katalog >

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

Mallen för bestämd integral kan användas för att numeriskt beräkna den bestämda integralen med samma metod som nInt().

Obs: Se även **nInt()**, på sidan 103.

Exempel:

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

Alfabetisk lista

Poster som inte är alfabetiska (till exempel, +, ! och >) listas i slutet av detta avsnitt och börjar, på sidan 177. Om inget annat anges har alla exempel i detta avsnitt utförts i det förinställda återställningsläget och alla variabler betraktas som odefinierade.

A

abs() Katalog >

abs(*Value1***)⇒***värde*

abs(*List1***)⇒***lista*

abs(*Matrix1***)⇒***matris*

Ger argumentets absolutbelopp.

Obs: Se även **Mall för Absolutbelopp**, på sidan 3.

Om argumentet är ett komplext tal erhålls talets modul.

Obs: Alla odefinierade variabler behandlas som reella variabler.

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

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

amortTbl() Katalog >

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

Amorteringsfunktion som ger en matris i form av en amorteringstabell för en uppsättning av TVM-argument.

NPmt är antalet inbetalningar som skall inkluderas i tabellen. Tabellen börjar med den första inbetalningen.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 163.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och

amortTbl(12,60,10,5000,,,12,12)

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

PmtAt är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

Kolumnerna i resultatmatrisen har följande ordning: Inbetalningsnummer, räntebelopp, kapitalbelopp och balans.

Balansen som visas på rad *n* är balansen efter inbetalning *n*.

Du kan använda resultatmatrisen som indata för de andra amorteringsfunktionerna **ΣInt()** och **ΣPrn()**, se på sidan 191, och **bal()**, se på sidan 15.

and (och)

BooleanExpr1 **and** *BooleanExpr2*⇒*Booleskt uttryck*

BooleanList1 **and** *BooleanList2*⇒*Boolesk lista*

BooleanMatrix1 **and** *BooleanMatrix2*⇒*Boolesk matris*

Ger resultatet sant eller falskt eller en förenklad form av den ursprungliga inmatningen.

Integer1 **and** *Integer2*⇒*heltal*

Jämför två reella heltal bit för bit med en **och**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1, annars blir resultatet 0. Det erhållna värdet representerar bitresultatet och visas enligt Bas-läget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

I hexadecimalt basläge:

0h7AC36 and 0h3D5F	0h2C16

Viktigt: Noll, inte bokstaven O.

I binärt basläge:

0b100101 and 0b100	0b100

I decimalt basläge:

37 and 0b100	4

and (och) Katalog >

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk moduloberäkning för att få ned värdet till lämplig nivå.

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

angle() Katalog >

angle(*Value1***)⇒***värde*

Ger argumentets vinkel med argumentet tolkat som ett komplext tal.

I vinkelläget Grader:

$\text{angle}(0+2\cdot i)$	90

I vinkelläget Nygrader:

$\text{angle}(0+3\cdot i)$	100

I vinkelläget Radianer:

$\text{angle}(1+i)$	0.785398
$\text{angle}(\{1+2\cdot i,3+0\cdot i,0-4\cdot i\})$	$\{1.10715,0.,-1.5708\}$

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

vinkel(*List1***)⇒***lista*

vinkel(*Matrix1***)⇒***matris*

Ger en lista eller matris över vinklarna hos elementen i *List1* eller *Matrix1*, där varje element tolkas som ett komplext tal som representerar en tvådimensionell rektangulär koordinatpunkt.

ANOVA Katalog >

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

Utför en 1-vägs variansanalys för att jämföra medelvärdena hos 2 till 20 populationer. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Flag=0 för Data, *Flag*=1 för Statistik

Resultatvariabel	Beskrivning
stat.F	Värdet på F-statistiken
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos grupperna
stat.SS	Kvadratsumma hos grupperna
stat.MS	Kvadratmedelvärde hos grupperna
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
stat.sp	Sammanslagen (pooled) standardavvikelse
stat.xbarlist	Medelvärdet på listornas indata
stat.CLowerList	95 % konfidensintervall för medelvärdet hos varje indatalista
stat.CUpperList	95 % konfidensintervall för medelvärdet hos varje indatalista

ANOVA2way

Katalog >

ANOVA 2-vägs*Lista1*,*Lista2* [,*Lista3*,...,*Lista10*][,*LevRow*]

Beräknar en 2-vägs variansanalys för att jämföra medelvärdena hos 2 till 10 populationer. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

NivRad=0 för Block

NivRad=2,3,...,*Län*-1, för Två Faktorer, där *Län*=längd(*Lista1*)=längd(*Lista2*) = ... = längd(*Lista10*) och *Län* / *NivRad* ∈{2,3,...}

Utdata: Block Design

Resultatvariabel	Beskrivning
stat.F	F statistik för kolumnfaktorn
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos kolumnfaktorn
stat.SS	Kvadratsumma hos kolumnfaktorn

Resultatvariabel	Beskrivning
stat.MS	Kvadratmedelvärde hos kolumnfaktorn
Statistik.FBlock	F statistik för faktor
stat.PValBlock	Lägsta sannolikhet vid vilken nollhypotesen kan förkastas
stat.dfBlock	Frihetsgrader hos faktor
stat.SSBlock	Kvadratsumma hos faktor
stat.MSBlock	Kvadratmedelvärde hos faktor
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
stat.s	Standardavvikelse hos felet

Utdata för KOLUMNFAKTOR

Resultatvariabel	Beskrivning
stat.Fcol	F statistik för kolumnfaktorn
stat.PValCol	Sannolikhetsvärde på kolumnfaktorn
stat.dfCol	Frihetsgrader hos kolumnfaktorn
stat.SSCol	Kvadratsumma hos kolumnfaktorn
stat.MSCol	Kvadratmedelvärde hos kolumnfaktorn

Utdata för RADFAKTOR

Resultatvariabel	Beskrivning
stat.FRow	F statistik för radfaktorn
stat.PValRow	Sannolikhetsvärde på radfaktorn
stat.dfRow	Frihetsgrader hos radfaktorn
stat.SSRow	Kvadratsumma hos radfaktorn
stat.MSRow	Kvadratmedelvärde hos radfaktorn

Utdata för INTERAKTION

Resultatvariabel	Beskrivning
stat.FInteract	F statistik för interaktionen

Resultatvariabel	Beskrivning
stat.PValInteract	Sannolikhetsvärde på interaktionen
stat.dfInteract	Frihetsgrader hos interaktionen
stat.SSInteract	Kvadratsumma hos interaktionen
stat.MSInteract	Kvadratmedelvärde hos interaktionen

Utdata för FEL

Resultatvariabel	Beskrivning
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
s	Standardavvikelse hos felet

Ans (svar)

ctrl (–) tangenter

Ans⇒*värde*

Ger resultatet på det senast beräknade uttrycket.

56	56
56+4	60
60+4	64

approx()

Katalog >

approx(*Value1***)**⇒*tal*

Visar resultatet av beräkningen av argumentet som ett uttryck med decimala värden, när så är möjligt, oavsett den aktuella inställningen av **Auto eller Ungefärlig**.

Detta motsvarar att skriva in argumentet och trycka på ctrl enter.

$\text{approx}\left(\frac{1}{3}\right)$	0.333333
$\text{approx}\left(\left\{\frac{1}{3},\frac{1}{9}\right\}\right)$	{0.333333,0.111111}
$\text{approx}(\{\sin(\pi),\cos(\pi)\})$	{0.,-1.}
$\text{approx}([\sqrt{2}\quad\sqrt{3}])$	[1.41421 1.73205]
$\text{approx}\left(\left[\frac{1}{3}\quad\frac{1}{9}\right]\right)$	[0.333333 0.111111]

approx(*List1***)**⇒*lista*

approx(*Matrix1***)**⇒*matris*

Ger en lista eller *matris* där varje element har beräknats till ett decimalt värde, när så är möjligt.

$\text{approx}(\{\sin(\pi),\cos(\pi)\})$	{0.,-1.}
$\text{approx}([\sqrt{2}\quad\sqrt{3}])$	[1.41421 1.73205]

▸approxFraction() Katalog >

Value ▸approxFraction([*Tol*]**)**⇒*värde*

List **▸approxFraction(**[*Tol*]**)**⇒*lista*

Matrix **▸approxFraction(**[*Tol*]**)**⇒*matris*

Ger indata som ett bråk med hjälp av toleransen hos *Tol*. Om *Tol* utelämnas används en tolerans på 5.E-14.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `@>approxFraction(...)`.

$\frac{1}{2}+\frac{1}{3}+\tan(\pi)$ 0.833333

0.83333333333333▸approxFraction(5.E⁻14)

$\frac{5}{6}$

{π,1.5}▸approxFraction(5.E⁻14)

$\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

approxRational() Katalog >

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

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

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

Ger argumentet som ett bråk med hjälp av toleransen hos *Tol*. Om *Tol* utelämnas används en tolerans på 5.E-14.

approxRational$(0.333,5\cdot10^{-5})$ $\frac{333}{1000}$

approxRational$(\{0.2,0.33,4.125\},5.E^{-14})$

$\left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

arccos() Se cos⁻¹(), på sidan 26.

arccosh() Se cosh⁻¹(), på sidan 28.

arccot() Se cot⁻¹(), på sidan 29.

arccoth() Se coth⁻¹(), på sidan 29.

arccsc() Se csc⁻¹(), på sidan 32.

arccsch() Se csch^{-1}(), på sidan 33.

arcsec() Se sec^{-1}(), på sidan 134.

arcsech() Se sech^{-1}(), på sidan 135.

arcsin() Se sin^{-1}(), på sidan 143.

arcsinh() Se sinh^{-1}(), på sidan 144.

arctan() Se tan^{-1}(), på sidan 154.

arctanh() Se tanh^{-1}(), på sidan 155.

augment() Katalog >

augment(*List1, List2***)⇒***lista*

Ger en ny lista med *List2* inlagd i slutet på *List1*.

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

augment(*Matrix1***, *Matrix2***)⇒***matris*

Ger en ny matris med *Matrix2* fogad till *Matrix1*. När kommatecknet (,) används måste matriserna ha samma raddimensioner och *Matrix2* fogas till *Matrix1* som nya kolumner. Ändrar inte *Matrix1* eller *Matrix2*.

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

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

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

avgRC() Katalog >

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

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

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

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

Ger differenskvoten i positiv riktning.

Expr1 kan vara ett användardefinierat funktionsnamn (se **Func**).

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Step är stegvärdet. Om *Step* utelämnas används förinställningen 0.001.

Observera att den liknande funktionen **centralDiff()** använder den symmetriska differenskvoten.

$x:=2$	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() Katalog >

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

bal(*NPmt*,*amortTable***)**⇒*värde*

Amorteringsfunktion som beräknar planerad balans efter en specificerad inbetalning.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 163.

NPmt anger numret på den inbetalning efter vilken du vill att data skall beräknas.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 163.

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

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

$$\begin{bmatrix} 0 & 0. & 0. & 5000. \\ 1 & -23.35 & -825.63 & 4174.37 \\ 2 & -19.49 & -829.49 & 3344.88 \\ 3 & -15.62 & -833.36 & 2511.52 \\ 4 & -11.73 & -837.25 & 1674.27 \\ 5 & -7.82 & -841.16 & 833.11 \\ 6 & -3.89 & -845.09 & -11.98 \end{bmatrix}$$

$\text{bal}(4,tbl)$	1674.27

bal() Katalog >

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

bal(*NPmt,amortTable***)** beräknar lånebalansen efter inbetalning nummer *NPmt*, baserat på amorteringstabell *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 7.

Obs: Se även **ΣInt()** och **ΣPrn()**, på sidan 191.

▸Base2 Katalog >

Integer1 **▸Base2**⇒*heltal*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Base2`**.

Omvandlar *Integer1* till ett binärt tal. Binära och hexadecimala tal har alltid prefixet 0b respektive 0h. Noll, inte bokstaven O, följt av b eller h.

0b *binärtTal*

0h *hexadecimaltTal*

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

Utan prefix behandlas *Integer1* som ett decimalt tal (bas 10). Resultatet visas i binär form, oavsett Bas-läget.

Negativa tal visas i “tvåkomplement”-form. Exempel,

256▸Base2	0b100000000
0h1F▸Base2	0b11111

▶Base2 Katalog >

−1 visas som 0hFFFFFFFFFFFFFFFF i Hexadecimalt basläge 0b111...111 (64 1's) i Binärt basläge

-2^{63} visas som 0h8000000000000000 i Hexadecimalt basläge 0b100...000 (63 zeros) i Binärt basläge

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. Se följande exempel på värden utanför området.

2^{63} blir -2^{63} och visas som 0h8000000000000000 i Hexadecimalt basläge 0b100...000 (63 nollor) i Binärt basläge

2^{64} blir 0 och visas som 0h0 i Hexadecimalt basläge 0b0 i Binärt basläge

$-2^{63} - 1$ blir $2^{63} - 1$ och visas som 0h7FFFFFFFFFFFFFFF i Hexadecimalt basläge 0b111...111 (64 ettor) i Binärt basläge

▶Base10 Katalog >

Integer1 ▶**Base10**⇒*heltal*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@>Base10`.

Omvandlar *Integer1* till ett decimalt tal (bas 10). En binär eller hexadecimal inmatning måste alltid ha prefixet 0b respektive 0h.

0b *binaryNumber*

0h *hexadecimalNumber*

Noll, inte bokstaven O, följt av b eller h.

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

0b10011▶Base10	19
0h1F▶Base10	31

▶Base10 Katalog >

Utan prefix behandlas *Integer1* som ett decimalt tal. Resultatet visas i decimal form, oavsett Bas-läget.

▶Base16 Katalog >

Integer1 ▶**Base16**⇒*heltal*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Base16**.

Konverterar *Integer1* till ett hexadecimalt tal. Binära och hexadecimala tal har alltid prefixet 0b respektive 0h.

0b *binaryNumber*

0h *hexadecimalNumber*

Noll, inte bokstaven O, följt av b eller h.

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

Utan prefix behandlas *Integer1* som ett decimalt tal (bas 10). Resultatet visas i hexadecimal form, oavsett Bas-läget.

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▶**Base2**, på sidan 16.

256▶Base16	0h100
0b111100001111▶Base16	0hF0F

binomCdf() Katalog >

binomCdf(*n***,***p***)**⇒*lista*

binomCdf(*n***,***p***,***lowBound***,***upBound***)**⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

binomCdf(*n***,***p***,***upBound***)**för P(0≤X≤*upBound*) ⇒*tal* om *upBound* är ett tal, *lista* om *upBound* är en lista

binomCdf() Katalog >

Beräknar en kumulativ sannolikhet för den diskreta binomialfördelningen med *n* antal försök och sannolikheten *p* för att lyckas vid varje försök.

För P(X ≤ *upBound*), sätt *lowBound*=0

binomPdf() Katalog >

binomPdf(*n***,***p***)**⇒*lista*

binomPdf(*n***,***p***,***XVal***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar en sannolikhet för den diskreta binomialfördelningen med *n* antal försök och sannolikheten *p* för att lyckas vid varje försök.

C

ceiling() Katalog >

ceiling(*Value1***)**⇒*värde*

Ger det närmaste heltal som är ≥ argumentet.

Argumentet kan vara ett reellt eller ett komplext tal.

Obs: Se även **floor()**.

ceiling(.456)	1.

ceiling(*List1***)**⇒*lista*

ceiling(*Matrix1***)**⇒*matris*

Ger en lista eller matris över taket för varje element.

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

centralDiff() Katalog >

centralDiff(*Uttr1***,***Var* [=*Värde*][,*Steg*]**)** ⇒*uttryck*

centralDiff(*Uttr1***,***Var* [,*Steg*]**)** **|***Var*=*Värde*⇒*uttryck*

centralDiff(*Uttr1***,***Var* [=*Värde*][,*Lista*]**)**

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

centralDiff() Katalog >

⇒*lista*

centralDiff(*Lista1***,***Var* [=*Värde*][,*Steg*]**)**
⇒*lista*

centralDiff(*Matris1***,***Var* [=*Värde*][,*Steg*]**)**
⇒*matris*

Ger den numeriska derivatan genom att använda formeln för symmetrisk differenskvot.

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Steg är stegvärdet. Om *Steg* utelämnas används förinställningen 0.001.

När du använder *Lista1* eller *Matris1* utförs operationen på värdena i listan eller matriselementen.

Obs: Se även **avgRC()**.

char() Katalog >

char(*Integer***)**⇒*tecken*

Ger en teckensträng som innehåller tecknet med numret *Integer* från handenhetens teckenuppsättning. Det giltiga området för *Integer* är 0–65535.

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

$\chi^2$2way Katalog >

χ^2**2way** *ObsMatrix*

chi22way *ObsMatrix*

Beräknar ett χ^2-test för association på 2-vägstabellen över antal i den observerade matrisen *ObsMatrix*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

För information om effekten av tomma element i en matris, se "Tomma element" (på sidan 200).

Resultatvariabel	Beskrivning
stat.χ^2	Chi-kvadratstatistik: summa (observerad - förväntad)^2/förväntad
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos chi-kvadratstatistiken
stat.ExpMat	Matris över förväntad elementräknetabell, baserad på nollhypotesen
stat.CompMat	Matris över elementbidrag till chi-kvadratstatistiken

χ^2Cdf() Katalog >

χ^2Cdf(*lowBound*,*upBound*,*df*)⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

chi2Cdf(*lowBound*,*upBound*,*df*)⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten för χ^2-fördelning mellan *lowBound* och *upBound* för den specificerade frihetsgraden *df*.

För P(*X upBound*), sätt *lowBound* = 0.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

χ^2GOF Katalog >

χ^2GOF *obsList*,*expList*,*df*

chi2GOF *obsList*,*expList*,*df*

Utför ett test för att bekräfta att urvalsdata är från en population som följer en specificerad fördelning. *obsList* är en lista med data och måste innehålla heltal. En sammanfattning av resultaten visas i variabeln *stat.results*, på sidan 148.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.χ^2	Chi-kvadratstatistik: summa (observerad - förväntad)^2/förväntad
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos chi-kvadratstatistiken
stat.CompList	Elementbidrag till chi-kvadratstatistiken

χ^2Pdf() Katalog >

χ^2Pdf(*XVal***,***df***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

chi2Pdf(*XVal***,***df***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen (pdf) för χ^2-fördelningen vid ett specificerat *XVal*-värde för den specificerade frihetsgraden *df*.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

ClearAZ Katalog >

ClearAZ

Rensar alla variabler som har ett enda tecken i det aktuella problemet.

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 166.

5→*b*	5
b	5
ClearAZ	*Done*
b	"Error: Variable is not defined"

ClrErr Katalog >

ClrErr

Rensar felstatusen och ställer in systemvariabeln *errCode* på noll.

För ett exempel på **ClrErr**, se exempel 2 under kommandot **Try**, på sidan 160.

ClrErr

Katalog >

Villkoret **Else** i blocket **Try...Else...EndTry** bör använda **ClrErr** eller **PassErr**. Om felet skall processas eller ignoreras, använd **ClrErr**. Om det är okänt hur felet skall hanteras, använd **PassErr** för att skicka felet vidare till nästa felhanterare. Om det inte finns någon ytterligare felhanterare för **Try...Else...EndTry** visas feldialogrutan som normal.

Obs: Se även **PassErr**, på sidan 111 och **Try**, på sidan 159.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

colAugment()

Katalog >

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

Ger en ny matris med *Matrix2* fogad till *Matrix1*. Matriserna måste ha samma kolumndimensioner och *Matrix2* fogas till *Matrix1* som nya rader. Ändrar inte *Matrix1* eller *Matrix2*.

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

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

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

colDim()

Katalog >

colDim(*Matrix***)⇒*uttryck***

Ger antalet kolumner i *Matrix*.

Obs: Se även **rowDim()**.

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

colNorm()

Katalog >

colNorm(*Matrix***)⇒*uttryck***

Ger maximum av summorna av absolutbeloppen på elementen i kolumnerna i *Matrix*.

Obs: Odefinierade matriselement är ej tillåtna. Se även **rowNorm()**.

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

$$\text{colNorm}(mat) \qquad 9$$

conj() Katalog >

conj(*Value1***)⇒***värde*

conj(*List1***)⇒***lista*

conj(*Matrix1***)⇒***matris*

Ger argumentets komplexkonjugat.

Obs: Alla odefinierade variabler behandlas som reella variabler.

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

constructMat() Katalog >

constructMat (*Expr***,***Var1***,***Var2***,***numRows***,***numCols***)** ⇒*matris*

Ger en matris baserad på argumenten.

Expr är ett uttryck i variablerna *Var1* och *Var2*. Element i den resulterande matrisen skapas genom att utvärdera *Expr* för varje ökat värde på *Var1* och *Var2*.

Var1 ökas automatiskt från **1** till och med *numRows*. Inom varje rad ökas *Var2* från **1** till och med *numCols*.

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

CopyVar Katalog >

CopyVar *Var1***,** *Var2*

CopyVar *Var1***.,** *Var2***.**

CopyVar *Var1, Var2* kopierar värdet på variabel *Var1* till variabel *Var2*, och skapar *Var2* vid behov. Variabeln *Var1* måste ha ett värde.

Om *Var1* är namnet på en befintlig användardefinierad funktion kopieras definitionen på denna funktion till funktionen *Var2*. Funktionen *Var1* måste vara definierad.

Var1 måste uppfylla kraven för namngivning av variabler eller måste vara ett indirection-uttryck som förenklas till ett variabelnamn som uppfyller kraven.

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

Katalog >

CopyVar *Var1*., *Var2*. kopierar alla led i variabelgruppen *Var1*. till gruppen *Var2*. och skapar *Var2*. vid behov.

Var1. måste vara namnet på en befintlig variabelgrupp, till exempel, den statistiska *stat.nn*-resultat eller variabler skapade med funktionen **LibShortcut()**. Om *Var2*. redan finns ersätter detta kommando alla led som är gemensamma för båda grupperna och lägger till de led som inte redan finns. Om en eller flera medlemmar av *Var2*. är låsta lämnas alla medlemmar av *Var2*. oförändrade.

$aa.a:=45$	45
$aa.b:=6.78$	6.78
CopyVar $aa.,bb.$	*Done*
getVarInfo()	$\begin{bmatrix} aa.a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ aa.b & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ bb.a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ bb.b & \text{"NUM"} & \text{"}\square\text{"} & 0 \end{bmatrix}$

corrMat()

Katalog >

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

Beräknar korrelationsmatrisen för den sammanfogade matrisen [*List1, List2*, ..., *List20*].

cos()

trig **tangent**

cos(*Value1***)⇒***värde*

cos(*List1***)⇒***lista*

cos(*Value1***)** ger argumentets cosinus som ett värde.

cos(*List1***)** ger en lista på cosinus för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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

cos(*squareMatrix1***)⇒***kvadratMatris*

I vinkelläget Radianer:

cos() trig tangent

Ger matrisen med cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna cosinus för varje element.

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

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

När en skalär funktion f(A) används på *squareMatrix1* (A) beräknas resultatet med algoritmen:

Beräkna egenvärdena (λ_i) och egenvektorerna (V_i) för A.

squareMatrix1 måste vara möjlig att diagonalisera. Den får inte heller ha symboliska variabler som inte har tilldelats ett värde.

Forma matriserna:

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

Då är $A = X\,B\,X^{-1}$ och $f(A) = X\,f(B)\,X^{-1}$. Exempelvis $\cos(A) = X\cos(B)\,X^{-1}$ där:

cos(B) =

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

Alla beräkningar utförs med flyttalsaritmetik.

cos⁻¹() trig tangent

cos⁻¹(*Value1***)⇒***värde*

cos⁻¹(*List1***)⇒***lista*

cos⁻¹(*Value1***)** ger den vinkel vars cosinus är *Value1*.

cos⁻¹(*List1***)** ger en lista på invers cosinus för varje element i *List1*.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

$\cos^{-1}()$ trig tangent

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

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

$\{1.5708,1.36944,1.0472\}$

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arccos(...)`.

$\cos^{-1}$(*squareMatrix1***)⇒***squareMatrix***

Ger matrisen med invers cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

cosh() Katalog >

cosh(*Value1***)⇒***värde***

cosh(*List1***)⇒***lista***

cosh(*Value1***)** ger argumentets hyperboliska cosinus.

cosh(*List1*) ger en lista på hyperbolisk cosinus för varje element i *List1*.

I vinkelläget Grader:

$$\cosh\left(\left(\frac{\pi}{4}\right)^{r}\right) \qquad 1.74671\text{E}19$$

cosh(*squareMatrix1***)⇒***kvadratMatris***

Ger matrisen med hyperbolisk cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

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

cosh⁻¹() Katalog >

cosh⁻¹(*Value1***)⇒***värde*

cosh⁻¹(*List1***)⇒***lista*

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

cosh⁻¹(*Value1***)** ger argumentets inversa hyperboliska cosinus.

cosh⁻¹(*List1***)** ger en lista på invers hyperbolisk cosinus för varje element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arccosh(...)`.

cosh⁻¹(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med invers hyperbolisk cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

$$\begin{bmatrix}2.52503+1.73485\cdot i & -0.009241-1.49086\\ 0.486969-0.725533\cdot i & 1.66262+0.623491\\ -0.322354-2.08316\cdot i & 1.26707+1.79018\end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

cot() trig tangent

cot(*Value1***)** ⇒ *värde*

cot(*List1***)** ⇒ *lista*

Ger cotangens för *Value1* eller en lista på cotangens för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

$\cot(45)$	1.

I vinkelläget Nygrader:

$\cot(50)$	1.

I vinkelläget Radianer:

$$\cot(\{1,2.1,3\})$$
$$\{0.642093,-0.584848,-7.01525\}$$

cot⁻¹() — trig tangent

cot⁻¹(*Value1***)⇒***värde*

cot⁻¹(*List1***)⇒***lista*

Ger den vinkel vars cotangens är *Value1* eller en lista på invers cotangens för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arccot(...)`**.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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

coth() — Katalog >

coth(*Value1***)⇒***värde*

coth(*List1***)⇒***lista*

Ger hyperbolisk cotangens för *Value1* eller en lista på hyperbolisk cotangens för alla element i *List1*.

$\coth(1.2)$	1.19954
$\coth(\{1,3.2\})$	$\{1.31304,1.00333\}$

coth⁻¹() — Katalog >

coth⁻¹(*Value1***)⇒***värde*

coth⁻¹(*List1***)⇒***lista*

Ger den inversa hyperboliska cotangensen för *Value1* eller en lista på invers hyperbolisk cotangens för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arccoth(...)`**.

$\coth^{-1}(3.5)$	0.293893
$\coth^{-1}(\{-2,2.1,6\})$	$\{-0.549306,0.518046,0.168236\}$

count() Katalog >

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

Ger det totala ackumulerade antalet element i argumenten som utvärderas till numeriska värden.

Varje argument kan vara ett uttryck, ett värde, en lista eller en matris. Du kan blanda datatyper och använda argument med olika dimensioner.

För en lista, en matris, eller ett område av celler, utvärderas varje element för att bestämma om det skall inkluderas i räkningen.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för ett argument.

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

$\text{count}(2,4,6)$ 3

$\text{count}(\{2,4,6\})$ 3

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

countif() Katalog >

countif(*List*,*Criteria*)⇒*värde*

Ger det totala ackumulerade antalet element i *List* som uppfyller specificerade *Criteria*.

Criteria kan vara:

- Ett värde, ett uttryck eller en sträng. Som exempel räknar **3** endast de element i *List* som förenklas till värdet 3.
- Ett booleskt uttryck som innehåller symbolen **?** fungerar som platshållare för varje element. Som exempel räknar **?<5** endast de element i *List* som är lägre än 5.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för *List*.

Tomma element i listan ignoreras. För mer information om tomma element, se på sidan 200.

$\text{countIf}(\{1,3,\text{"abc"},\text{undef},3,1\},3)$ 2

Räknar antalet element som är lika med 3.

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

Räknar antalet element som är lika med “def.”

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

Räknar 1 och 3.

$\text{countIf}(\{1,3,5,7,9\},2<?<8)$ 3

Räknar 3, 5 och 7.

countif() Katalog >

Obs: Se även **sumIf()**, på sidan 152 och **frequency()**, på sidan 57.

countIf({1,3,5,7,9},?<4 or ?>6) 4

Räknar 1, 3, 7 och 9.

cPolyRoots() Katalog >

cPolyRoots(*Poly***,***Var***)⇒***lista***

cPolyRoots(*ListaPåKoeff***)⇒***lista***

Den första syntaxen, **cPolyRoots(***Poly***,***Var***)**, ger en lista på komplexa rötter i polynomet *Poly* med avseende på *Var*.

Poly måste vara ett polynom i expanderad form i en variabel. Använd inte oexpanderade former såsom $y^2 \cdot y+1$ eller $x \cdot x+2 \cdot x+1$

Den andra syntaxen, **cPolyRoots (***ListaPåKoeff***)**, ger en lista på komplexa rötter för koefficienterna i *ListaPåKoeff*.

Obs: Se även **polyRoots()**, på sidan 114.

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·x+1,x) {-1,-1}

cPolyRoots({1,2,1}) {-1,-1}

crossP() Katalog >

crossP(*List1***,** *List2***)⇒***lista***

Ger vektorprodukten av *List1* och *List2* som en lista.

List1 och *List2* måste ha samma dimension och dimensionen måste vara antingen 2 eller 3.

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

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

Ger en rad- eller kolumnvektor (beroende på argumenten) som är vektorprodukten av *Vector1* och *Vector2*.

Både *Vector1* och *Vector2* måste vara radvektorer eller båda måste vara kolumnvektorer. Båda vektorerna måste ha samma dimension och dimensionen måste vara antingen 2 eller 3.

crossP([1 2 3],[4 5 6]) [⁻3 6 ⁻3]

crossP([1 2],[3 4]) [0 0 ⁻2]

csc() trig tangent

csc(*Value1***)⇒***värde*

csc(*List1***)⇒***lista*

Ger cosekanten för *Value1* eller en lista på cosekanten för alla element i *List1*.

I vinkelläget Grader:

$\csc(45)$	1.41421

I vinkelläget Nygrader:

$\csc(50)$	1.41421

I vinkelläget Radianer:

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

csc⁻¹() trig tangent

csc⁻¹(*Value1***)** ⇒ *värde*

csc⁻¹(*List1***)** ⇒ *lista*

Ger den vinkel vars cosekant är *Value1* eller en lista på den inversa cosekanten för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arccsc(...)`.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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

csch() Katalog >

csch(*Value1***)** ⇒ *värde*

csch(*List1***)** ⇒ *lista*

Ger den hyperboliska cosekanten för *Value1* eller en lista på den hyperboliska cosekanten för alla element i *List1*.

$\operatorname{csch}(3)$	0.099822
$\operatorname{csch}(\{1,2.1,4\})$	$\{0.850918,0.248641,0.036644\}$

csch⁻¹() Katalog >

csch⁻¹(*Value***)** ⇒ *värde*

csch⁻¹(*List1***)** ⇒ *lista*

Ger den inversa hyperboliska cosekanten för *Value1* eller en lista på den inversa hyperboliska cosekanten för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arccsch(…)`**.

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

CubicReg Katalog >

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

Utför en tredjegrads regressionsanalysy = $a \cdot x^3+b \cdot x^2+c \cdot x+d$på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*, på sidan 148.

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·x^3+b ·x^2+c ·x+d
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionsanalysen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

cumulativeSum() Katalog >

cumulativeSum(*List1***)⇒***lista***

Ger en lista på de kumulativa summorna av elementen i *List1* och börjar med element 1.

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

cumulativeSum(*Matrix1***)⇒***matris***

Ger en matris över de kumulativa summorna av elementen i *Matrix1*. Varje element är den kumulativa summan i kolumnen räknat uppifrån och ned.

Ett tomt element i *List1* eller *Matrix1* ger ett tomt element i den resulterande listan eller matrisen. För mer information om tomma element, se på sidan 200.

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

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

Cycle Katalog >

Cycle

Överför omedelbart kontroll till nästa iteration i den aktuella slingan (**For**, **While** eller **Loop**).

Cycle tillåts inte utanför de tre slingstrukturerna (**For**, **While** eller **Loop**).

Funktion som listar heltal från 1 till 100 utom 50.

Cycle

Katalog >

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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
```

▸Cylind

Katalog >

Vector **▸Cylind**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Cylind`**.

Visar rad- eller kolumnvektorn i cylindrisk form [r,∠θ, z].

Vector måste ha exakt tre element. Den kan vara antingen en rad eller en kolumn.

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

D

dbd()

Katalog >

dbd(*date1,date2***)⇒***värde*

Ger antalet dagar mellan *date1* och *date2* med dagräkningsmetoden.

date1 och *date2* kan vara tal eller listor på tal inom den normala kalendern. Om både *date1* och *date2* är listor måste de vara lika långa.

date1 och *date2* måste vara mellan 1950 och 2049.

Du kan mata in datumen i ett av två format. Decimalplaceringen skiljer sig mellan de två datumformaten.

MM.DDYY (format som ofta används i USA)

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

dbd() Katalog >

DDMM.YY (format som ofta används i Europa)

▸DD Katalog >

Expr1 ▸**DD**⇒*värde*

List1 ▸**DD**⇒*lista*

Matrix1 ▸**DD**⇒*matris*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>DD`**.

Ger den decimala ekvivalenten till argumentet uttryckt i grader. Argumentet är ett tal, en lista eller en matris som tolkas av vinkelläget i nygrader, radianer eller grader.

I vinkelläget Grader:

$(1.5°)$▸DD	$1.5°$
$(45°22'14.3")$▸DD	$45.3706°$
$\{\{45°22'14.3",60°0'0"\}\}$▸DD	$\{45.3706°,60°\}$

I vinkelläget Nygrader:

1▸DD	$\frac{9}{10}°$

I vinkelläget Radianer:

(1.5)▸DD	$85.9437°$

▸Decimal Katalog >

Number1 ▸**Decimal**⇒*värde*

List1 ▸**Decimal**⇒*värde*

Matrix1 ▸**Decimal**⇒*värde*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Decimal`**.

Visar argumentet i decimal form. Operatorn kan endast användas i slutet av inmatningsraden.

$\frac{1}{3}$▸Decimal	0.333333

Define (Definiera) Katalog >

Define *Var* **=** *Uttryck*

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

Definierar variabeln *Var* eller den användardefinierade funktionen *Function*.

Parametrar såsom *Param1* utgör platshållare för att överföra argument till funktionen. När du anropar en användardefinierad funktion måste du ha argument (till exempel, värden eller variabler) som överensstämmer med parametrarna. När funktionen anropas beräknar den *Expression* med de givna argumenten.

Var och *Function* får inte vara namnet på en systemvariabel eller inbyggd funktion eller ett kommando.

Obs: Denna form av **Define** är ekvivalent med att exekvera uttrycket: *expression* → *Function*(*Param1,Param2*).

```
Define g(x,y)=2·x−3·y                    Done
g(1,2)                                   -4
1→a: 2→b: g(a,b)                         -4
Define h(x)=when(x<2,2·x−3,-2·x+3)       Done
h(-3)                                    -9
h(4)                                     -5
```

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

Block

EndFunc

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

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

Block

EndPrgm

I denna form kan den användardefinierade funktionen eller programmet exekvera ett block av flera påståenden.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden på separata rader. *Block* kan även inkludera uttryck och instruktioner (till exempel, **If**, **Then**, **Else** och **For**).

```
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 (Definiera) Katalog >

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Obs: Se även **Define LibPriv**, på sidan 38 och **Define LibPub**, på sidan 38.

Define LibPriv Katalog >

Define LibPriv *Var* **=** *Uttryck*

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

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

Block

EndFunc

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

Block

EndPrgm

Fungerar på samma sätt som **Define** förutom att en privat biblioteksvariabel, funktion eller program definieras. Privata funktioner och program visas inte i Katalogen.

Obs: Se även **Define**, på sidan 36 och **Define LibPub**, på sidan 38.

Define LibPub Katalog >

Define LibPub *Var* **=** *Uttryck*

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

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

Block

Define LibPub Katalog >

EndFunc

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

Block

EndPrgm

Fungerar på samma sätt som **Define** förutom att en allmän biblioteksvariabel, funktion eller program definieras. Allmänna funktioner och program visas i Katalogen när biblioteket har sparats och uppdaterats.

Obs: Se även **Define**, på sidan 36 och **Define LibPriv**, på sidan 38.

deltaList() Se ΔList(), på sidan 84.

DelVar Katalog >

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

DelVar *Var*.

Tar bort den specificerade variabeln eller variabelgruppen från minnet.

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 166.

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

DelVar *Var*. tar bort alla led i variabelgruppen *Var*. variabelgrupp (till exempel, den statistiska *stat.nn*-resultaten eller variabler skapade med funktionen **LibShortcut()**). Punkten (.) i denna form av **DelVar**-kommandot begränsar kommandot till borttagning av en variabelgrupp: den enkla variabeln *Var* påverkas inte.

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

delVoid() Katalog >

delVoid($List1$**)**$\Rightarrow lista$

Ger en lista med innehållet i $List1$ med alla tomma element borttagna.

För mer information om tomma element, se på sidan 200.

$\text{delVoid}(\{1,\text{void},3\})$	$\{1,3\}$

det() Katalog >

det($squareMatrix$[**,** $Tolerance$]**)**$\Rightarrow uttryck$

Ger determinanten för $squareMatrix$.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än $Tolerance$. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras $Tolerance$.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om $Tolerance$ utelämnas eller inte används beräknas standardtoleransen som:

  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() Katalog >

diag($List$**)**$\Rightarrow matris$

diag($rowMatrix$**)**$\Rightarrow matris$

diag($columnMatrix$**)**$\Rightarrow matris$

Ger en matris med värdena i argumentlistan eller matrisen i dess huvuddiagonal.

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

diag() Katalog >

diag(*squareMatrix***)⇒***radMatris*

Ger en radmatris som innehåller elementen från huvuddiagonalen hos *squareMatrix*.

squareMatrix måste vara kvadratisk.

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

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

dim() Katalog >

dim(*List***)⇒***heltal*

Ger dimensionen på *List*.

$\text{dim}(\{0,1,2\})$ $\quad 3$

dim(*Matrix***)⇒***lista*

Ger dimensionerna på en matris som en lista med två element {rader, kolumner}.

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

dim(*String***)⇒***heltal*

Ger antalet tecken i teckensträngen *String*.

$\text{dim}(\text{"Hello"})$ $\quad 5$

$\text{dim}(\text{"Hello "\&"there"})$ $\quad 11$

Disp Katalog >

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

Visar argumenten i *Calculator*-historiken. Argumenten visas i ordningsföljd med utslutning som separatorer.

Huvudsakligen användbart i program och funktioner för att säkerställa visningen av mellanliggande beräkningar.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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
```

DispAt Katalog >

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

DispAt låter dig specificera den rad där det specifika uttrycket eller strängen kommer att visas på skärmen.

```
DispAt
```

Exempel

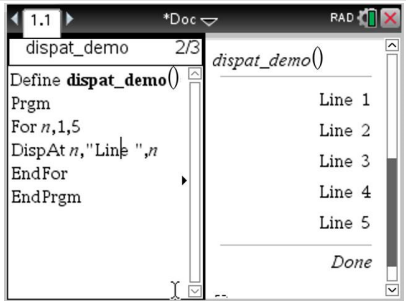
Radnumret kan specificeras som ett uttryck.

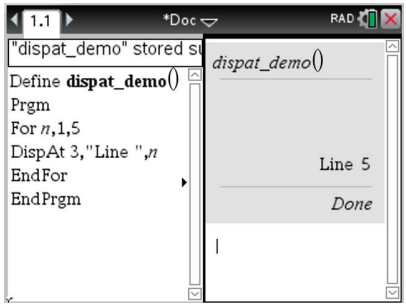
Observera att radnumret inte gäller för hela skärmen utan för det område som direkt följer kommandot/programmet.

Detta kommando möjliggör kontrollpanelsliknande utdata från program där värdet av ett uttryck eller från en sensoravläsning uppdateras på samma rad.

DispAtoch Disp kan användas i samma program.

Obs: Maxnummer är inställt på 8 eftersom detta motsvarar en skärm fylld med rader på en skärm hos en handenhet – så länge raderna inte har matematiska uttryck i 2D. Det exakta antalet rader beror på innehållet i den information som visas.





Illustrativa exempel:

Define z()=	Utdata
Prgm	z()
For n,1,3	Iteration 1:
DispAt 1,"N: ",n	Rad 1: N:1
Disp "Hej"	Rad 2: Hej
EndFor	
EndPrgm	Iteration 2:
	Rad 1: N:2
	Rad 2: Hej
	Rad 3: Hej
	Iteration 3:
	Rad 1: N:3
	Rad 2: Hej
	Rad 3: Hej

	Rad 4: Hej
Define z1()=	z1()
Prgm	Rad 1: N:3
For n,1,3	Rad 2: Hej
DispAt 1,"N: ",n	Rad 3: Hej
EndFor	Rad 4: Hej
	Rad 5: Hej
For n,1,4	
Disp "Hej"	
EndFor	
EndPrgm	

Feltillstånd:

Felmeddelande	**Beskrivning**
Radnummer för DispAt måste vara mellan 1 och 8	Uttryck utvärderar radnummer utanför intervallet 1-8 (inklusive)
Too few arguments (För få argument)	Funktionen eller kommandot saknar ett eller flera argument.
Inga argument	Samma som nuvarande dialogruta för "syntaxfel" dialog
Too many arguments (För många argument)	Begränsa argument. Samma fel som Disp.
Invalid data type (Ogiltig datatyp)	Första argumentet måste vara ett tal.
Tom: DispAt tom	Datatypfelet "Hello World" kastas bort (om återanrop definieras)

▸DMS

Katalog >

Value ▸*DMS*

List ▸**DMS**

Matrix ▸**DMS**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>DMS**.

I vinkelläget Grader:

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

▸DMS

Katalog >

Tolkar argumentet som en vinkel och visar motsvarande DMS-värde (DDDDDD°MM'SS.ss''). Se °, ', '', på sidan 194 för DMS-format (grad, minuter, sekunder).

Obs: ▸DMS konverterar från radianer till grader vid användning i läget radianer. Om inmatningen följs av en gradsymbol ° sker ingen konvertering. Du kan bara använda **▸DMS** i slutet av en inmatningsrad.

dotP()

Katalog >

dotP(*List1***, *List2*)⇒*uttryck***

Ger "prick"-produkten av två listor.

dotP({1,2},{5,6})	17

dotP(*Vector1***, *Vector2*)⇒*uttryck***

Ger "prick"-produkten av två vektorer.

dotP([1 2 3],[4 5 6])	32

Båda måste vara radvektorer eller båda måste vara kolumnvektorer.

E

e^()

e^x-tangent

e^(*Value1***)⇒*värde***

Ger **e** upphöjt till potensen *Value1*.

e^1	2.71828
e^{3^2}	8103.08

Obs: Se även **e exponentmall**, på sidan 2.

Obs: Att trycka på e^x för att visa e^(skiljer sig från att trycka på tecknet **E** på tangentbordet.

Du kan skriva in ett komplext tal i den polära formen $re^{i\theta}$. Använd dock denna form endast i vinkelläget Radianer: den orsakar ett områdesfel i vinkelläget Grader eller Nygrader.

e^(*List1***)⇒*lista***

Ger **e** upphöjt till potensen för varje element i *List1*.

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

e^() e^x-tangent

e^(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med exponenten för *squareMatrix1*. Detta är inte detsamma som att beräkna e upphöjt till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

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

eff() Katalog >

eff(*nominalRate,CpY***)⇒***värde*

Finansiell funktion som konverterar den nominella räntan *nominalRate* till en årlig effektiv ränta, given av *CpY* som antalet ränteperioder per år.

nominalRate måste vara ett reellt tal och *CpY* måste vara ett reellt tal > 0.

Obs: Se även **nom()**, på sidan 103.

$$\text{eff}(5.75,12) \qquad 5.90398$$

eigVc() Katalog >

eigVc(*squareMatrix***)⇒***matris*

Ger en matris som innehåller egenvektorerna för en reell eller komplex *squareMatrix* där varje kolumn i resultatet motsvarar ett egenvärde. Observera att en egenvektor inte är unik: den kan vara skalad med vilken konstant faktor som helst. Egenvektorerna är normaliserade, vilket innebär att om V = [x_1, x_2, ..., x_n], så är:

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

squareMatrix balanseras först med liknande transformationer tills rad- och kolumnnormerna har så nära samma värde som möjligt. *squareMatrix* reduceras sedan till övre Hessenberg-form och egenvektorerna beräknas med en Schur-faktorisering.

I Rektangulärt komplext format:

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

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

eigVl() Katalog >

eigVl(*squareMatrix***)⇒***lista*

Ger en lista på egenvärdena för en reell eller komplex *squareMatrix*.

squareMatrix balanseras först med likhetstransformationer tills rad- och kolumnnormerna har så nära samma värde som möjligt. *squareMatrix* reduceras sedan till övre Hessenberg-form och egenvektorerna beräknas från den övre Hessenberg-matrisen.

I läget Rektangulärt komplext format:

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

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Else Se If, på sidan 69.

ElseIf Katalog >

If *BooleanExpr1* **Then**

Block1

ElseIf *BooleanExpr2* **Then**

Block2

⋮

ElseIf *BooleanExprN* **Then**

BlockN

EndIf

⋮

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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 Se For, på sidan 55.

EndFunc

Se Func, på sidan 58.

EndIf

Se If, på sidan 69.

EndLoop

Se Loop, på sidan 90.

EndPrgm

Se Prgm, på sidan 115.

EndTry

Se Try, på sidan 159.

EndWhile

Se While, på sidan 169.

euler ()

Katalog >

euler(*Expr*, *Var*, *depVar*, {*Var0, VarMax*}, *depVar0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

euler(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

euler(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}**x** *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

Använder Eulers metod för att lösa systemet

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

Differentialekvation:

y'=0,001*y*(100-y) och y(0)=10

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Ekvationssystem:

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

med $y1(0)=2$ och $y2(0)=5$

med *depVar*(*Var0*)=*depVar0* på intervallet [*Var0*,*VarMax*]. Ger en matris vars första rad definierar resultatvärdena för *Var* och vars andra rad definierar den första lösningskomponenten vid motsvarande *Var*-värden, och så vidare.

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

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

Expr är det högra ledet som definierar den ordinära differentialekvationen (ODE).

SystemOfExpr är systemet av högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

ListOfExpr är en lista på högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

Var är den oberoende variabeln.

ListOfDepVars är en lista på oberoende variabler.

{*Var0*, *VarMax*} är en lista med två element som instruerar funktionen att integrera från *Var0* till *VarMax*.

ListOfDepVars0 är en lista på startvärden för oberoende variabler.

VarStep är ett tal skilt från noll så att **sign (***VarStep***)** = **sign(***VarMax*-*Var0***)** och lösningar ges vid *Var0*+i·*VarStep* för alla i=0,1,2,... sådana att *Var0*+i·*VarStep* är i intervallet [*var0*,*VarMax*] (det kanske inte finns ett lösningsvärde vid *VarMax*).

eulerStep är ett positivt heltal (förinställs på 1) som definierar antalet euler-steg mellan resultatvärden. Den verkliga stegstorleken som används av euler-metoden är *VarStep*/*eulerStep*.

eval () Hubb-meny

eval(*Expr***)** ⇒ *string*

Ställ in den blå komponenten hos RGB-lysdioden på halv intensitet.

eval ()

Hubb-meny

eval() är giltig endast i [[[Undefined variable nspire_all_var.HubFullName]]] Kommandoargument i programmeringskommandona **Get**, **GetStr** och **Send**. Programmet beräknar uttrycket *Expr* och ersätter **eval()**-meddelandet med resultatet som en teckensträng.

Argumentet *Expr* måste generera ett reellt tal.

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

Återställ den blå komponenten till OFF.

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

Argumentet eval() måste generera ett reellt tal.

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

Program för att tona in den röda komponenten

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

Kör programmet.

```
fadein()                                   Done
```

Även om **eval()** inte visar dess resultat kan du visa den resulterande Hubb-kommandosträngen efter att ha kört kommandot genom att kontrollera någon av följande speciella variabler.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

```
n:=0.25                                    0.25
m:=8                                       8
n·m                                        2.
Send "SET COLOR.BLUE ON TIME eval(n·m)"
                                           Done
iostr.SendAns     "SET COLOR.BLUE ON TIME 2"
```

Obs: Se även **Get** (på sidan 60), **GetStr** (på sidan 67), och **Send** (på sidan 135).

Exit

Katalog >

Exit

Avslutar det aktuella blocket **For**, **While** eller **Loop**.

Funktionslista:

Exit

Katalog >

Exit tillåts inte utanför de tre slingstrukturerna (**For**, **While** eller **Loop**).

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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-tangent

exp(*Value1***)⇒***värde*

Ger **e** upphöjt till potensen *Value1*.

Obs: Se även **e** exponentmall, på sidan 2.

Du kan skriva in ett komplext tal i den polära formen $re^{i\theta}$. Använd dock denna form endast i vinkelläget Radianer: den orsakar ett områdesfel i vinkelläget Grader eller Nygrader.

```
e^1                                2.71828
e^(3^2)                            8103.08
```

exp(*List1***)⇒***lista*

Ger **e** upphöjt till potensen för varje element i *List1*.

```
e^{1,1.,0.5}        {2.71828,2.71828,1.64872}
```

exp(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med exponenten för *squareMatrix1*. Detta är inte detsamma som att beräkna **e** upphöjt till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

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

expr() Katalog >

expr(*String***)⇒***uttryck*

Ger teckensträngen i *String* som ett uttryck och exekverar det omedelbart.

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

ExpReg Katalog >

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

Beräknar den exponentiella regressioneny = a ·(b)^{x}på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·(b)^x
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Koefficient för linjär bestämning av transformerade data

Resultatvariabel	Beskrivning
stat.r	Korrelationskoefficient för transformerade data (x, ln(y))
stat.Resid	Residualer associerade med den exponentiella modellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

F

factor() Katalog >

factor(*rationalNumber***)** ger det rationella talet faktoriserat i primtal. För sammansatta tal ökar beräkningstiden exponentiellt med antalet siffror i den näst största faktorn. Som exempel kan faktorisering av ett 30-siffrigt heltal ta mer än en dag och faktorisering av ett 100-siffrigt tal kan ta mer än 100 år.

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

För att stoppa en beräkning manuellt,

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Om du endast vill bestämma om ett tal är ett primtal, använd **isPrime()** i stället. Detta går mycket fortare, särskilt om *rationalNumber* inte är ett primtal och om den näst största faktorn har mer än fem siffror.

FCdf() Katalog >

FCdf
(*lowBound***,***upBound***,***dfNumer***,***dfDenom***)**
⇒*number* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

FCdf
(*lowBound***,***upBound***,***dfNumer***,***dfDenom***)**
⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten för F -fördelningen mellan *undre gräns* och *övre gräns* för det specificerade *dfNämn* (frihetsgrader) och *dfTälj*.

För P($X \le$ *övrGräns*), ange *undrGräns* = 0.

Fill Katalog >

Fill *Value, matrixVar*⇒*matris*

Ersätter varje element i variabeln *matrixVar* med *Value*.

matrixVar måste redan existera.

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

Fill 1.01,*amatrix* *Done*

amatrix $\begin{bmatrix} 1.01 & 1.01 \\ 1.01 & 1.01 \end{bmatrix}$

Fill *Value, listVar*⇒*lista*

Ersätter varje element i variabeln *listVar* med *Value*.

listVar måste redan existera.

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

Fill 1.01,*alist* *Done*

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

FiveNumSummary Katalog >

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

Ger en förkortad version av envariabelstatistiken för listan *X*.
En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

X representerar en lista på aktuella data.

FiveNumSummary

Katalog >

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*-värde. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 200.

Resultatvariabel	Beskrivning
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde

floor()

Katalog >

floor(*Value1***)⇒***heltal*

floor(−2.14) −3.

Ger det största heltal som är ≤ argumentet. Denna funktion är identisk med **int()**.

Argumentet kan vara ett reellt eller ett komplext tal.

floor(*List1***)⇒***lista*

floor({3/2, 0, −5.3}) {1, 0, −6.}

floor(*Matrix1***)⇒***matris*

floor([1.2 3.4; 2.5 4.8]) [1. 3.; 2. 4.]

Ger en lista eller matris med golvvärden för varje element.

Obs: Se även **ceiling()** och **int()**.

For

Katalog >

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

Block

EndFor

Exekverar iterativt påståendena i *Block* för varje värde på *Var*, från *Low* till *High*, i steg enligt *Step*.

Var får inte vara en systemvariabel.

Step kan vara positivt eller negativt. Det förinställda värdet är 1.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

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

format()

Katalog >

format(*Value*[, *formatString*]**)**⇒*sträng*

Ger *Value* som en teckensträng baserad på formatmallen.

formatString är en sträng och måste ha formen: "F[n]", "S[n]", "E[n]", "G[n][c]", där [ ] indikerar frivilliga delar.

F[n]: Fast format. n är antalet siffror som skall visas efter decimalpunkten.

S[n]: Scientific format (Grundpotensform). n är antalet siffror som skall visas efter decimalpunkten.

E[n]: Engineering format. n är antalet siffror efter den första signifikanta siffran. Exponenten justeras till en multipel av tre och decimalpunkten flyttas åt höger med noll, en eller två siffror.

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"

format() Katalog >

G[n][c]: Samma som fast format, men separerar också siffor till vänster om basen i grupper om tre. c specificerar det gruppseparerande tecknet och är förinställt på kommatecken. Om c är en punkt visas basen som ett kommatecken.

[Rc]: Samtliga ovanstående specifikationssymboler kan förses med suffix med Rc-basflaggan, där c är ett enstaka tecken som specificerar vad som skall ersättas för baspunkten.

fPart() Katalog >

fPart(*Expr1***)⇒***uttryck*

fPart(*List1***)⇒***lista*

fPart(*Matrix1***)⇒***matris*

Ger argumentets bråkdel.

Ger, för en lista eller matris, elementens bråkdelar.

Argumentet kan vara ett reellt eller ett komplext tal.

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

FPdf() Katalog >

FPdf(*XVal***,***dfNumer***,***dfDenom***)⇒***tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar sannolikheten för F-fördelning vid *XVal* för specificerad *dfNumer* (frihetsgrader) och *dfDenom*.

freqTable▸list() Katalog >

freqTable▸list(*List1***,***freqIntegerList***)** ⇒*lista*

Ger en lista som innehåller elementen från *List1* expanderad enligt frekvenserna i *freqIntegerList*. Denna funktion kan användas för att skapa en frekvenstabell för applikationen Data & Statistik.

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}

freqTable▸list() Katalog >

List1 kan vara vilken giltig lista som helst.

freqIntegerList måste ha samma dimensioner som *List1* och får endast innehålla icke-negativa heltalselement. Varje element specificerar antalet gånger motsvarande *List1*-element kommer att upprepas i resultatlistan. Ett värde på noll utesluter motsvarande *List1*-element.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`freqTable@>list(…)`**.

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

frequency() Katalog >

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

Ger en lista med antalet element i *List1*. Talen baseras på områden (bins = staplar) som du definierar i *binsList*.

Om *binsList* är {b(1), b(2), …, b(n)} är de specificerade områdena {**?**≤b(1), b(1)<**?**≤b (2),…,b(n-1)<**?**≤b(n), b(n)>**?**}. Den resulterande listan är ett element längre än *binsList*.

Varje element i resultatet motsvarar antalet element från *List1* som är i området för denna stapel. Uttryckt enligt funktionen **countIf()** är resultatet { countIf (list, **?**≤b(1)), countIf(list, b(1)<**?**≤b(2)), …, countIf(list, b(n-1)<**?**≤b(n)), countIf(list, b(n) >**?**)}.

Element i *List1* som inte kan "placeras i en stapel" ignoreras. Även tomma element ignoreras. För mer information om tomma element, se på sidan 200.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för båda argumenten.

Obs: Se även **countIf()**, på sidan 30.

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}

Förklaring av resultat:

2 element från *Datalist* är ≤2.5

4 element från *Datalist* är >2.5 och ≤4.5

3 element från *Datalist* är >4.5

Elementet "hello" är en sträng och kan inte placeras i någon av de definierade staplarna.

FTest_2Samp Katalog >

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

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

(Indatalista)

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

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

(Summary stats indata)

Utför ett 2-sampel F test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

eller H_a: σ1 > σ2, sätt *Hypoth*>0

För H_a: σ1 ≠ σ2 (förinställning), sätt *Hypoth* =0

För H_a: σ1 < σ2, sätt *Hypoth*<0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.F	Beräknad Û-statistik för datasekvensen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.dfNumer	täljare, frihetsgrader = n1-1
stat.dfDenom	nämnare, frihetsgrader = n2-1
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.x1_bar stat.x2_bar	Medelvärden hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.n1, stat.n2	Storlek på urvalen

Func Katalog >

Func Definiera en stegvis funktion:

Block

Func Katalog >

EndFunc

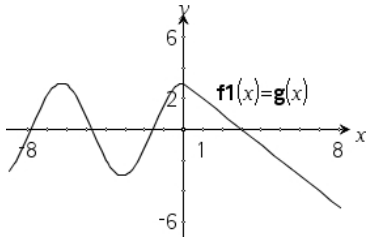
Mall för att skapa en användardefinierad funktion.

Block kan vara ett enstaka påstående, en serie av påståenden separerade med tecknet ":" eller en serie av påståenden på separata rader. Funktionen kan använda instruktionen **Return** för att ge ett specifikt resultat.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

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

Resultat från plottning av g(x)



G

gcd() Katalog >

gcd(*Value1, Value2***)⇒***uttryck*

Ger den största gemensamma delaren för de två argumenten. **gcd** för två bråk är **gcd** för deras täljare dividerat med **lcm** för deras nämnare.

I läge Auto eller Approximate (Ungefärlig) är **gcd** 1.0 för bråktal i flyttalsform.

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

gcd(*List1, List2***)⇒***lista*

Ger största gemensamma delare för motsvarande element i *List1* och *List2*.

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

gcd(*Matrix1, Matrix2***)⇒***matris*

Ger största gemensamma delare för motsvarande element i *Matrix1* och *Matrix2*.

```
gcd([2 4; 6 8],[4 8; 12 16])        [2 4; 6 8]
```

geomCdf() Katalog >

geomCdf(*p,lowBound,upBound***)⇒***tal* om *lowBound* och *upBound* är tal, *lista* om

geomCdf() Katalog >

lowBound och *upBound* är listor

geomCdf(*p*,*upBound***)**för P(1≤X≤*upBound*) ⇒*tal* om *upBound* är ett tal, *lista* om *upBound* är en lista

Beräknar en kumulativ geometrisk sannolikhet från *lowBound* till *upBound* med den specificerade sannolikheten *p* för att lyckas.

För P(X ≤ *upBound*), sätt *lowBound* = 1.

geomPdf() Katalog >

geomPdf(*p*,*XVal***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar en sannolikhet vid *XVal*, vid vilket försök i försöksomgången som man lyckas första gången, för den diskreta geometriska fördelningen med den specificerade sannolikheten p för att lyckas.

Get Hubb-meny

Get[*promtString*,] *var*[, *statusVar*]

Get[*promtString*,] *func***(***arg1*, *...argn***)** [, *statusVar*]

Programmeringskommando: Hämtar ett värde från en ansluten [[[Undefined variable nspire_all_var.HubFullName]]] och tilldelar värdet till variabeln *var*.

Värdet måste begäras:

- På förhand genom ett **Send "READ ..."** -kommando.

  — eller —

- Genom att bädda in en **"READ ..."** -begäran som alternativt *promptString*-argument. Med denna metod kan du använda ett enda kommando för att begära värdet och hämta det.

Exempel: Begär nuvarande värde från hubbens inbyggda ljusnivåsensor. Använd **Get** för att hämta värdet och tilldela det till variabeln *lightval*.

Send "READ BRIGHTNESS"	*Done*
Get *lightval*	*Done*
lightval	0.347922

Bädda in READ-begäran i **Get**-kommandot.

Get "READ BRIGHTNESS",*lightval*	*Done*
lightval	0.378441

Get — Hubb-meny

Implicit förenkling äger rum. Till exempel tolkas en mottagen sträng ”123” som ett numeriskt värde. För att bevara strängen, använd **GetStr** istället för **Get**.

Om du inkluderar det valfria argumentet *statusVar*, tilldelas det ett värde baserat på om operationen har lyckats. Värdet noll betyder att inga data mottogs.

I den andra syntaxen kan ett program använda argumentet *func*() för att lagra den mottagna strängen som en funktionsdefinition. Denna syntax fungerar som om programmet exekverade kommandot:

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

Programmet kan sedan använda den definierade funktionen *func*().

Obs: Du kan använda kommandot **Get** i ett användardefinierat program, men inte i en funktion.

Obs: Se även **GetStr**, på sidan 67 och **Send**, på sidan 135.

getDenom() — Katalog >

getDenom(*Fraction1***)⇒***värde*

Transformerar argumentet till ett uttryck med reducerad gemensam nämnare och ger sedan dess nämnare.

$x:=5: y:=6$	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

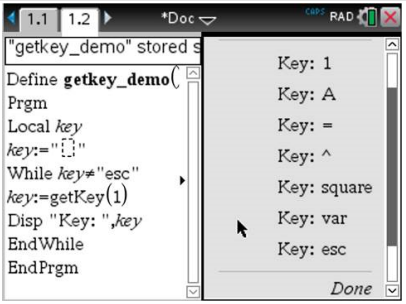
getKey() — Katalog >

getKey([0|1]) ⇒ returnString

getKey()

Exempel:

Beskrivning:getKey() – låter ett TI-Basic-program tolka tangentbordsinmatning – på handenhet, dator och emulator på dator.



Exempel:

- keypressed := **getKey()** kommer att returnera en tangent eller tom sträng om ingen tangent har tryckts ned. Detta anrop returneras omedelbart.
- keypressed := **getKey(1)** väntar till en tangent trycks ned. Detta anrop pausar start av ett program tills en tangent trycks ned.

Hantering av tangentnedtryckningar:

Handhållen enhet/emulatortangent	Desktop (dator)	Returvärde
Esc	Esc	"esc"
Pekplatta - toppklick	Ej tillämpligt	"up"
På	Ej tillämpligt	"home"
Scratchapps	Ej tillämpligt	"scratchpad"
Pekplatta - vänsterklick	Ej tillämpligt	"left"
Pekplatta - mittklick	Ej tillämpligt	"center"
Pekplatta - högerklick	Ej tillämpligt	"right"
Doc	Ej tillämpligt	"doc"
Tab	Tab	"tab"
Pekplatta – nederklick	Pil ned	"down"
Meny	Ej tillämpligt	"menu"
Ctrl	Ctrl	ej retur
Skift	Skift	ej retur
Var	Ej tillämpligt	"var"
Radera	Ej tillämpligt	"del"

Handhållen enhet/emulatortangent	Desktop (dator)	Returvärde
=	=	"="
trig	Ej tillämpligt	"trig"
0 till 9	0–9	"0" ... "9"
Mallar	Ej tillämpligt	"template"
Katalog	Ej tillämpligt	"cat"
^	^	"^"
X^2	Ej tillämpligt	"square"
/ (divisionstangent)	/	"/"
* (multiplikationstangent)	*	"*"
e^x	Ej tillämpligt	"exp"
10^x	Ej tillämpligt	"10power"
+	+	"+"
-	-	"-"
(	(	"("
)	)	")"
.	.	"."
(-)	Ej tillämpligt	"-" (negativt tecken)
Mata in	Mata in	"enter"
ee	Ej tillämpligt	"E" (grundpotensform)
a - z	a-z	alpha = bokstav nedtryckt (gemen) ("a" - "z")
shift a-z	shift a-z	alpha = bokstav nedtryckt "A" - "Z"
		Obs: ctrl-shift låser versaler (caps)
?!	Ej tillämpligt	"?!"
pi	Ej tillämpligt	"pi"

Handhållen enhet/emulatortangent	Desktop (dator)	Returvärde
Flagga	Ej tillämpligt	ej retur
,	,	","
Retur	Ej tillämpligt	"return"
Mellanslag	Mellanslag	" " (mellanslag)
Ej tillgänglig	Specialtecken såsom @,!,^, etc.	Tecknet returneras
Ej tillämpligt	Funktionstangenter	Inga returnerade tecken
Ej tillämpligt	Specialkontrolltangenter för dator	Inga returnerade tecken
Ej tillgänglig	Andra datortangenter som inte finns tillgängliga på handenheten medan getkey () väntar på en tangentnedtryckning. ({, },;, :, ...)	Samma tecken du får i Anteckningar (inte i en matematikruta)

Obs: Det är viktigt att observera att närvaron av **getKey()** i ett program ändrar hur systemet hanterar vissa händelser. Vissa av dessa beskrivs nedan.

Avsluta program och hantera händelse – Precis som om användaren vill lämna programmet genom att trycka på tangenten **ON**

"**Support**" nedan innebär – Systemet fungerar som förväntat - programmet fortsätter att köras.

Händelse	Enhet	Dator – TI-Nspire™ Student Software
Snabbtest	Avsluta program, hantera händelse	Samma som handenhet (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-enbart)
Hantering av fjärrfil (Inkl. utskick av fil "Exit Press 2 Test" från en annan handenhet eller dator)	Avsluta program, hantera händelse	Samma som handenhet. (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-enbart)
Avsluta klass	Avsluta program, hantera händelse	Support (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-enbart)

Händelse	Enhet	Dator - TI-Nspire™ Alla versioner
TI-Innovator™ Hub anslut/koppla från	Support – Kan enkelt utfärda kommandon till TI-Innovator™ Hub. Efter att du lämnat programmet jobbar TI-Innovator™ Hub fortfarande med den handenheten.	Samma som handenheten

getLangInfo() Katalog >

getLangInfo()⇒*sträng*

Ger en sträng som motsvarar det korta namnet på det aktuella aktiva språket. Du kan exempelvis använda den i ett program eller i en funktion för att bestämma det aktuella språket.

Engelska = “en”
Danska = “da”
Tyska = “de”
Finska = “fi”
Franska = “fr”
Italienska = “it”
Holländska = “nl”
Belgisk holländska = “nl_BE”
Norska = “no”
Portugisiska = “pt”
Spanska = “es”
Svenska = “sv”

getLangInfo()	"en"

getLockInfo() Katalog >

getLockInfo(*Var*)⇒*värde*

Ger den aktuella låsta/olåsta statusen för variabeln *Var*.

värde = 0: *Var* är olåst eller finns inte.

värde = 1: *Var* är låst och kan inte modifieras eller tas bort.

Se **Lock**, på sidan 87 och**unLock**, på sidan 166.

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() Katalog >

getMode(*ModeNameInteger***)⇒***värde*

getMode(0)⇒*lista*

getMode(*ModeNameInteger***)** ger ett värde som representerar den aktuella lägesinställningen för *ModeNameInteger*.

getMode(0) ger en lista på talpar. Varje par består av ett lägesheltal och ett inställningsheltal.

Se nedan för en lista på lägen och deras inställningar.

Om du sparar inställningarna med **getMode (0)** → *var* kan du använda **setMode(***var***)** i en funktion eller ett program för att temporärt återställa inställningarna endast inom exekveringen av funktionen eller programmet. Se **setMode()**, på sidan 138.

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

Lägets namn	Läges-heltal	Heltal för inställningar
Display Digits (Antal siffror)	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 (Vinkel)	2	**1**=Radian, **2**=Degree, **3**=Gradian
Exponential Format (Exponentiellt format)	3	**1**=Normal, **2**=Scientific, **3**=Engineering
Real or Complex (Reellt eller Komplext)	4	**1**=Real, **2**=Rectangular, **3**=Polar
Auto or Approx. (Auto eller Ungefärlig)	5	**1**=Auto, **2**=Approximate
Vector Format (Vektorformat)	6	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base (Bas)	7	**1**=Decimal, **2**=Hex, **3**=Binary

getNum() Katalog >

getNum(*Fraction1***)⇒***värde*

Transformerar argumentet till ett uttryck med reducerad gemensam nämnare och ger sedan dess täljare.

$x:=5: y:=6$	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

GetStr Hubb-meny

GetStr[*promtString*,] *var*[, *statusVar*]

GetStr[*promtString*,] *func*(*arg1*, ...*argn*) [, *statusVar*]

Programmeringskommando: Fungerar precis som kommandot **Get**, förutom att det mottagna värdet alltid tolkas som en sträng. I motsats tolkar kommandot **Get** svaret som ett uttryck såvida det inte är omgivet av citationstecken ("").

Obs: Se även **Get**, på sidan 60 och **Send**, på sidan 135.

Se **Get** för exempel.

getType() Katalog >

getType(*var***)⇒***sträng*

Ger en sträng som anger datatypen för variabeln *var*.

Om *var* inte har definierats erhålls strängen "NONE".

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

getVarInfo()⇒*matris* eller *sträng*

getVarInfo(*LibNameString***)**⇒*matris* eller *sträng*

getVarInfo() ger en matris med information (variabelnamn, typ, åtkomlighet till bibliotek och låst/olåst status) för alla variabler och biblioteksobjekt som är definierade i det aktuella problemet.

Om inga variabler är definierade ger **getVarInfo()** strängen "NONE".

getVarInfo(*LibNameString***)**ger en matris med information för alla biblioteksobjekt som är definierade i biblioteket *LibNameString*. *LibNameString* måste vara en sträng (text omsluten med citationstecken) eller en strängvariabel.

Om biblioteket *LibNameString* inte finns uppstår ett fel.

Se exemplet till vänster där resultatet av **getVarInfo()** tilldelas variabeln *vs*. Ett försök att visa rad 2 eller rad 3 av *vs* ger ett “Ogiltig lista eller matris”-fel eftersom minst ett av elementen i dessa rader (till exempel, variabel *b*) omvärderas till en matris.

Detta fel kan också inträffa när *Ans* används för att utvärdera ett **getVarInfo()**-resultat på nytt.

Systemet ger ovanstående fel eftersom den aktuella versionen av programvaran inte stöder en generaliserad matrisstruktur där ett element i en matris kan vara antingen en matris eller en lista.

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*
getVarInfo()	$\begin{bmatrix} x & \text{"NUM"} & \text{"}\square\text{"} & 1 \\ y & \text{"LIST"} & \text{"LibPriv "} & 0 \\ z & \text{"FUNC"} & \text{"LibPub "} & 0 \end{bmatrix}$
getVarInfo(*tmp3*)	"Error: Argument must be a string"
getVarInfo("tmp3")	$\begin{bmatrix} volcyl2 & \text{"NONE"} & \text{"LibPub "} & 0 \end{bmatrix}$

a:=1	1
b:=[1 2]	[1 2]
c:=[1 3 7]	[1 3 7]
vs:=getVarInfo()	$\begin{bmatrix} a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ b & \text{"MAT"} & \text{"}\square\text{"} & 0 \\ c & \text{"MAT"} & \text{"}\square\text{"} & 0 \end{bmatrix}$
$vs[1]$	[1 "NUM" "□" 0]
$vs[1,1]$	1
$vs[2]$	"Error: Invalid list or matrix"
$vs[2,1]$	[1 2]

Goto

Katalog >

Goto *labelName*

Överför kontroll till etiketten *labelName*.

labelName måste definieras i samma funktion med en **Lbl**-instruktion.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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

Katalog >

Expr1 ▸ **Grad**⇒*uttryck*

Konverterar *Expr1* till en vinkelmätning i nygrader.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Grad**.

I vinkelläget Grader:

```
(1.5)▸Grad                         (1.66667)g
```

I vinkelläget Radianer:

```
(1.5)▸Grad                         (95.493)g
```

I

identity()

Katalog >

identity(*Integer***)** ⇒ *matrix*

Ger identitetsmatrisen med dimensionen *Integer*.

Integer måste vara positivt heltal.

```
identity(4)                        [1 0 0 0]
                                   [0 1 0 0]
                                   [0 0 1 0]
                                   [0 0 0 1]
```

If

Katalog >

If *BooleanExpr*
  Påståenden

If *BoolesktUttr* **Then**
  Block
EndIf

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

Om *BooleanExpr* är sant och exekverar sedan det enstaka påståendet *Statement* eller blocket av påståenden *Block* innan exekveringen fortsätter.

Om *BooleanExpr* är falskt, fortsätter exekveringen utan att exekvera påståendet eller blocket av påståenden.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":" .

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

If *BoolesktUttr* **Then**
    Block1
Else
    Block2
EndIf

Om *BooleanExpr* är sant, exekverar *Block1* och hoppar sedan över *Block2*.

Om *BooleanExpr* är falskt, hoppas *Block1* över och *Block2* exekveras.

Block1 och *Block2* kan vara enstaka påståenden.

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

If *BoolesktUttr1* **Then**
    Block1
ElseIf *BoolesktUttr2* **Then**
    Block2
⋮
ElseIf *BoolesktUttrN* **Then**
    BlockN
EndIf

Medger förgrening. If *BooleanExpr1* utvärderar till sant och exekverar *Block1*. If *BooleanExpr1* utvärderar till falskt, utvärderar *BooleanExpr2*, osv.

```
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( ) Katalog >

ifFn(*BooleanExpr*,*Value_If_true* [,*Value_If_false* [,*Value_If_unknown*]]**)** ⇒ *uttryck, lista eller matris*

Utvärderar det booleska uttrycket *BooleanExpr* (eller varje element från *BooleanExpr*) och producerar ett resultat baserat på följande regler:

- *BooleanExpr* kan testa ett enstaka värde, en lista eller en matris.
- Om ett element i *BooleanExpr* utvärderas som sant erhålls motsvarande element från *Value_If_true*.
- Om ett element i *BooleanExpr* utvärderas som falskt erhålls motsvarande element från *Value_If_false*. Om du utelämnar *Value_If_false* erhålls undef.
- Om ett element i *BooleanExpr* är varken sant eller falskt erhålls motsvarande element från *Value_If_unknown*. Om du utelämnar *Value_If_unknown* erhålls undef.
- Om det andra, tredje eller fjärde argumentet i funktionen **ifFn()** är ett enstaka uttryck tillämpas det booleska testet på varje position i *BooleanExpr*.

Obs: Om det förenklade påståendet *BooleanExpr* inbegriper en lista eller matris måste alla övriga list- eller matrisargument ha samma dimensioner, och resultatet får då samma dimensioner.

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

Testvärdet på **1** är mindre än 2.5, varför dess motsvarande

Value_If_True element **5** kopieras till resultatlistan.

Testvärdet på **2** är mindre än 2.5, varför dess motsvarande

Value_If_True element **6** kopieras till resultatlistan.

Testvärdet på **3** är inte mindre än 2,5, varför dess motsvarande *Value_If_False* element **10** kopieras till resultatlistan.

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

Value_If_true är ett enstaka värde och motsvarar varje vald position.

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

Value_If_false är ej specificerat. Undef används.

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

Ett valt element från *Value_If_true*. Ett valt element från *Value_If_unknown*.

imag() Katalog >

imag(*Value1***)** ⇒ *värde*

Ger argumentets imaginärdel.

$\text{imag}(1+2\cdot i)$ 2

imag(*List1***)** ⇒ *lista*

Ger en lista på elementens imaginärdelar.

$\text{imag}(\{-3,4-i,i\})$ $\{0,-1,1\}$

imag() Katalog >

imag(*Matrix1***)** ⇒ *matris*

Ger en matris med elementens imaginärdelar.

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

Indirection Se #(), på sidan 192.

inString() Katalog >

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

Ger teckenpositionen i strängen *srcString* där den första förekomsten av strängen *subString* börjar.

Start, om inkluderad, specificerar teckenpositionen inom *srcString* där sökningen börjar. Förinställning = 1 (det första tecknet i strängen *srcString*).

Återgår till noll om *srcString* inte innehåller *subString* eller om *Start* har en större längd än *srcString*.

```
inString("Hello there","the")        7
inString("ABCEFG","D")               0
```

int() Katalog >

int(*Value***)** ⇒ *heltal*
int(*List1***)** ⇒ *lista*
int(*Matrix1***)** ⇒ *matris*

Ger det största heltalet som är mindre än eller lika med argumentet. Denna funktion är identisk med **floor()**.

Argumentet kan vara ett reellt eller ett komplext tal.

Ger, för en lista eller matris, det största heltalet för varje element.

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

intDiv() Katalog >

intDiv(*Number1*, *Number2***)** ⇒ *heltal*
intDiv(*List1*, *List2***)** ⇒ *lista*
intDiv(*Matrix1*, *Matrix2***)** ⇒ *matris*

Ger heltalsdelen med tecken av (*Number1* ÷ *Number2*).

Ger, för listor och matriser, heltalsdelen med tecken av (argument1 ÷ argument2) för varje elementpar.

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

Interpolera () Katalog >

Interpolera(*xValue*, *xList*, *yList*, *yPrimeList***)** ⇒ *lista*

Denna funktion gör följande:

Förutsatt *xList*, *yList*=**f(***xList***)** och *yPrimeList*=**f'(***xList***)** används för en okänd funktion **f** en kubisk interpolant för att uppskatta funktionen **f** vid *xValue*. Det förutsätts att *xList* är en lista på monotont ökande eller minskande tal, men denna funktion kan ge ett värde även när listan inte uppfyller förutsättningarna. Denna funktion går igenom *xList* och söker ett intervall [*xList*[i], *xList*[i+1]] som innehåller *xValue*. Om funktionen hittar ett sådant intervall ger den ett interpolerat värde på **f(***xValue***)**, annars ger den **undef.**

xList, *yList* och *yPrimeList* måste ha samma dimension ≥2 och innehålla uttryck som förenklas till tal.

xValue kan vara ett tal eller en lista av tal.

Differentialekvation:
y'=-3•*y*+6•*t*+5 och *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  10
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Använd funktionen interpolate() för att beräkna funktionsvärdena för xvaluelist:

```
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
```

invχ^2() Katalog >

invχ^2(*Area*,*df***)**

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

Beräknar den inversa kumulativa sannolikhetsfunktionen χ^2 (chi-kvadrat) specificerad av frihetsgraden *df* för en given *Area* under kurvan.

invF() Katalog >

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

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

beräknar den inversa kumulativa fördelningsfunktionen F specificerad av *dfNumer* och *dfDenom* för en given *Area* under kurvan.

invBinom() Katalog >

invBinom (*CumulativeProb***,***NumTrials***,***Prob***,** *OutputForm***)⇒** *skalär* eller *matris*

Baserat på antalet försök (*NumTrials*) och sannolikheten för önskat utfall av varje försök (*Prob*) ger denna funktion det minimala antalet lyckade utfall, *k*, så att den kumulativa sannolikheten,*k*, är större än eller lika med den givna kumulativa sannolikheten (*CumulativeProb*).

OutputForm=**0**, visar resultatet som en skalär (förvalt).

OutputForm=**1**, visar resultatet som en matris.

Exempel: Marie and Kalle spelar ett tärningsspel. Marie ska gissa hur många gånger som mest tärningen visar en sexa under 30 kast. Om tärningen ger en sexa detta antal gånger eller mindre, vinner Marie. Dessutom får Marie högre poäng ju mindre antal sexor hon gissar. Vilket är det minsta antal gånger Marie kan gissa om hon vill att sannolikheten att vinna ska vara högre än 77 %?

$\text{invBinom}\left(0.77,30,\frac{1}{6}\right)$	6
$\text{invBinom}\left(0.77,30,\frac{1}{6},1\right)$	$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

invBinomN() Katalog >

invBinomN(*CumulativeProb***,***Prob***,** *NumSuccess,OutputForm***)⇒** *skalär* eller *matris*

Baserat på sannolikheten för lyckat utfall i varje försök (*Prob*) och antalet lyckade försök (NumSuccess) beräknar funktionen det minsta antalet försök, *N*, så att den kumulativa sannolikheten för *x* lyckade utfall är mindre eller lika med den givna kumulativ sannolikheten (*CumulativeProb*).

OutputForm=**0**, visar resultatet som en skalär (förvalt).

OutputForm=**1**, visar resultatet som en matris.

Exempel: Monika övar målskott i basket. Hon vet av erfarenhet att chansen att göra mål är 70 % i varje skott. Hon bestämmer sig för att öva tills hon har gjort 50 mål. Hur många försök måste hon göra för att sannolikheten för att göra minst 50 mål ska vara högre än 0,99?

$\text{invBinomN}(0.01,0.7,49)$	86
$\text{invBinomN}(0.01,0.7,49,1)$	$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$

invNorm() Katalog >

invNorm(*Area*[,μ[,σ]]**)**

Beräknar den inversa kumulativa normalfördelningen för en given *Area* under normalfördelningskurvan specificerad av μ och *σ*.

invt() Katalog >

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

Beräknar den inversa kumulativa student-t-fördelningsfunktionen specificerad av frihetsgraden, *df*, för en given *Area* under kurvan.

iPart() Katalog >

iPart(*Number***)** ⇒ *heltal*
iPart(*List1***)** ⇒ *lista*
iPart(*Matrix1***)** ⇒ *matris*

Ger argumentets heltalsdel.

Ger, för listor och matriser, heltalsdelen för varje element.

Argumentet kan vara ett reellt eller ett komplext tal.

iPart(⁻1.234)	⁻1.
iPart({3/2,⁻2.3,7.003})	{1,⁻2.,7.}

irr() Katalog >

irr(*CF0*,*CFList* [,*CFFreq*]**)** ⇒ *värde*

Ekonomifunktion som beräknar internräntan på en investering.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet CF0.

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

irr() Katalog >

CFFreq är en frivillig lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal <10 000.

Obs: Se även **mirr()**, på sidan 95.

isPrime() Katalog >

isPrime(*Number***)** ⇒ *Booleskt konstantuttryck*

Ger sant eller falskt för att indikera om *number* är ett heltal ≥2 som är jämnt delbart endast med sig självt och 1.

Om *Number* överskrider cirka 306 siffror och saknar faktorer ≤1021, visar **isPrime (***Number***)** ett felmeddelande.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

isPrime(5)	true
isPrime(6)	false

Funktion för att hitta nästa primtal efter ett specificerat tal:

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

isVoid() Katalog >

isVoid(*Var***)** ⇒ *Booleskt konstantuttryck*
isVoid(*Expr***)** ⇒ *Booleskt konstantuttryck*
isVoid(*List***)** ⇒ *lista av Booleska konstantuttryck*

Ger sant eller falskt för att indikera huruvida argumentet är en tom datatyp.

För mer information om tomma element, se på sidan 200.

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

Lbl

Katalog >

Lbl *labelName*

Definierar en etikett med namnet *labelName* inom en funktion.

Du kan använda en **Goto** *labelName*-instruktion för att överföra kontroll till instruktionen direkt efter etiketten.

labelName måste uppfylla samma krav på namngivning som ett variabelnamn.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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()

Katalog >

lcm(*Number1***,** *Number2***)⇒***uttryck*

lcm(*List1***,** *List2***)⇒***lista*

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

Ger den minsta gemensamma multipeln för de två argumenten. **lcm** för två bråk är **lcm** för deras täljare dividerat med **gcd** för deras nämnare. **lcm** för tal i flyttalsform är deras produkt.

Ger, för två listor eller matriser, den minsta gemensamma multipeln för de motsvarande elementen.

```
lcm(6,9)                                   18
lcm({1/3,-14,16},{2/15,7,5})   {2/3,14,80}
```

left()

Katalog >

left(*sourceString***[,** *Num***])⇒***sträng*

Ger *Num*-tecknen längst till vänster i teckensträngen *sourceString*.

Om du utelämnar *Num* erhålls alla i *sourceString*.

```
left("Hello",2)                            "He"
```

left(*List1***[,** *Num***])⇒***lista*

```
left({1,3,-2,4},3)                   {1,3,-2}
```

left() Katalog >

Ger *Num*-elementen längst till vänster i *List1*.

Om du utelämnar *Num* erhålls alla i *List1*.

left(*Comparison***)⇒***uttryck*

Ger den vänstra sidan av en ekvation eller olikhet.

libShortcut() Katalog >

libShortcut(*LibNameString***,** *ShortcutNameString*

[**,** *LibPrivFlag*]**)⇒***lista på variabler*

Skapar en variabelgrupp i det aktuella problemet som innehåller referenser till alla objekt i det specificerade biblioteksdokumentet *libNameString*. Lägger också till gruppmedlemmarna på menyn Variables. Du kan sedan referera till varje objekt med hjälp av dess *ShortcutNameString*.

Ställ *LibPrivFlag*=**0** för att utesluta privata biblioteksobjekt (förinställning)

Ställ *LibPrivFlag*=**1** för att inkludera privata biblioteksobjekt

För att kopiera en variabelgrupp, se **CopyVar**, på sidan 24.

För att ta bort en variabelgrupp, se **DelVar**, på sidan 39.

Detta exempel förutsätter ett korrekt lagrat och uppdaterat biblioteksdokument med namnet **linalg2** och som innehåller objekt definierade som *clearmat*, *gauss1* och *gauss2*.

```
getVarInfo("linalg2")
        [clearmat  "FUNC"  "LibPub "
         gauss1    "PRGM"  "LibPriv "
         gauss2    "FUNC"  "LibPub "]
libShortcut("linalg2","la")
                {la.clearmat,la.gauss2}
libShortcut("linalg2","la",1)
                {la.clearmat,la.gauss1,la.gauss2}
```

LinRegBx Katalog >

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

Utför den linjära regressionsanalysen y = a+b ·xpå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

LinRegBx Katalog >

X och Y är listor på oberoende och beroende variabler.

$Freq$ är en frivillig lista på frekvensvärden. Varje element i $Freq$ specificerar frekvensen för varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

$Category$ är en lista på numeriska eller strängkategorikoder för motsvarande X- och Y-data.

$Include$ är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a+b ·x
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade $X\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.YReg	Lista på datapunkter i den modifierade $Y\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.FreqReg	Lista på frekvenser som motsvarar $stat.XReg$ och $stat.YReg$

LinRegMx Katalog >

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

Beräknar den linjära regressionen y = m ·x+b på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln $stat.results$. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: m ·x+b
stat.m, stat.b	Regressionskoefficienter
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

LinRegtIntervals

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

För Slope (Lutning). Beräknar ett nivå-C-konfidensintervall för lutningen.

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

För Response (Svar). Beräknar ett prognostiserat y-värde, ett nivå-C-prediktionsintervall för en enstaka observation och ett nivå-C-konfidensintervall för medelvärdet på svaret.

En sammanfattning av resultaten visas i variabeln $stat.results$. (Se på sidan 148.)

Alla listor måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

F är en frivillig lista på frekvensvärden. Varje element i F specificerar förekomsten av varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: a+b ·x
stat.a, stat.b	Regressionskoefficienter
stat.df	Frihetsgrader
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen

Endast för typen Slope

Resultatvariabel	**Beskrivning**
[stat.CLower, stat.CUpper]	Konfidensintervall för lutningen
stat.ME	Konfidensintervall - felmarginal
stat.SESlope	Standardfel hos lutning
stat.s	Standardfel hos linjen

Endast för typen Response

Resultatvariabel	Beskrivning
[stat.CLower, stat.CUpper]	Konfidensintervall för medelvärdet på svaret
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelsvar
[stat.LowerPred, stat.UpperPred]	Prediktionsintervall för en enstaka observation
stat.MEPred	Prognostiseringsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.$\hat{y}$	a + b ·XVal

LinRegtTest

Katalog >

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

Utför en linjär regressionsanalys på listorna *X* och *Y* och ett *t*-test på lutningens värde β samt korrelationskoefficienten ρ för ekvationen $y=\alpha+\beta x$. Det testar nollhypotesen $H_0:\beta=0$ (equivalently, $\rho=0$) mot en av tre alternativa hypoteser.

Alla listor måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Hypoth är ett valfritt värde som specificerar en av tre alternativa hypoteser mot vilka nollhypotesen ($H_0:\beta=\rho=0$) kommer att testas.

För $H_a: \beta\neq0$ och $\rho\neq0$ (förinställning), ställ *Hypoth*=0

För $H_a: \beta<0$ och $\rho<0$, ställ *Hypoth*<0

För $H_a: \beta>0$ och $\rho>0$, ställ *Hypoth*>0

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

LinRegtTest

Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a + b ·x
stat.t	*t*-Statistik för signifikanstest
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader
stat.a, stat.b	Regressionskoefficienter
stat.s	Standardfel hos linjen
stat.SESlope	Standardfel hos lutning
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen

linSolve()

Katalog >

linSolve(*SystemAvLinjäraEkv*, *Var1*, *Var2*, ...**)⇒***lista*

linSolve(*LinjärEkv1* **och** *LinjärEkv2* **och** ..., *Var1*, *Var2*, ...**)⇒***lista*

linSolve({*LinjärEkv1*, *LinjärEkv2*, ...**},** *Var1*, *Var2*, ...**) ⇒***lista*

linSolve(*SystemAvLinjäraEkv*, **{***Var1*, *Var2*, ...**}) ⇒***lista*

linSolve(*LinjärEkv1* **och** *LinjärEkv2* **och** ..., **{***Var1*, *Var2*, ...**})⇒***lista*

linSolve({*LinjärEkv1*, *LinjärEkv2*, ...**},** **{***Var1*, *Var2*, ...**}) ⇒***lista*

Ger en lista på lösningar för variablerna *Var1*, *Var2*, ...

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

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

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

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

linSolve() Katalog >

Det första argumentet måste beräknas till ett system av linjära ekvationer eller en enda linjär ekvation. Annars inträffar ett argumentfel.

Som exempel leder beräkningen **`linSolve(x=1 och x=2,x)`** till ett "Argumentfel"-resultat.

ΔList() Katalog >

ΔList(*List1***)⇒***lista*

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

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`deltaList(...)`**.

Ger en lista på skillnaderna mellan konsekutiva element i *List1*. Varje element i *List1* subtraheras från nästa element i *List1*. Den resulterande listan är alltid ett element kortare än den ursprungliga *List1*.

list▸mat() Katalog >

list▸mat(*List* **[,** *elementsPerRow***])** ⇒*matris*

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

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

Ger en matris fylld rad efter rad med elementen från *List*.

elementsPerRow, om inkluderad, specificerar antalet element per rad. Förinställningen är antalet element i *List* (en rad).

Om *List* inte fyller den resulterande listan läggs nollor till.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`list@>mat(...)`**.

ln() ctrl e^x tangenter

ln(*Value1***)⇒***värde*

$\ln(2.)$ 0.693147

ln(*List1***)⇒***lista*

ln()

ctrl e^x **tangenter**

Ger argumentets naturliga logaritm.

Ger, för en lista, elementens naturliga logaritmer.

Om det komplexa formatläget är Real:

```
ln({-3,1.2,5})
                    "Error: Non-real calculation"
```

Om det komplexa formatläget är Rectangular:

```
ln({-3,1.2,5})
    {1.09861+3.14159·i,0.182322,1.60944}
```

ln(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med naturlig logaritm för *squareMatrix1*. Detta är inte detsamma som att beräkna den naturliga logaritmen för varje element. För information om beräkningsmetoden, se **cos()**.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

```
ln([1  5  3
    4  2  1
    6 -2  1])
[ 1.83145+1.73485·i    0.009193-1.49086
 0.448761-0.725533·i  1.06491+0.623491▸
-0.266891-2.08316·i   1.12436+1.79018·
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

LnReg

Katalog >

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

Utför en en logaritmisk regressionsanalys y = a+b ·ln(x) på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

LnReg

Katalog >

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a+b ·ln(x)
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data (ln(x), y)
stat.Resid	Residualer associerade med den logaritmiska modellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Local

Katalog >

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

Betecknar specificerade *vars* som lokala variabler. Dessa variabler existerar endast under utvärderingen av ett uttryck och tas bort när exekveringen av uttrycket är klar.

Obs: Lokala variabler sparar minne eftersom de endast existerar tillfälligt. De stör heller inga befintliga globala variabelvärden. Lokala variabler måste användas för **For**-slingor och för att temporärt spara värden i en flerradig funktion eftersom modifieringar av globala värden inte är tillåtna i en funktion.

```
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
```

Local

Katalog >

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Lock

Katalog >

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

Lock *Var*.

Låser den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.

Du kan inte låsa eller låsa upp systemvariabeln *Ans* och du kan inte låsa systemvariabelgrupperna *stat*. och *tvm*.

Obs: Kommandot **Lås** (**Lock**) rensar Ångra/Upprepa-historiken när det används på olåsta variabler.

Se **unLock**, på sidan 166 och**getLockInfo()**, på sidan 65.

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 tangenter

log(*Value1*[,*Value2*]**)⇒***värde*

log(*List1*[,*Value2*]**)⇒***lista*

Ger bas-*Value2*-logaritmen för det första argumentet.

Obs: Se även **Log template**, på sidan 2.

Ger, för en lista, bas-*Value2*-logaritmen för elementen.

Om det andra argumentet utelämnas används 10 som bas.

$\log_{10}(2.)$	0.30103
$\log_{4}(2.)$	0.5
$\log_{3}(10)-\log_{3}(5)$	0.63093

Om det komplexa formatläget är Real:

$\log_{10}(\{-3,1.2,5\})$	"Error: Non−real calculation"

Om det komplexa formatläget är Rectangular:

log() ctrl 10^x tangenter

$$\log_{10}(\{-3,1.2,5\})$$
$$\{0.477121+1.36438\cdot i, 0.079181, 0.69897\}$$

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

Ger matrisen med bas-*Value*-logaritm för *squareMatrix1*. Detta är inte detsamma som att beräkna bas-*Value*-logaritmen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

Om basargumentet utelämnas används 10 som bas.

I vinkelläget Radianer och i Rektangulärt komplext format:

$$\log_{10}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} 0.795387+0.753438\cdot i & 0.003993-0.6474 \\ 0.194895-0.315095\cdot i & 0.462485+0.2707 \\ -0.115909-0.904706\cdot i & 0.488304+0.7774 \end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Logistic Katalog >

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

Utför den logistiska regressionsanalyseny = (c/(1+a ·e^{-bx}))på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Logistic

Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

LogisticD

Katalog >

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

Utför den logistiska regressionsanalysen $y = (c/(1+a \cdot e^{-bx})+d)$ på listorna *X* och *Y* med frekvensen *Freq*, med ett specificerat antal *Iterationer*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Iterations är ett valfritt värde som specificerar det maximala antalet gånger en lösning kommer att provas. Om denna utelämnas används 64. Normalt ger större värden bättre noggrannhet, men längre exekveringstider, och vice versa.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

LogisticD

Katalog >

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $c/(1+a \cdot e^{-bx})+d$)
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Loop

Katalog >

Loop

Block

EndLoop

Exekverar påståendena i *Block* upprepade gånger. Observera att slingan upprepas i all oändlighet såvida inte en **Goto**- eller **Exit**-instruktion exekveras inom *Block*.

Block är en serie av påståenden separerade med tecknet “:”.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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 Katalog >

LU *Matris*, *lMatris*, *uMatris*, *pMatris[,Tol]*

Beräknar uppdelningen Doolittle LU (undre-övre) av en reell eller komplex matris. Den undertriangulära matrisen lagras i *lMatris*, den övertriangulära matrisen lagras i *uMatris* och permutationsmatrisen (som beskriver radväxlingarna som har gjorts under beräkningen) lagras i *pMatris*.

lMatris · *uMatris* = *pMatris* · *matris*

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning med tal i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E-14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

Faktoriseringsalgoritmen **LU** använder partiell pivotering med radutbyten.

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

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() Katalog >

mat▸list(*Matrix***)⇒***lista*

Ger en lista med elementen i *Matrix*. Elementen kopieras från *Matrix* rad för rad.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `mat@>list(…)`.

mat▸list([1 2 3]) — {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() Katalog >

max(*Value1***, *Value2***)⇒***uttryck***

max(*List1***, *List2***)⇒***lista***

max(*Matrix1***, *Matrix2***)⇒***matris***

Ger de två argumentens maximum. Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller maximumvärdet för varje par av motsvarande element.

$\max(2.3,1.4)$ → 2.3

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

max(*List***)⇒***uttryck***

Ger maximumelementet i *list*.

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matrix1***)⇒***matris***

Ger en radvektor som innehåller maximumelementet för varje kolumn i *Matrix1*.

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

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

Obs: Se även **min()**.

mean() Katalog >

mean(*List***[, *freqList***])⇒***uttryck***

Ger medelvärdet för elementen i *List*.

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

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

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

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

Ger en radvektor med medelvärdena för alla kolumner i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

I vektorformatet Rectangular:

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

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

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

median() Katalog >

median(*List*[**,** *freqList*]**)⇒***uttryck*

Ger medianen för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

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

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

Ger en radvektor som innehåller medianerna för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

$$\text{median}\left(\begin{bmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{bmatrix}\right) \qquad \begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$$

Obs:

- Alla inmatningar i listan eller matrisen måste förenklas till tal.
- Tomma element i listan eller matrisen ignoreras. För mer information om tomma element, se på sidan 200.

MedMed Katalog >

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

Beräknar median-median-linjeny = (m ·x+b) på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

MedMed

Katalog >

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Ekvation för median-median-linje m ·x+b
stat.m, stat.b	Modellkoefficienter
stat.Resid	Residualer från median-median-linjen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

mid()

Katalog >

mid(*sourceString***, *Start*[, *Count*])⇒***sträng***

Ger *Count*-tecknen från teckensträngen *sourceString* och börjar med teckennumret *Start*.

Ger, om *Count* utelämnas eller är större än dimensionen på *sourceString*, alla tecken från *sourceString* med start från teckennumret *Start*.

Count måste vara ≥ 0. Om *Count* = 0 erhålls en tom sträng.

```
mid("Hello there",2)        "ello there"
mid("Hello there",7,3)      "the"
mid("Hello there",1,5)      "Hello"
mid("Hello there",1,0)      "⬚"
```

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

Ger *Count*-elementen från *sourceList* och börjar med elementnumret *Start*.

Ger, om *Count* utelämnas eller är större än dimensionen på *sourceList*, alla element från *sourceList* med start från elementnumret *Start*.

Count måste vara ≥ 0. Om Count = 0 erhålls en tom lista.

```
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() Katalog >

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

Ger *Count*-strängarna från stränglistan *sourceStringList* och börjar med elementnumret *Start*.

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

min() Katalog >

min(*Value1***,** *Value2***)**⇒*uttryck*

min(*List1, List2***)**⇒*lista*

min(*Matrix1, Matrix2***)**⇒*matris*

Ger de två argumentens minimum. Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller minimumvärdet för varje par av motsvarande element.

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

min(*List***)**⇒*uttryck*

Ger minimumelementet för *List*.

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

min(*Matrix1***)**⇒*matris*

Ger en radvektor som innehåller minimumelementet för varje kolumn i *Matrix1*.

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

Obs: Se även **max()**.

mirr() Katalog >

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

Finansiell funktion som beräknar den modifierade internräntan på en investering.

financeRate är den räntesats som du betalar på kassaflödesbeloppen.

reinvestRate är den räntesats vid vilken kassaflödena återinvesteras.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

```
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
```

mirr() Katalog >

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet CF0.

CFFreq är en frivillig lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

Obs: Se även **irr()**, på sidan 75.

mod() Katalog >

mod(*Value1***,** *Value2***)⇒***uttryck*

mod(*List1***,** *List2***)⇒***lista*

mod(*Matrix1***,** *Matrix2***)⇒***matris*

Ger det första argumentet modulo det andra argumentet definierat av identiteterna:

mod(x,0) = x

mod(x,y) = x − y floor(x/y)

När det andra argumentet är skilt från noll är resultatet periodiskt i det argumentet. Resultatet är antingen noll eller har samma tecken som det andra argumentet.

Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller modulen för varje par av motsvarande element.

Obs: Se även **remain()**, på sidan 126

$\text{mod}(7,0)$	7
$\text{mod}(7,3)$	1
$\text{mod}(-7,3)$	2
$\text{mod}(7,-3)$	-2
$\text{mod}(-7,-3)$	-1
$\text{mod}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,-4\}$

mRow() Katalog >

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

Ger en kopia av *Matrix1* med varje element i rad *Index* i *Matrix1* multiplicerat med *Value*.

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

mRowAdd() Katalog >

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

Ger en kopia av *Matrix1* med varje element i rad *Index2* i *Matrix1* ersatt med:

Value · *rad Index1* + *rad Index2*

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

MultReg Katalog >

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

Beräknar den multipla linjära regressionen i lista *Y* på listorna *X1*, *X2*, ..., *X10*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200.

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ ...
stat.b0, stat.b1, ...	Regressionskoefficienter
stat.R^2	Koefficient för multipel bestämning
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ ...
stat.Resid	Residualer från regressionsanalysen

MultRegIntervals Katalog >

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

Beräknar ett prognostiserat y-värde, ett nivå-C-prediktionsintervall för en enstaka observation och ett nivå-C-konfidensintervall för medelvärdet på svaret.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor måste ha samma dimensioner.

MultRegIntervals Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ ...
stat.ŷ	En punktuppskattning: ŷ = b0 + b1 · xl + ... för *XValList*
stat.dfError	Fel hos frihetsgrader
stat.CLower, stat.CUpper	Konfidensintervall för ett medelvärde på svaret
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelvärdet på svaret
stat.LowerPred, stat.UpperrPred	Prediktionsintervall för en enstaka observation
stat.MEPred	Prediktionsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.bList	Lista på regressionskoefficienter, {b0,b1,b3,...}
stat.Resid	Residualer från regressionen

MultRegTests Katalog >

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

Ett multipelt linjärt regressionstest utför en multipel linjär regressionsanalys på givna data och ger den globala *F*-teststatistiken och *t*-teststatistiken för koefficienterna.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Utdata

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ ...
stat.F	Global *F*-teststatistik

Resultatvariabel	Beskrivning
stat.PVal	P-värde associerat med global F-statistik
stat.R^2	Koefficient för multipel bestämning
stat.AdjR^2	Justerad koefficient för multipel bestämning
stat.s	Standardavvikelse hos felet
stat.DW	Durbin-Watson-statistik: används för att bestämma om modellen innehåller autokorrelation av första ordningen
stat.dfReg	Frihetsgrader hos regressionen
stat.SSReg	Regressionens kvadratsumma
stat.MSReg	Regression medelkvadrat
stat.dfError	Fel hos frihetsgrader
stat.SSError	Felens kvadratsumma
stat.MSError	Felens medelkvadrat
stat.bList	{b0,b1,…} Lista på koefficienter
stat.tList	Lista på t-statistik för varje koefficient i bList
stat.PList	Lista på P-värden för varje t-statistik
stat.SEList	Lista på standardfel för koefficienter i bList
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ . . .
stat.Resid	Residualer från regressionen
stat.sResid	Standardiserade residualer: erhållna genom att dividera en residual med dess standardavvikelse
stat.CookDist	Cooks avstånd: mått på den influens som en observation har, baserat på residual och stigning
stat.Leverage	Mått på hur långt värdena i den oberoende variabeln är från sina medelvärden

N

nand

ctrl = **knappar**

BooleskUttr1 **nand** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **nand** *BooleskLista2* ger *Boolesk lista*

nand

ctrl = knappar

*BooleskMatris1***nand***BooleskMatris2* ger *Boolesk matris*

Ger negation av en logisk **and** uppgift på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

*Heltal1***nand***Heltal2*⇒*heltal*

Jämför två reella heltal bit för bit med en **nand**-operation Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs är resultatet 0 om båda bitarna är 1; annars är resultatet 1. Det returnerade värdet representerar bitresultaten och visas enligt basläget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

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

nCr()

Katalog >

nCr(*Value1***,** *Value2***)**⇒*uttryck*

För heltal *Value1* och *Value2* med *Value1* ≥ *Value2* ≥ 0 är **nCr()** antalet kombinationer av *Value1* tagna *Value2* åt gången. (Detta kallas också en binomial koefficient.)

nCr(*Value***, 0)⇒1**

nCr(*Value***,** *negInteger***)⇒0**

nCr(*Value***,** *posInteger***)⇒** *Value* · **(***Value*−1**)... (***Value*−*posInteger***+1) /***posInteger***!**

nCr(*Value***,** *nonInteger***)⇒***expression***!/ ((***Value*−*nonInteger***)! ·***nonInteger***!)**

nCr(*List1***,** *List2***)⇒***lista*

nCr(z,3)|z=5	10
nCr(z,3)|z=6	20
nCr({5,4,3},{2,4,2})	{10,1,3}

nCr() Katalog >

Ger en lista på kombinationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.

nCr(*Matrix1***, *Matrix2***)⇒***matris*

Ger en matris över kombinationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.

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

nDerivative() Katalog >

nDerivative(*Uttr1***,***Var***=***Värde***[,***Ordning***])** ⇒*värde*

nDerivative(*Uttr1***,***Var***[,***Ordning***]) |** *Var***=***Värde***⇒***värde*

Ger den numeriska derivatan beräknad med automatiska deriveringsmetoder.

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Om variabeln *Var* inte innehåller ett numeriskt värde måste du tillhandahålla *Värde*.

Ordning för derivatan måste vara **1** eller **2**.

$$\mathrm{nDerivative}(|x|,x=1) \qquad 1$$

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

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

Obs: Algoritmen **nDerivative()** har en begränsning: den arbetar rekursivt genom det ej förenklade uttrycket och beräknar det numeriska värdet för förstaderivatan (och andraderivatan om tillämpligt) samt beräknar varje deluttryck, vilket kan leda till ett oväntat resultat.

Studera exemplet till höger. Förstaderivatan av x·(x^2+x)^(1/3) för x=0 är lika med 0. Men eftersom förstaderivatan av deluttrycket (x^2+x)^(1/3) är odefinierat för x=0, och detta värde används för att beräkna derivatan av hela uttrycket, anger **nDerivative()** resultatet som odefinierat och visar ett varningsmeddelande.

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

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

nDerivative() Katalog >

Om du stöter på denna begränsning, verifiera lösningen grafiskt. Du kan också prova att använda**centralDiff()**.

newList() Katalog >

newList(*numElements***)⇒***lista*

Ger en lista med dimensionen på *numElements*. Varje element är noll.

newList(4)	{0,0,0,0}

newMat() Katalog >

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

Ger en matris med nollor med dimensionen *numRows* gånger *numColumns*.

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

nfMax() Katalog >

nfMax(*Expr***,** *Var***)⇒***värde*

nfMax(*Expr***,** *Var, undrGräns***)⇒***värde*

nfMax(*Expr***,** *Var, undrGräns***,** *övrGräns***)** ⇒*värde*

nfMax(*Expr, Var***) |** *undrGräns*≤*Var* ≤*övrGräns*⇒*värde*

Ger ett möjligt numeriskt värde på variabeln *Var* där lokalt maximum för *Expr* inträffar.

Om du inför *undrGräns* och *övrGräns* söker funktionen i det stängda intervallet [*undrGräns*,*övrGräns*] efter lokalt maximum.

$\text{nfMax}(-x^2-2\cdot x-1,x)$	-1.
$\text{nfMax}(0.5\cdot x^3-x-2,x,-5,5)$	5.

nfMin() Katalog >

nfMin(*Expr***,** *Var***)⇒***värde*

nfMin(*Expr***,** *Var, undrGräns***)⇒***värde*

nfMin(*Expr***,** *Var, undrGräns***,** *övrGräns***)** ⇒*värde*

$\text{nfMin}(x^2+2\cdot x+5,x)$	-1.
$\text{nfMin}(0.5\cdot x^3-x-2,x,-5,5)$	-5.

nfMin() Katalog >

nfMin(*Expr, Var***) |** *undrGräns≤Var ≤övrGräns⇒värde*

Ger ett möjligt numeriskt värde på variabeln *Var* där lokalt minimum för *Expr* inträffar.

Om du inför *undrGräns* och *övrGräns* söker funktionen i det stängda intervallet [*undrGräns*,*övrGräns*] efter lokalt minimum.

nInt() Katalog >

nInt(*Expr1, Var, Lower, Upper***)⇒***uttryck*

Om integranden *Expr1* inte innehåller någon variabel utöver *Var*, och om *Lower* och *Upper* är konstanter, positiv ∞ eller negativ ∞, ger **nInt()** en uppskattning av ∫**(***Expr1*, *Var*, *Lower*, *Upper***)**. Denna uppskattning är ett vägt genomsnitt av vissa sampelvärden hos integranden i intervallet *Lower*<*Var*<*Upper*.

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

Målsättningen är sex signifikanta siffror. Den adaptiva algoritmen bestämmer när det verkar sannolikt att målet har uppnåtts, eller när det verkar osannolikt att ytterligare sampling ger en nämnvärd förbättring.

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

En varning (“Questionable accuracy”) visas när det verkar som om målet inte har uppnåtts.

Man kan kapsla in **nInt()** för att utföra multipel numerisk integrering. Integrationsgränser kan bero på integrationsvariabler utanför gränserna.

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

nom() Katalog >

nom(*effectiveRate,CpY***)⇒***värde*

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

Finansiell funktion som konverterar den årliga effektiva räntan *effectiveRate* till en nominell ränta, given av *CpY* som antalet sammansatta ränteperioder per år.

nom() Katalog >

effectiveRate måste vara ett reellt tal och *CpY* måste vara ett reellt tal > 0.

Obs: Se även **eff()**, på sidan 45.

nor ctrl = knappar

BooleskUttr1 **nor** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **nor** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **nor** *BooleskMatris2* ger *Boolesk matris*

Ger negation av en logisk **or** operation på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Heltal1 **nor** *Heltal2*⇒*heltal*

Jämför två reella heltal bit för bit med en **nor**-operation Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1, annars blir resultatet 0. Det erhållna värdet representerar bitresultatet och visas enligt Bas-läget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

3 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

norm() Katalog >

norm(*Matrix***)**⇒*uttryck*

norm(*Vector***)**⇒*uttryck*

Ger Frobenius norm.

$\text{norm}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$	5.47723
$\text{norm}\left(\begin{bmatrix}1 & 2\end{bmatrix}\right)$	2.23607
$\text{norm}\left(\begin{bmatrix}1\\ 2\end{bmatrix}\right)$	2.23607

normCdf() Katalog >

normCdf(*lowBound*,*upBound*[,μ[,σ]])⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten vid en normalfördelning mellan *lowBound* och *upBound* för specificerad μ (förinställning=0) och σ (förinställning=1).

För P(X ≤ *upBound*), sätt *lowBound* = −9**E**999.

normPdf() Katalog >

normPdf(*XVal*[,μ[,σ]])⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen för normalfördelning vid ett specificerat *XVal*-värde för specificerad μ och σ.

not (icke) Katalog >

not *BooleanExpr*⇒*Booleskt uttryck*

Ger en sann, falsk eller förenklad form av argumentet.

not (2≥3)	true
not 0hB0▸Base16	0hFFFFFFFFFFFFFF4F
not not 2	2

not *Integer1*⇒*heltal*

Ger ettkomplementet till ett reellt heltal. Internt omvandlas *Integer1* till ett 64-bitars binärt tal. Värdet på varje bit växlas (0 blir 1 och vice versa) för ettans komplement. Resultaten visas enligt det inställda basläget.

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

not 0h7AC36	0hFFFFFFFFFFF853C9

Du kan skriva in heltalet i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalet som ett decimalt tal (bas 10).

I binärt basläge:

0b100101▸Base10	37
not 0b100101	0b1111111111111111111111111111111▸
not 0b100101▸Base10	−38

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▸**Base2**, på sidan 16.

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

not (icke) Katalog >

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

nPr() Katalog >

nPr(*Value1***, *Value2***)⇒***uttryck*

För heltal *Value1* och *Value2* med *Value1* ≥ *Value2* ≥ 0 är **nPr()** antalet permutationer av *Value1* saker tagna *Value2* åt gången.

nPr(z,3)|z=5	60
nPr(z,3)|z=6	120
nPr({5,4,3},{2,4,2})	{20,24,6}
nPr([6 5; 4 3],[2 2; 2 2])	[30 20; 12 6]

nPr(*Value***, 0)⇒1**

nPr(*Value***, *negInteger***)⇒1/((***Value***+1)** · **(***Value***+2)**... **(***Value***−***negInteger***))**

nPr(*Value***, *posInteger***)⇒***Value* · **(***Value***−1)... (***Value***−***posInteger***+1)**

nPr(*Value***, *nonInteger***)⇒***Value***! / (***Value***−***nonInteger***)!**

nPr(*List1***, *List2***)⇒***lista*

Ger en lista på permutationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.

nPr({5,4,3},{2,4,2})	{20,24,6}

nPr(*Matrix1***, *Matrix2***)⇒***matris*

Ger en matris över permutationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.

nPr([6 5; 4 3],[2 2; 2 2])	[30 20; 12 6]

npv() Katalog >

npv(*InterestRate***,***CFO***,***CFList***[,***CFFreq***])**

Finansiell funktion som beräknar nettovärdet, dvs. summan av aktuella värden för kassainflöden och kassautflöden. Ett positivt resultat för npv indikerar en vinstgivande investering.

list1:={6000,⁻8000,2000,⁻3000}	{6000,⁻8000,2000,⁻3000}
list2:={2,2,2,1}	{2,2,2,1}
npv(10,5000,*list1*,*list2*)	4769.91

npv() Katalog >

InterestRate är räntan med vilken kassaflödena (kapitalanskaffningskostnaderna) diskonteras under en period.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet *CF0*.

CFFreq är en lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

nSolve() Katalog >

nSolve(*Equation*,*Var***[=***Guess***])**⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var***[=***Guess***]**,*lowBound***)** ⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var* **[=***Guess***]**,*lowBound*,*upBound***)** ⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var***[=***Guess***])** | *lowBound* ≤*Var*≤*upBound* ⇒*tal eller fel_sträng*

Söker iterativt efter en ungefärlig reell numerisk lösning på *Equation* för dess variabel. Specificera variabeln som:

variabel

– eller –

variabel = *reellt tal*

Som exempel är x giltigt och likaså x=3.

$\text{nSolve}\left(x^2+5\cdot x-25=9,x\right)$	3.84429
$\text{nSolve}\left(x^2=4,x=-1\right)$	-2.
$\text{nSolve}\left(x^2=4,x=1\right)$	2.

Obs: Om det finns flera lösningar kan du använda en gissning för att lättare hitta en viss lösning.

nSolve() Katalog >

nSolve() försöker att bestämma antingen en punkt där residualen är noll eller två relativt närliggande punkter där residualen har motsatta tecken och inte är överdrivet stor. Om detta inte kan uppnås med ett måttligt antal samplingspunkter erhålls strängen “no solution found”.

$\text{nSolve}\left(x^2+5\cdot x-25=9,x\right)\|x<0$	-8.84429
$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right)\|r>0 \text{ and } r<0.25$	0.006886
$\text{nSolve}\left(x^2=-1,x\right)$	"No solution found"

O

OneVar Katalog >

OneVar [**1,**]*X*[**,**[*Freq*][**,***Category***,***Include*]]

OneVar [*n***,**]*X1***,***X2*[*X3*[**,**...[**,***X20*]]]

Beräknar 1-variabelstatistik på upp till 20 listor. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X-argumenten är datalistor.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*-värde. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. Ett tomt element i någon av listorna *X1* till och med *X20* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 200.

Resultatvariabel	Beskrivning
stat.$\bar{x}$	Medelvärde av x-värden
stat.Σx	Summa av x-värden
stat.Σx^2	Summa av x^2-värden
stat.sx	Standardavvikelse för x (sampling)
stat. x	Standardavvikelse för x (population)
stat.n	Antal datapunkter
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.SSX	Kvadratsumma av avvikelser från medelvärdet på x

or (eller) Katalog >

BooleskUttr1 **or** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **or** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **or** *BooleskMatris2* ger *Boolesk matris*

Ger resultatet sant eller falskt eller en förenklad form av den ursprungliga inmatningen.

Ger sant om ettdera eller båda uttrycken förenklas till sant. Ger resultatet falskt om båda uttrycken utvärderas som falska.

Obs: Se **xor**.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
Define g(x)=Func                    Done
            If x≤0 or x≥5
            Goto end
            Return x·3
            Lbl end
            EndFunc
g(3)                                9
g(0)        A function did not return a value
```

Integer1 **or** *Integer2*⇒*heltal*

I hexadecimalt basläge:

```
0h7AC36 or 0h3D5F                   0h7BD7F
```

or (eller) Katalog >

Jämför två reella heltal bit för bit med en or-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1. Resultatet blir 0 endast om båda bitarna är 0. Det erhållna värdet representerar bitresultaten och visas enligt det inställda basläget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▶**Base2**, på sidan 16.

Obs: Se **xor.**

Viktigt: Noll, inte bokstaven O.

I binärt basläge:

0b100101 or 0b100	0b100101

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

ord() Katalog >

ord(*String***)⇒***integer*

ord(*List1***)⇒***lista*

Ger den numeriska koden för det första tecknet i teckensträngen *String* eller en lista på de första tecknen i varje listelement.

ord("hello")	104
char(104)	"h"
ord(char(24))	24
ord({"alpha","beta"})	{97,98}

P

P▶Rx() Katalog >

P▶Rx(*rExpr***, θ***Expr***)⇒***uttryck***

P▶Rx(*rList***, θ***List***)⇒***lista***

P▶Rx(*rMatrix***, θ***Matrix***)⇒***matris***

Ger den ekvivalenta x-koordinaten för paret (r, θ).

I vinkelläget Radianer:

P▶Rx(4,60°)	2.
P▶Rx({⁻3,10,1.3},{π/3,⁻π/4,0})	{⁻1.5,7.07107,1.3}

P▸Rx() Katalog >

Obs: Argumentet θ tolkas som en vinkel i antingen grader, nygrader eller i radianer beroende på det aktuella vinkelläget. Om argumentet är ett uttryck kan du använda °, ^{G} eller ^{r} för att tillfälligt överstyra vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **P@>Rx(...)**.

P▸Ry() Katalog >

P▸Ry(*rValue***, θ***Value***)⇒***värde***

P▸Ry(*rList***, θ***List***)⇒***lista***

P▸Ry(*rMatrix***, θ***Matrix***)⇒***matris***

Ger den ekvivalenta y-koordinaten för paret (r, θ).

Obs: Argumentet θ tolkas som en vinkel i antingen grader, radianer eller nygrader beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **P@>Ry(...)**.

I vinkelläget Radianer:

$$P\blacktriangleright Ry(4,60°) \qquad 3.4641$$

$$P\blacktriangleright Ry\left(\left\{-3,10,1.3\right\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right) \qquad \left\{-2.59808,-7.07107,0\right\}$$

PassErr Katalog >

PassErr

Flyttar ett fel till nästa nivå.

Om systemvariabeln *errCode* är noll utför **PassErr** ingenting.

Villkoret **Else** i blocket **Try...Else...EndTry** bör använda **ClrErr** eller **PassErr**. Om felet skall processas eller ignoreras, använd **ClrErr**. Om det är okänt hur felet skall hanteras, använd **PassErr** för att skicka felet vidare till nästa felhanterare. Om det inte finns någon ytterligare felhanterare för **Try...Else...EndTry** visas feldialogrutan som normal.

För ett exempel på **PassErr**, se exempel 2 under kommandot **Try**, på sidan 160.

PassErr

Katalog >

Obs: Se även **ClrErr**, på sidan 22 och **Try**, på sidan 159.

Kommentarer om inmatningen av exemplet: I applikationen Räknare kan du skriva in flerradiga definitioner genom att trycka på ↵ i stället för enter i slutet av varje linje. På datorns tangentbord, håll ned **Alt** och tryck på **Enter**.

piecewise()

Katalog >

piecewise(*Expr1* [**,** *Condition1* [**,** *Expr2* [**,** *Condition2* [**,** ...]]]]**)**

Ger definitioner för en stegvis funktion i form av en lista. Du kan också skapa stegvisa definitioner med hjälp av en mall.

Obs: Se även **Piecewise template**, på sidan 3.

Define $p(x)=\begin{cases} x, & x>0 \\ \text{undef}, & x\le 0 \end{cases}$	*Done*
$p(1)$	1
$p(-1)$	undef

poissCdf()

Katalog >

poissCdf(λ**,***lowBound*,*upBound***)**$\Rightarrow$*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

poissCdf(λ**,***upBound***)**(för P(0≤X≤*upBound*) $\Rightarrow$*tal* om *upBound* är ett tal, lista om *upBound* är en lista

Beräknar en kumulativ sannolikhet för den diskreta Poisson-fördelningen med det specificerade medelvärdet λ.

För P(X ≤ *upBound*), sätt *lowBound*=0

poissPdf()

Katalog >

poissPdf(λ**,***XVal***)**$\Rightarrow$*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar en sannolikhet för den diskreta Poisson-fördelningen med det specificerade medelvärdet λ.

▸Polar Katalog >

Vector ▸**Polar**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Polar**.

Visar *vector* i polär form [r∠ θ]. Vektorn måste ha dimensionen 2 och kan vara en rad eller en kolumn.

Obs: ▸**Polar** är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad, och den uppdaterar inte *ans*.

Obs: Se även ▸**Rect**, på sidan 123.

$[1 \quad 3.]$▸Polar $\quad [3.16228 \quad \angle 71.5651]$

complexValue ▸**Polar**

Visar *complexVector* i polär form.

- Vinkelläget Grader ger (r∠ θ).
- Vinkelläget Radianer ger $re^{i\theta}$.

complexValue kan ha valfri komplex form. En inmatning av $re^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

Obs: Du måste använda parenteserna för en (r∠ θ) polär inmatning.

I vinkelläget Radianer:

$(3+4\cdot i)$▸Polar $\quad e^{.927295\cdot i}\cdot 5$

$\left(\left\{4 \angle \frac{\pi}{3}\right\}\right)$▸Polar $\quad e^{1.0472\cdot i}\cdot 4.$

I vinkelläget Nygrader:

$(4\cdot i)$▸Polar $\quad (4\angle\ 100.)$

I vinkelläget Grader:

$(3+4\cdot i)$▸Polar $\quad (5 \angle\ 53.1301)$

polyEval() Katalog >

polyEval(*List1***,** *Expr1***)⇒***uttryck*

polyEval(*List1***,** *List2***)⇒***uttryck*

Tolkar det första argumentet som koefficienten för ett polynom med fallande ordning och ger polynomet utvärderat för det andra argumentets värde.

polyEval({1,2,3,4},2) $\quad 26$

polyEval({1,2,3,4},{2,-7}) $\quad \{26,-262\}$

polyRoots() Katalog >

polyRoots(*Poly***,***Var***)** ⇒*lista*

polyRoots(*ListaPåKoeff***)** ⇒*lista*

Den första syntaxen, **polyRoots(***Poly***,***Var***)**, ger en lista på reella rötter i polynomet *Poly* med avseende på variabeln *Var*. Om inga reella rötter existerar ger den en tom lista: { }.

Poly måste vara ett polynom i expanderad form i en variabel. Använd inte oexpanderade former såsom $y^2 \cdot y+1$ eller $x \cdot x+2 \cdot x+1$

Den andra syntaxen, **polyRoots (***ListaPåKoeff***)** ger en lista på reella rötter för koefficienterna i *ListaPåKoeff*.

Obs: Se även **cPolyRoots()**, på sidan 31.

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}
polyRoots({1,2,1})	{-1,-1}

PowerReg Katalog >

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

Utför potensregressionsanalys $y = (a \cdot (x)^b)$på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

PowerReg Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·(x)^b
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data (ln(x), ln(y))
stat.Resid	Residualer associerade med potensmodellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Prgm Katalog >

Prgm

Block

EndPrgm

Mall för att skapa ett användardefinierat program. Måste användas med kommandot **Define**, **Define LibPub** eller **Define LibPriv**.

Block kan vara ett enstaka påstående, en serie av påståenden separerade med tecknet “:” eller en serie av påståenden på separata rader.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Beräkna GCD och visa mellanliggande resultat.

```
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()

Se Π(), på sidan 189.

Product (PI)

Se Π(), på sidan 189.

product()

Katalog >

product(*List*[, *Start*[, *end*]])⇒*uttryck*

Ger produkten av elementen i *List*. *Start* och *end* är valfria. De specificerar ett område med element.

$\text{product}(\{1,2,3,4\})$	24
$\text{product}(\{4,5,8,9\},2,3)$	40

product(*Matrix1*[, *Start*[, *end*]])⇒*matris*

Ger en radvektor som innehåller produkterna av elementen i kolumnerna i *Matrix1*. *Start* och *end* är valfria. De specificerar ett område med rader.

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

$\text{product}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right)$	$\begin{bmatrix}28&80&162\end{bmatrix}$
$\text{product}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},1,2\right)$	$\begin{bmatrix}4&10&18\end{bmatrix}$

propFrac()

Katalog >

propFrac(*Value1*[, *Var*])⇒*värde*

propFrac(*rational_number***)** ger *rational_number* som summan av ett heltal och ett bråk med samma tecken och med större nämnare än täljare.

propFrac(*rational_expression*,*Var***)** ger summan av egentliga bråk och ett polynom med avseende på *Var*. Graden hos *Var* i nämnaren överskrider graden hos *Var* i täljaren i varje egentligt bråk. Liknande potenser av *Var* samlas in. Termerna och deras faktorer sorteras med *Var* som huvudvariabel.

Om *Var* utelämnas utförs en utveckling av ett egentligt bråk med avseende på den mest betydande variabeln. Koefficienterna för polynomdelen görs sedan egentliga, först med avseende på deras mest betydande variabel och så vidare.

$\text{propFrac}\left(\frac{4}{3}\right)$	$1+\frac{1}{3}$
$\text{propFrac}\left(\frac{-4}{3}\right)$	$-1-\frac{1}{3}$

propFrac() Katalog >

Du kan använda funktionen **propFrac()** för att representera blandade bråk och demonstrera addition och subtraktion av blandade bråk.

$\text{propFrac}\left(\frac{11}{7}\right)$	$1+\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8+\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2-\frac{29}{44}$

Q

QR Katalog >

QR *Matrix*, *qMatName*, *rMatName*[, *Tol*]

Beräknar faktoriseringen Householder QR av en reell eller komplex matris. De resulterande Q- och R-matriserna lagras i specificerad *MatNames*. Q-matrisen är unitär. R-matrisen är övertriangulär.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

QR-faktoriseringen beräknas numeriskt med Householder-transformationer. Den symboliska lösningen beräknas med Gram-Schmidt. Kolumnerna i *qMatName* är de ortonormerade basvektorerna som täcker utrymmet som definieras av *matrix*.

Siffran för flytande komma (9.) i m1 medför att resultat beräknas med flyttalsaritmetik.

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}$
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	$\begin{bmatrix}8.12404&9.60114&11.0782\\0.&0.904534&1.80907\\0.&0.&0.\end{bmatrix}$

QuadReg Katalog >

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

Utför den kvadratiska regressionsanalysen y = a ·x^2+b ·x+cpå listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln $stat.results$. (Se på sidan 148.)

Alla listor utom $Include$ måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

$Freq$ är en frivillig lista på frekvensvärden. Varje element i $Freq$ specificerar frekvensen för varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

$Category$ är en lista på numeriska eller strängkategorikoder för motsvarande X- och Y-data.

$Include$ är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element”, på sidan 200.

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·x^2+b ·x+c
stat.a, stat.b, stat.c	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade $X\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.YReg	Lista på datapunkter i den modifierade $Y\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.FreqReg	Lista på frekvenser som motsvarar $stat.XReg$ och $stat.YReg$

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

Utför en fjärdegrads regressionsanalys y = a ·x^4+b ·x^3+c · x^2+d ·x+epå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: a ·x^4+b ·x^3+c · x^2+d ·x+e
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

R▶Pθ() Katalog >

R▶Pθ (*xValue*, *yValue*) ⇒ *värde*
R▶Pθ (*xList*, *yList*) ⇒ *lista*
R▶Pθ (*xMatrix*, *yMatrix*) ⇒ *matris*

Ger den ekvivalenta θ-koordinaten för (*x*,*y*) värden.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer för respektive inställning av vinkelenhet.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`R@Ptheta(...)`**.

Med vinkelmått Grader:

R▸Pθ(2,2)	45.

Med vinkelenhet Nygrader:

R▸Pθ(2,2)	50.

Med vinkelenhet Radianer:

R▸Pθ(3,2)	0.588003
R▸Pθ([3 -4 2],[0 π/4 1.5])	[0. 2.94771 0.643501]

R▶Pr() Katalog >

R▶Pr (*xValue*, *yValue*) ⇒ *värde*
R▶Pr (*xList*, *yList*) ⇒ *lista*
R▶Pr (*xMatrix*, *yMatrix*) ⇒ *matris*

Ger den ekvivalenta r-koordinaten för argumentparen (*x*,*y*).

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`R@Pr(...)`**.

Med vinkelenhet Radianer:

R▸Pr(3,2)	3.60555
R▸Pr([3 -4 2],[0 π/4 1.5])	[3 4.07638 5/2]

▶Rad Katalog >

Value1▶*Rad* ⇒ *värde*

Konverterar argumentet till en vinkel i radianer.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Rad`**.

Med vinkelmått Grader:

(1.5)▸Rad	$(0.02618)^r$

Med vinkelenhet Nygrader:

(1.5)▸Rad	$(0.023562)^r$

rand( ) Katalog >

rand( ) ⇒ *uttryck*
rand(*#Trials***)** ⇒ *lista*

rand( ) ger ett slumpvärde mellan 0 och 1.

rand(*#Trials***)** ger en lista med *#Trials* slumpvärden mellan 0 och 1.

Bestämmer slumptalsfröet.

RandSeed 1147	*Done*
rand(2)	{0.158206,0.717917}

slumpBin( ) Katalog >

randBin(*n***,** *p***)** ⇒ *uttryck*
randBin(*n***,** *p***,** *#Trials***)** ⇒ *lista*

randBin(*n***,** *p***)** ger ett reellt slumptal från en specificerad binomialfördelning.

randBin(*n***,** *p***,** *#Trials***)** ger en lista med *#Trials* reella slumptal från en specificerad binomialfördelning.

randBin(80,0.5)	46.
randBin(80,0.5,3)	{43.,39.,41.}

slumpBin( ) Katalog >

randInt (*lowBound***,***upBound***)** ⇒ *uttryck*
randInt (*lowBound***,***upBound* **,***#Trials***)** ⇒ *lista*

randInt (*lowBound***,***upBound***)** ger ett slumptal med heltalsvärde inom det område som specificeras av heltalsgränserna *lowBound* och *upBound*.

randInt (*lowBound* **,***upBound***,***#Trials***)** ger en lista med *#Trials* slumptal med heltalsvärden inom det specificerade området.

randInt(3,10)	3.
randInt(3,10,4)	{9.,3.,4.,7.}

randMat( ) Katalog >

randMat(*numRows*, *numColumns***)** ⇒ *matris*

Ger en matris med heltal mellan -9 och 9 med specificerad dimension.

Båda argumenten måste förenklas till heltal.

RandSeed 1147	*Done*
randMat(3,3)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

Obs: Värdena i denna matris ändras varje gång du trycker på enter.

slumpNorm( ) Katalog >

randNorm(μ, σ**)** ⇒ *uttryck*
randNorm(μ, σ, *#Trials***)** ⇒ *lista*

randNorm(μ, σ**)** ger ett decimalt tal från den specificerade normalfördelningen. Det kan vara ett reellt tal vilket som helst, men det blir kraftigt koncentrerat i intervallet [μ−3•σ, μ+3•σ].

randNorm(μ, σ, *#Trials***)** ger en lista med *#Trials* decimaltal från den specificerade normalfördelningen.

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	−3.54356

randPoly() Katalog >

randPoly(*Var*, *Order***)** ⇒ *uttryck*

Ger ett polynom i *Var* i specificerad *Order*. Koefficienterna är slumpmässiga heltal i intervallet −9 till 9. Den första koefficienten kommer inte att vara noll.

Order måste vara 0–99.

RandSeed 1147	*Done*
randPoly(x,5)	$-2\cdot x^5+3\cdot x^4-6\cdot x^3+4\cdot x-6$

randSamp( ) Katalog >

randSamp(*List*,*#Trials*[,*noRepl*])**⇒ *lista*

Ger en lista på ett slumpmässigt urval av *#Trials* försök från *List* med ett alternativ för återläggning (*noRepl*=0) eller ingen återläggning(*noRepl*=1). Förinställningen är med urvalsutbyte.

Define *list3*={1,2,3,4,5}	*Done*
Define *list4*=randSamp(*list3*,6)	*Done*
list4	{1.,3.,3.,1.,3.,1.}

RandSeed Katalog >

RandSeed *Tal*

Om *Number* = 0 ställs fröna in på fabriksinställningarna för slumptalsgeneratorn. Om *Number* ≠0, används det för att generera två frön, vilka lagras i systemvariablerna seed1 och seed2.

RandSeed 1147	*Done*
rand()	0.158206

real( ) Katalog >

real(*Value1***)** ⇒ *värde*

Ger argumentets reella del.

real(*List1***)** ⇒ *lista*

Ger de reella delarna av alla element.

real(*Matrix1***)** ⇒ *matris*

Ger de reella delarna av alla element.

$\text{real}(2+3\cdot i)$	2
$\text{real}(\{1+3\cdot i,3,i\})$	$\{1,3,0\}$
$\text{real}\left(\begin{bmatrix}1+3\cdot i & 3\\ 2 & i\end{bmatrix}\right)$	$\begin{bmatrix}1 & 3\\ 2 & 0\end{bmatrix}$

▶Rect Katalog >

Vector **▶Rect**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@Rect`.

Visar *Vector* i rektangulär form [x, y, z]. Vektorn måste ha dimensionen 2 eller 3 och kan vara en rad eller en kolumn.

Obs: **▶Rect** är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad, och den uppdaterar inte *ans*.

Obs: Se även **▶Polar**, på sidan 113.

$$\left(\begin{bmatrix}3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6}\end{bmatrix}\right)\blacktriangleright\text{Rect}$$

$$\begin{bmatrix}1.06066 & 1.06066 & 2.59808\end{bmatrix}$$

complexValue **▶Rect**

Visar *complexValue* i rektangulär form a+bi. *complexValue* kan ha valfri komplex form. En inmatning av $re^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

Obs: Du måste använda parenteserna för en ($r\angle\theta$) polär inmatning.

Med vinkelenhet Radianer:

$\left(4\cdot e^{\frac{\pi}{3}}\right)\blacktriangleright\text{Rect}$	11.3986
$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Rect}$	$2.+3.4641\cdot i$

Med vinkelenhet Nygrader:

$$\{(1\angle 100)\}\blacktriangleright\text{Rect} \qquad i$$

Med vinkelmått Grader:

$$\{(4\angle 60)\}\blacktriangleright\text{Rect} \qquad 2.+3.4641\cdot i$$

Obs: För att skriva in tecknet ∠, välj det från symbollistan i Katalog.

ref( )

ref(*Matrix1*[, *Tol*]**)** ⇒ *matris*

Ger radtrappstegsformen av *Matrix1*.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl·enter eller ställer in **Auto or Approximate** på Approximate, utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E−14 •max(dim(*Matrix1*)) •rowNorm (*Matrix1*)

Undvik odefinierade element i *Matrix1*. De kan leda till oväntade resultat.

Om till exempel *a* är odefinierat i följande uttryck visas ett varningsmeddelande och resultatet ges som:

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{ref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

Varningen visas på grund av att det generaliserade elementet 1/a inte skulle vara giltigt för a=0.

Du kan undvika detta genom att i förväg lagra ett värde i a eller genom att använda (“|”)-operatorn begränsning för att ersätta ett värde såsom visas i följande exempel.

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) | a=0 \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Obs: Se även **rref( )**, på sidan 133.

RefreshProbeVars

RefreshProbeVars

Låter dig visa sensordata från alla anslutna sensorer via TI-grundprogrammet.

StatusVar-värde	Status
$statusVar$ =0	Normal (fortsätt med programmet) Vernier DataQuest™-applikationen är i datainsamlingsläge.
$statusVar$ =1	**Obs:** Vernier DataQuest™-applikationen måste vara i läge "meter" för att detta kommando ska fungera.
$statusVar$ =2	Vernier DataQuest™-applikationen har inte startats.
$statusVar$ =3	Vernier DataQuest™-applikationen har startats men inga givare har anslutits.

Exempel

```
Define temp()=
Prgm
© Kontrollera om systemet är klart
RefreshProbeVars status
If status=0 Then
Disp "klar"
För n,1,50
RefreshProbeVars status
temperatur:=mätare.temperatur
Disp "Temperatur: ",temperatur
If temperature>30 Then
Disp "För varm"
EndIf
© Vänta 1 sekund mellan mätningarna
Wait 1
```

```
EndFor

Else

Disp "Ej klar. Försök igen
senare"

EndIf

EndPrgm
```

Obs: Detta kan även användas med TI-Innovator™ hubb.

remain( )

remain(*Value1***,** *Value2***)** ⇒ *värde*
remain(*List1***,** *List2***)** ⇒ *lista*
remain(*Matrix1***,** *Matrix2***)** ⇒ *matris*

Ger resten på det första argumentet med avseende på det andra argumentet definierat av identiteterna:

remain(x,0) x
remain(x,y) x−y•iPart(x/y)

Som en följd av detta, observera att **remain(**−x,y**)** − **remain(**x,y**)**. Resultatet är antingen noll eller har samma tecken som det första argumentet.

Obs: Se även **mod( )**, på sidan 96.

remain(7,0)	7
remain(7,3)	1
remain(⁻7,3)	⁻1
remain(7,⁻3)	1
remain(⁻7,⁻3)	⁻1
remain({12,⁻14,16},{9,7,⁻5})	{3,0,1}

$$\text{remain}\left(\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Request

Begär *promptString*, *var*[, *DispFlag* [, *statusVar*]]

Request *promptString*, *func*(*arg1*, *...argn*) [, *DispFlag* [, *statusVar*]]

Programmeringskommando: Pausar programmet och visar en dialogruta med meddelandet *promptString* och en svarsruta där användaren ska skriva in svaret.

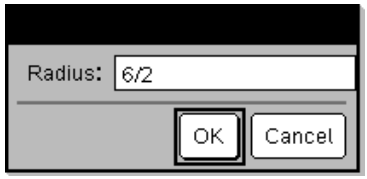
Definiera ett program:

```
Definiera begär_demo()=Prgm
    Begär ”Radie: ”,r
    Disp ”Area = ”,pi*r²
EndPrgm
```

Kör programmet och skriv in ett svar:

```
request_demo( )
```

När användaren skriver in svaret och klickar på **OK** tilldelas variabeln *var*det värde som står i svarsrutan.



Om användaren klickar på **Cancel** (Avbryt) fortsätter programmet utan att acceptera någon inmatning. Programmet använder det tidigare värdet på *var* om *var* redan var definierad.

Det valfria argumentet *DispFlag* kan vara ett valfritt uttryck.

- Om *DispFlag* utelämnas eller beräknas till **1**, visas promptmeddelandet och användarens svar i Räknarens historik.
- Om *DispFlag* beräknas till **0** visas inte prompten och svaret i historiken.

Det valfria argumentet *statusVar* ger programmet ett sätt att bestämma hur användaren lämnade dialogrutan. Observera att *statusVar* är beroende av argumentet *DispFlag*.

- Om användaren klickade på **OK** eller tryckte på **Enter** eller **Ctrl+Enter** ges variabeln *statusVar* värdet **1**.
- Annars får variabeln *statusVar* värdet **0**.

Med argumentet *funk*( ) kan programmet lagra användarens svar som en funktionsdefinition. Denna syntax fungerar som om användaren exekverade kommandot:

Definiera *funk*(*arg1*, ...*argn*) = *användarens svar*

Programmet kan sedan använda den definierade funktionen *func*(). Strängen *promptString* bör vägleda användaren till att skriva in ett lämpligt *användarsvar* som fullbordar funktionsdefinitionen.

Obs: Du kan använda Request -kommandot med ett användardefinierat program, men inte inom en funktion.

Resultat efter tryckning på **OK**:

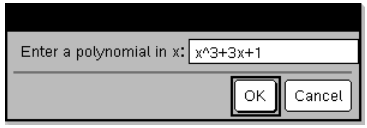
```
Radie: 6/2
Area= 28,2743
```

Definiera ett program:

```
Definiera polynom()=Prgm
    Request "Skriv in ett polynom i
x:",p(x)
    Disp "Reella rötter
är:",polyRoots(p(x),x)
EndPrgm
```

Kör programmet och skriv in ett svar:

```
polynom()
```



Resultat efter inmatning av x^3+3x+1 och tryckning på **OK**:

```
Reella rötter är: {-0,322185}
```

Request

Katalog >

För att stoppa ett program som innehåller ett **Request**-kommando inne i en oändlig slinga:

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Se även **RequestStr**, på sidan 128.

RequestStr

Katalog >

RequestStr *promptString*, *var*[, *DispFlag*]

Programmeringskommando: Fungerar på precis samma sätt som den första syntaxen i kommandot **Request**, förutom att användarens svar alltid tolkas som en sträng. Kommandot **Request** tolkar svaret som ett uttryck såvida inte användaren omger det med citationstecken ("").

Obs: Kommandot **RequestStr** kan användas inom ett användardefinierat program, men inte inom en funktion.

Att stoppa ett program som innehåller ett **RequestStr**-kommando i en oändlig loop:

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Se även **Request**, på sidan 126.

Definiera ett program:

```
Definiera requestStr_demo()=Prgm
    BegärStr "Ditt namn:",namn,0
    Disp “Svaret har “,dim(namn),”
tecken.”
EndPrgm
```

Kör programmet och skriv in ett svar:

```
begärStr_demo()
```

Resultat efter tryckning på **OK** (observera att argumentet *DispFlag* för **0** förbiser prompten och svarar från historiken):

```
begärStr_demo()
                    Svaret har 5 tecken.
```

Return

Katalog >

Return [*Expr*]

Ger *Expr* som resultatet av funktionen. Använd inom ett **Func**...**EndFunc**-block.

Obs: Använd **Return** utan ett argument inom ett **Prgm**...**EndPrgm**-block för att gå ur ett program.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
Define factorial(nn)=
Func
Local answer,counter
1→answer
For counter,1,nn
answer·counter→answer
EndFor
Return answer
EndFunc
```

```
factorial(3)                    6
```

right()

Katalog >

right(*List1*[, *Num*]**)** ⇒ *lista*

Ger *Num*-elementen längst till höger i *List1*.

Om du utelämnar *Num* erhålls alla i *List1*.

```
right({1,3,-2,4},3)             {3,-2,4}
```

right(*sourceString*[, *Num*]**)** ⇒ *sträng*

Ger *Num*-tecknen längst till höger i teckensträngen *sourceString*.

Om du utelämnar *Num* erhålls alla i *sourceString*.

```
right("Hello",2)                "lo"
```

right(*Comparison***)** ⇒ *uttryck*

Ger den högra sidan av en ekvation eller olikhet.

rk23 ()

Katalog >

rk23(*Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}, *depVar0*, *VarStep* [, *diftol*]**)** ⇒ *matris*

rk23(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep*[, *diftol*]**)** ⇒ *matrix*

rk23(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep*[, *diftol*]**)** ⇒ *matrix*

Differentialekvation:

y'=0,001*y*(100-y) och y(0)=10

```
rk23(0.001·y·(100−y),t,y,{0,100},10,1)
[0.      1.        2.        3.      4
 10.  10.9367  11.9493  13.042  14.2
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Använder Runge-Kuttas metod för att lösa systemet

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

med $depVar(Var0)=depVar0$ i intervallet $[Var0, VarMax]$. Ger en matris vars första rad definierar resultatvärdena för Var, definierade av $VarStep$. Den andra raden definierar värdet på den första lösningskomponenten vid motsvarande värden på Var, och så vidare.

$Expr$ är det högra ledet som definierar den ordinära differentialekvationen (ODE).

$SystemOfExpr$ är ett system av högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i $ListOfDepVars$).

$ListOfExpr$ är en lista på högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i $ListOfDepVars$).

Var är den oberoende variabeln.

$ListOfDepVars$ är en lista på oberoende variabler.

$\{Var0, VarMax\}$ är en lista med två element som instruerar funktionen att integrera från $Var0$ till $VarMax$.

$ListOfDepVars0$ är en lista på startvärden för oberoende variabler.

Om $VarStep$ utvärderas till ett tal skilt från noll ges sign($VarStep$) = sign($VarMax$-$Var0$) och lösningar vid $Var0$+i*$VarStep$ för alla i=0,1,2,… sådana att $Var0$+i*$VarStep$ är i intervallet $[var0, VarMax]$ (kanske inte ger ett lösningsvärde vid $VarMax$).

Om $VarStep$ utvärderas till noll ges lösningar vid "Runge-Kutta"-värdena för Var.

$diftol$ är feltoleransen (förinställs på 0,001).

Samma ekvation med $diftol$ inställd på 1.E−6

$$\text{rk23}\left(0.001\cdot y\cdot(100-y), t, y, \{0,100\}, 10, 1, 1.\text{E}^{-6}\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix}$$

Ekvationssystem:

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

med $y1(0)=2$ och $y2(0)=5$

$$\text{rk23}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases}, t, \{y1, y2\}, \{0,5\}, \{2,5\}, 1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

Rot() Katalog >

root(*Value***)** ⇒ *root*
root(*Value1***, *Value2***)** ⇒ *root*

root(*Value*) ger kvadratroten ur *Value*.

root(*Value1***, *Value2***)** ger *Value2*-roten ur *Value1*. *Value1* kan vara en reell eller komplex konstant i flyttalsform, ett heltal eller komplex rationell konstant.

Obs: Se även **Nth root template**, på sidan 1.

$\sqrt[3]{8}$	2
$\sqrt[3]{3}$	1.44225

rotate() Katalog >

rotate(*Integer1***[,***#ofRotations***])** ⇒ *heltal*

Roterar bitarna i ett binärt heltal. *Integer1* kan anges i valfri talbas. Det omvandlas automatiskt till 64 positioners binär form. Om storleken på *Integer1* är alltför stor för denna form, för en symmetrisk modulooperation talet inom området. För mer information, se ▶**Base2**, på sidan 16.

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är −1 (rotera en position åt höger).

Vid exempelvis rotation åt höger:

Varje bit roteras åt höger.

0b00000000000000111101011000011010 1

Biten längst till höger roteras till positionen längst till vänster.

ger:

0b10000000000000011110101100001101 0

Resultatet visas enligt det inställda basläget.

I binärt basläge:

rotate(0b11111111111111111111111111111111)	0b10000000000000000000000000000001▸
rotate(256,1)	0b1000000000

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

I hexadecimalt basläge:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

Viktigt: För att skriva in ett binärt eller hexadecimalt tal, använd alltid prefixet 0b eller 0h (noll, inte bokstaven O).

rotate(*List1***[,***#ofRotations***])** ⇒ *lista*

Ger en kopia av *List1* roterad åt höger eller vänster av *#ofRotations*-elementen. Ändrar inte *List1*.

I decimalt basläge:

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() Katalog >

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är −1 (rotera ett element åt höger).

rotate(*String1*[,*#ofRotations*]**)** ⇒ *sträng*

Ger en kopia av *String1* roterad åt höger eller vänster av *#ofRotations*-tecknen. Ändrar inte *String1*.

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är −1 (rotera ett tecken åt höger).

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

round() Katalog >

round(*Value1*[, *digits*]**)** ⇒ *värde*

Ger argumentet avrundat till det specificerade antalet siffror efter decimalpunkten.

digits måste vara ett heltal i intervallet 0–12. Om *digits* inte ingår, ges argumentet avrundat till 12 signifikanta siffror.

Obs: Läget Display digits (Visa siffror) kan påverka hur detta visas.

round(1.234567,3)	1.235

round(*List1*[, *digits*]**)** ⇒ *lista*

Ger en lista på elementen avrundade till det specificerade antalet siffror.

round(*Matrix1*[, *digits*]**)** ⇒ *matris*

Ger en matris över elementen avrundade till det specificerade antalet siffror.

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() Katalog >

rowAdd(*Matrix1*, *rIndex1*, *rIndex2*) ⇒ *matris*

Ger en kopia av *Matrix1* med rad *rIndex2* ersatt av summan av raderna *rIndex1* och *rIndex2*.

rowAdd([3 4; -3 -2],1,2)	[3 4; 0 2]

rowDim() Katalog >

rowDim(*Matrix***)** ⇒ *uttryck*

Ger antalet rader i *Matrix*.

Obs: Se även **colDim()**, på sidan 23.

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

$$\text{rowDim}(m1) \qquad 3$$

rowNorm() Katalog >

rowNorm(*Matrix***)** ⇒ *uttryck*

Ger maximum av summorna av absolutbeloppen på elementen i raderna i *Matrix*.

Obs: Alla matriselement måste förenklas till tal. Se även **colNorm()**, på sidan 23.

$$\text{rowNorm}\left(\begin{bmatrix}-5&6&-7\\3&4&9\\9&-9&-7\end{bmatrix}\right) \qquad 25$$

rowSwap() Katalog >

rowSwap(*Matrix1***, *rIndex1*, *rIndex2***)** ⇒ *matris*

Ger *Matrix1* med raderna *rIndex1* och *rIndex2* växlade.

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

$$\text{rowSwap}(mat,1,3) \qquad \begin{bmatrix}5&6\\3&4\\1&2\end{bmatrix}$$

rref() Katalog >

rref(*Matrix1*[**,** *Tol*]**)** ⇒ *matris*

Ger den reducerade radtrappstegsformen av *Matrix1*.

$$\text{rref}\left(\begin{bmatrix}-2&-2&0&-6\\1&-1&9&-9\\-5&2&4&-4\end{bmatrix}\right) \qquad \begin{bmatrix}1&0&0&\frac{66}{71}\\0&1&0&\frac{147}{71}\\0&0&1&\frac{-62}{71}\end{bmatrix}$$

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl eller ställer in **Auto or Approximate** på Approximate, utförs beräkningarna med

flyttalsaritmetik.

- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E−14 •max(dim(*Matrix1*)) •rowNorm (*Matrix1*)

Obs: Se även **ref( )**, på sidan 124.

S

sec() trig tangent

sec(*Value1***)** ⇒ *värde*

sec(*List1***)** ⇒ *lista*

Ger sekansfunktionen för *Value1* eller en lista på sekansfunktionerna för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

sec(45)	1.41421
sec({1,2.3,4})	{1.00015,1.00081,1.00244}

sec⁻¹() trig tangent

sec⁻¹(*Value1***)** ⇒ *värde*

sec⁻¹(*List1***)** ⇒ *lista*

Ger den vinkel vars sekansfunktion är *Value1* eller en lista på de inversa sekansfunktionerna för alla element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

I vinkelläget Grader:

$\sec^{-1}(1)$	0.

I vinkelläget Nygrader:

$\sec^{-1}(\sqrt{2})$	50.

I vinkelläget Radianer:

$\sec^{-1}(\{1,2,5\})$	{0,1.0472,1.36944}

$\sec^{-1}()$ — trig tangent

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arcsec(...)`**.

sech() — Katalog >

sech(*Value1***)** ⇒ *värde*

sech(*List1***)** ⇒ *lista*

Ger den hyperboliska sekansfunktionen för *Value1* eller en lista på de hyperboliska sekansfunktionerna för elementen i *List1*.

sech(3)	0.099328
sech({1,2.3,4})	{0.648054,0.198522,0.036619}

$\text{sech}^{-1}()$ — Katalog >

sech⁻¹(*Value1***)** ⇒ *värde*

sech⁻¹ (*List1***)** ⇒ *lista*

Ger den inversa hyperboliska sekansfunktionen för *Value1* eller en lista på den inversa hyperboliska sekansfunktionen för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arcsech(...)`**.

I vinkelläget Radianer och i Rektangulärt komplext läge:

$\text{sech}^{-1}(1)$	0
$\text{sech}^{-1}(\{1,-2,2.1\})$	$\{0,2.0944\cdot i,8.E^{-15}+1.07448\cdot i\}$

Send — Hubb-meny

Send*UttrEllerSträng1* [, *UttrEllerSträng2*] ...

Programmeringskommando: Skickar ett eller flera [[[Undefined variable nspire_all_var.HubFullName]]] kommandon till en ansluten hubb.

exprOrString måste vara ett giltigt [[[Undefined variable nspire_all_var.HubFullName]]] Kommando. *exprOrString* innehåller vanligen ett **"SET ..."** -kommando för att styra en enhet eller ett **"READ ..."** -kommando för att begära data.

Exempel: Slå på den blå komponenten i den inbyggda RGB-lysdioden i 0,5 sekunder.

```
Send "SET COLOR.BLUE ON TIME .5"
                                Done
```

Exempel: Begär nuvarande värde från hubbens inbyggda ljusnivåsensor. Ett **Get**-kommando hämtar värdet och tilldelar det till variabeln *lightval*.

Send — Hubb-meny

Argumenten skickas till hubben i turordning.

Obs: Du kan använda kommandot **Send** i ett användardefinierat program, men inte i en funktion.

Obs: Se även **Get** (på sidan 60), **GetStr** (på sidan 67) och **eval()** (på sidan 48).

Send "READ BRIGHTNESS"	*Done*
Get *lightval*	*Done*
lightval	0.347922

Exempel: Skicka en beräknad frekvens till hubbens inbyggda högtalare. Använd den speciella variabeln *iostr.SendAns* för att visa hubb-kommandot med det beräknade uttrycket.

n:=50	50
m:=4	4
Send "SET SOUND eval(m· n)"	*Done*
iostr.SendAns	"SET SOUND 200"

seq() — Katalog >

seq(*Expr***,** *Var***,** *Low***,** *High***[,** *Step***])⇒***lista*

Ökar *Var* från *Low* till *High* i steg om *Step*, utvärderar *Expr* och ger resultaten som en lista. De ursprungliga innehållen i *Var* är fortfarande där när **seq()** har beräknats.

Det förinställda värdet på *Step* = 1.

$\text{seq}(n^2,n,1,6)$	$\{1,4,9,16,25,36\}$
$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$	$\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	$\frac{1968329}{1270080}$

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	1.54977

seqGen() — Katalog >

seqGen(*Expr***,** *Var***,** *depVar***,** {*Var0***,** *VarMax*}**[,** *ListOfInitTerms* **[,** *VarStep* **[,** *CeilingValue*]]]**) ⇒***lista*

Genererar de första 5 termerna i talföljden $u(n) = u(n\text{-}1)^2/2$, med $u(1)$=**2** och *VarStep*=**1**.

seqGen() Katalog >

Genererar en lista på termer för talföljden *depVar*(*Var*)=*Expr* enligt följande: Ökar den oberoende variabeln *Var* från *Var0* till *VarMax* i steg om *VarStep*, utvärderar *depVar*(*Var*) för motsvarande värden på *Var* med formeln *Expr* och *ListOfInitTerms* och ger resultaten som en lista.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n},n,u,\{1,5\},\{2\}\right)$$

$$\left\{2,2,\frac{4}{3},\frac{4}{9},\frac{16}{405}\right\}$$

seqGen(*ListOrSystemOfExpr***,** *Var***,** *ListOfDepVars***,** {*Var0***,** *VarMax*} [**,** *MatrixOfInitTerms* [**,** *VarStep* [**,** *CeilingValue*]]]**)** ⇒*matris*

Genererar en matris med termer för ett system (eller en lista) av talföljder *ListOfDepVars*(*Var*) =*ListOrSystemOfExpr* enligt följande: Ökar den oberoende variabeln *Var* från *Var0* till *VarMax* i steg om *VarStep*, utvärderar *ListOfDepVars*(*Var*) för motsvarande värden på *Var* med formeln *ListOrSystemOfExpr* och *MatrixOfInitTerms* och ger resultaten som en matris.

De ursprungliga innehållen i *Var* är oförändrade när **seqGen()** har beräknats.

Det förinställda värdet på *VarStep* = **1**.

Exempel i vilket Var0=2:

$$\text{seqGen}\left(\frac{u(n-1)+1}{n},n,u,\{2,5\},\{3\}\right)$$

$$\left\{3,\frac{4}{3},\frac{7}{12},\frac{19}{60}\right\}$$

System med två talföljder:

$$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2(n-1)}{2}+u1(n-1)\right\},n,\{u1,u2\},\{1,5\},\left[\frac{_}{2}\right]\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Obs: Tomrummet (_) i den initiala termmatrisen ovan används för att indikera att den initiala termen för u1(n) beräknas med den explicita talföljdsformeln u1(n)=1/n.

seqn() Katalog >

seqn(*Expr***(***u***,** *n* [**,** *ListOfInitTerms*[**,** *nMax* [**,** *CeilingValue*]]]**)**⇒*lista*

Genererar en lista på termer för en talföljd, *u*(*n*)=*Expr*(*u*, *n*), enligt följande: Ökar *n* från 1 till *nMax* i steg om 1, utvärderar *u*(*n*) för motsvarande värden på *n* med formeln *Expr*(*u*, *n*) och *ListOfInitTerms* och ger resultaten som en lista.

seqn(*Expr***(***n* [**,** *nMax* [**,** *CeilingValue*]]**)** ⇒*lista*

Genererar de första 6 termerna i talföljden *u*(*n*) = *u*(*n*-1)/2, med *u*(1)=**2**.

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right)$$

$$\left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

seqn() Katalog >

Genererar en lista på termer för en icke-rekursiv talföljd, $u(n)=Expr(n)$, enligt följande: Ökar n från 1 till $nMax$ i steg om 1, utvärderar $u(n)$ för motsvarande värden på n med formeln $Expr(n)$ och ger resultaten som en lista.

Om $nMax$ saknas ställs $nMax$ in på 2500

Om $nMax$=0 ställs $nMax$ in på 2500

Obs: seqn() anropar **seqGen( )** med $n0$=**1** och $nstep$=**1**

setMode() Katalog >

setMode(*modeNameInteger***,** *settingInteger***)** ⇒*heltal*

setMode(*list***)** ⇒*lista på heltal*

Endast giltigt inom en funktion eller ett program.

setMode(*modeNameInteger***,** *settingInteger***)** ställer temporärt in läget *modeNameInteger* på den nya inställningen *settingInteger* och ger ett heltal som motsvarar den ursprungliga inställningen på det läget. Ändringen är begränsad till den tid det tar att exekvera programmet/funktionen.

modeNameInteger specificerar vilket läge du vill ställa in. Det måste vara något av de lägesheltal som anges i nedanstående tabell.

settingInteger specificerar den nya inställningen för läget. Det måste vara något av de inställningsheltal som anges i nedanstående tabell för det läge som du ställer in.

setMode(*list***)** låter dig ändra flera inställningar. *list* innehåller par av lägesheltal och inställningsheltal.. **setMode (***list***)** ger en liknande lista vars heltalspar representerar de ursprungliga lägena och inställningarna.

Visa ungefärligt värde på π med förinställningen för Display Digits (Visa siffror) och visa sedan π med inställningen Fix2. Kontrollera sedan att förinställningen har återställts efter exekveringen av programmet.

```
Define prog1()=Prgm                    Done
               Disp π
               setMode(1,16)
               Disp π
               EndPrgm
prog1()
                                  3.14159
                                  3.14
                                     Done
```

Om du har sparat alla lägesinställningar med **getMode(0)** → *var* kan du använda **setMode(***var***)** för att återställa dessa inställningar tills funktionen eller programmet avslutas. Se **getMode()**, på sidan 66.

Obs: De aktuella lägesinställningarna förs vidare till anropade delrutiner. Om någon delrutin ändrar en lägesinställning förloras lägesändringen när kontrollen återgår till den anropande delrutinen.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Lägets namn	Läges-heltal	Heltal för inställningar
Display Digits (Visa siffror)	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 (Vinkel)	2	**1**=Radian, **2**=Degree, **3**=Gradian
Exponential Format (Exponentiellt format)	3	**1**=Normal, **2**=Scientific, **3**=Engineering
Real or Complex (Reellt eller Komplext)	4	**1**=Real, **2**=Rectangular, **3**=Polar
Auto or Approx. (Auto eller Ungefärlig)	5	**1**=Auto, **2**=Approximate
Vector Format (Vektorformat)	6	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base (Bas)	7	**1**=Decimal, **2**=Hex, **3**=Binary

shift(*Integer1*[**,***#ofShifts*]**)⇒***heltal*

Skiftar bitarna i ett binärt heltal. Du kan skriva in *Integer1* i valfri talbas. Det omvandlas automatiskt till 64-bitars binär form. Om storleken på *Integer1* är alltför stor för denna form för en symmetrisk modulooperation talet inom området. För mer information, se ▶**Base2**, på sidan 16.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är ⁻1 (skifta en bit åt höger).

Vid en skiftning åt höger "droppas" biten längst till höger och 1 infogas som denna bit. Vid skiftning åt vänster "droppas" biten längst till vänster och 0 infogas som denna bit.

Vid exempelvis skiftning åt höger:

Varje bit skiftas åt höger.

0b00000000000000111101011000011010

Infogar 0 om biten längst till vänster är 0 eller 1 om denna bit är 1.

ger:

0b00000000000000011110101100001101 0

Resultatet visas enligt det inställda basläget. Inledande nollor visas inte.

I binärt basläge:

shift(0b1111010110000110101)	0b111101011000011010
shift(256,1)	0b1000000000

I hexadecimalt basläge:

shift(0h78E)	0h3C7
shift(0h78E,⁻2)	0h1E3
shift(0h78E,2)	0h1E38

Viktigt: För att skriva in ett binärt eller hexadecimalt tal, använd alltid prefixet 0b eller 0h (noll, inte bokstaven O).

shift(*List1* [**,***#ofShifts*]**)⇒***lista*

Ger en kopia av *List1* skiftad åt höger eller vänster av *#ofShifts*-elementen. Ändrar inte *List1*.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är ⁻1 (skifta ett element åt höger).

Element som infogas i början eller slutet av *list* av skiftningen tilldelas symbolen "undef".

I decimalt basläge:

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() Katalog >

shift(*String1* [,*#ofShifts*]**)⇒***sträng*

Ger en kopia av *String1* skiftad åt höger eller vänster av *#ofShifts*-tecknen. Ändrar inte *String1*.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är ⁻1 (skifta ett tecken åt höger).

Tecken som infogas i början eller slutet av *string* av skiftningen får ett mellanslag.

shift("abcd")	" abc"
shift("abcd",⁻2)	" ab"
shift("abcd",1)	"bcd "

sign() Katalog >

sign(*Value1***)⇒***värde*

sign(*List1***)⇒***lista*

sign(*Matrix1***)⇒***matris*

Ger, för ett reellt eller komplext *Value1*, *Value1* / **abs(***Value1***)** när *Value1* ≠ 0.

Ger 1 om *Value1* är positivt.

Ger ⁻1 om *Value1* är negativt.

sign(0) ger ±1 om det komplexa formatläget är Real, annars ger det sig självt.

sign(0) representerar enhetscirkeln i det komplexa området.

Ger, för en lista eller matris, tecknen för alla element.

sign(⁻3.2)	⁻1
sign({2,3,4,⁻5})	{1,1,1,⁻1}

Om det komplexa formatläget är Real:

sign([⁻3 0 3])	[⁻1 undef 1]

simult() Katalog >

simult(*coeffMatrix***,** *constVector*[**,** *Tol*]**)** ⇒*matris*

Ger en kolumnvektor som innehåller lösningarna för ett system av linjära ekvationer.

Obs: Se även **linSolve()**, på sidan 83.

Lös för x och y:

x + 2y = 1

3x + 4y = ⁻1

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

Lösningen är x=⁻3 och y=2.

coeffMatrix måste vara en kvadratmatis som innehåller koefficienterna för ekvationerna.

constVector måste ha samma antal rader (samma dimension) som *coeffMatrix* och innehålla konstanterna.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du ställer in läget **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*coeffMatrix*)) ·rowNorm(*coeffMatrix*)

Lös:

ax + by = 1

cx + dy = 2

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\rightarrow matx1 \qquad \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$$

$$\text{simult}\left(matx1,\begin{bmatrix}1\\2\end{bmatrix}\right) \qquad \begin{bmatrix}0\\\frac{1}{2}\end{bmatrix}$$

simult(*coeffMatrix***,** *constMatrix***[,** *Tol***])** ⇒*matris*

Löser multipla system av linjära ekvationer där varje system har samma koefficienter för variablerna, men olika konstanter.

Varje kolumn i *constMatrix* måste innehålla konstanterna för ett ekvationssystem. Varje kolumn i den resulterande matrisen innehåller lösningen på motsvarande system.

Lös:

x + 2y = 1

3x + 4y = ⁻1

x + 2y = 2

3x + 4y = ⁻3

$$\text{simult}\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix},\begin{bmatrix}1 & 2\\-1 & -3\end{bmatrix}\right) \qquad \begin{bmatrix}-3 & -7\\2 & \frac{9}{2}\end{bmatrix}$$

För det första systemet är lösningen x=⁻3 och y=2. För det andra systemet är lösningen x=⁻7 och y=9/2.

sin() tangent

sin(*Value1***)**⇒*värde*

sin(*List1***)**⇒*lista*

sin(*Value1***)** Ger sinus för argumentet.

I vinkelläget Grader:

sin() tangent trig

sin(*List1***)** ger en lista på sinus för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i antingen grader, nygrader eller radianer beroende på det aktuella vinkelläget. Du kan använda °,G eller r för att tillfälligt överstyra vinkelläget.

$\sin\left(\left(\frac{\pi}{4}\right)r\right)$	0.707107
$\sin(45)$	0.707107
$\sin(\{0,60,90\})$	$\{0.,0.866025,1.\}$

I vinkelläget Nygrader:

$\sin(50)$	0.707107

I vinkelläget Radianer:

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

sin(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

$\sin^{-1}$() tangent trig

$\sin^{-1}$(*Value1***)⇒***värde*

$\sin^{-1}$(*List1***)⇒***lista*

$\sin^{-1}$(*Value1***)** ger den vinkel vars sinus är *Value1*.

$\sin^{-1}$(*List1***)** ger en lista på invers sinus för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arcsin(…)`**.

I vinkelläget Grader:

$\sin^{-1}(1)$	90.

I vinkelläget Nygrader:

$\sin^{-1}(1)$	100.

I vinkelläget Radianer:

$\sin^{-1}(\{0,0.2,0.5\})$	$\{0.,0.201358,0.523599\}$

$\sin^{-1}()$ tangent

$\sin^{-1}$(*squareMatrix1*)⇒*kvadratMatris*

Ger matrisen med invers sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

$$\sin^{-1}\left(\begin{bmatrix}1 & 5\\ 4 & 2\end{bmatrix}\right)$$
$$\begin{bmatrix}-0.174533-0.12198\cdot i & 1.74533-2.35591\cdot i\\ 1.39626-1.88473\cdot i & 0.174533-0.593162\cdot i\end{bmatrix}$$

sinh() Katalog >

sinh(*Numver1*)⇒*värde*

sinh(*List1*)⇒*lista*

sinh (*Value1*) ger argumentets hyperboliska sinus.

sinh (*List1*) ger en lista på hyperboliska sinus för varje element i *List1*.

$$\sinh(1.2) \qquad 1.50946$$
$$\sinh(\{0,1.2,3.\}) \qquad \{0,1.50946,10.0179\}$$

sinh(*squareMatrix1*)⇒*kvadratMatris*

Ger matrisen med hyperbolisk sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

$$\sinh\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$
$$\begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

$\sinh^{-1}()$ Katalog >

$\sinh^{-1}$(*Value1*)⇒*värde*

$\sinh^{-1}$(*List1*)⇒*lista*

$\sinh^{-1}$(*Value1*) ger argumentets inversa hyperboliska sinus.

$\sinh^{-1}$(*List1*) ger en lista på invers hyperbolisk sinus för varje element i *List1*.

$$\sinh^{-1}(0) \qquad 0$$
$$\sinh^{-1}(\{0,2.1,3\}) \qquad \{0,1.48748,1.81845\}$$

$\sinh^{-1}()$ Katalog >

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arcsinh(...)`.

$\sinh^{-1}$(*squareMatrix1***)⇒***kvadratMatris***

Ger matrisen med invers hyperbolisk sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet innehåller alltid tal med flytande komma.

I vinkelläget Radianer:

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

$$\begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$$

SinReg Katalog >

SinReg *X*, *Y* [, [*Iterations*],[*Period*] [, *Category*, *Include*]]

Utför en trigonometrisk regressionsanalys på listorna *X* och *Y*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Iterations är ett värde som specificerar det maximala antalet gånger (1 till och med 16) en lösning kommer att provas. Om denna utelämnas används 8. Normalt ger större värden bättre noggrannhet, men längre exekveringstider, och vice versa.

Period specificerar en uppskattad period. Om denna utelämnas bör skillnaden mellan värdena i *X* vara lika och i ordningsföljd. Om du specificerar *Period* kan skillnaderna mellan x-värden vara olika.

Category är en lista på numeriska eller strängkategorikoder för motsvarande *X*- och *Y*-data.

SinReg

Katalog >

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Resultatet av **SinReg** är alltid i radianer oavsett det inställda vinkelläget.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·sin(bx+c)+d
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

SortA

Katalog >

SortA *List1*[, *List2*] [, *List3*] ...

SortA *Vector1*[, *Vector2*] [, *Vector3*] ...

Sorterar elementen i det första argumentet i stigande ordning.

Om du inkluderar ytterligare argument sorteras elementen i varje argument så att deras nya positioner matchar de nya positionerna för elementen i det första argumentet.

Alla argument måste vara namn på listor eller vektorer. Alla argument måste ha samma dimensioner.

Tomma element inom det första argumentet flyttas till botten. För mer information om tomma element, se på sidan 200.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
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

Katalog >

SortD *List1*[, *List2*] [, *List3*] ...

SortD *Vector1*[,*Vector2*] [,*Vector3*] ...

Identisk med **SortA** med undantag för att **SortD** sorterar elementen i fallande ordning.

Tomma element inom det första argumentet flyttas till botten. För mer information om tomma element, se på sidan 200.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

▸Sphere

Katalog >

Vector **▸Sphere**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Sphere`**.

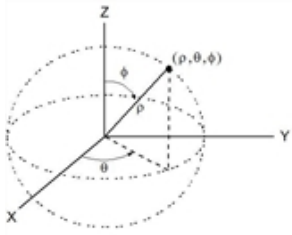
$$[1 \quad 2 \quad 3] \blacktriangleright \text{Sphere}$$
$$[3.74166 \quad \angle 1.10715 \quad \angle 0.640522]$$

Visar rad- eller kolumnvektorn i sfärisk form [ρ ∠θ ∠ϕ].

Vector måste ha dimensionen 3 och kan vara antingen en rad- eller en kolumnvektor.

$$\left(2 \quad \angle\frac{\pi}{4} \quad 3\right) \blacktriangleright \text{Sphere}$$
$$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$$

Obs: ▸Sphere är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad.



sqrt()

Katalog >

sqrt(*Value1***)⇒***värde*

sqrt(*List1***)⇒***lista*

Ger kvadratroten ur argumentet.

Ger, för en lista, kvadratrötterna ur alla element i *List1*.

$\sqrt{4}$	2
$\sqrt{\{9,2,4\}}$	$\{3,1.41421,2\}$

sqrt() Katalog >

Obs: Se även **Square root template**, på sidan 1.

stat.results Katalog >

stat.results

Visar resultaten av en statistisk beräkning.

Resultaten visas som en uppsättning av namn-värdepar. De specifika namnen som visas beror på den/det senast utvärderade statistiska funktionen/kommandot.

Du kan kopiera ett namn eller ett värde och klistra in det på andra platser.

Obs: Undvik att definiera variabler med samma namn som de variabler vilka används för statistisk analys. I vissa fall kan ett feltillstånd uppstå. Variabelnamn som används för statistisk analys listas i nedanstående tabell.

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.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.F̄	stat.n	stat.Σx	stat.x̄
stat.b9	stat.F̄Block	stat.p̂	stat.Σx²	stat.x̄1
stat.b10	stat.F̄col	stat.p̂1	stat.Σxy	stat.x̄2

stat.bList	stat.F̄Interact	stat.$\hat{p}$2	stat.Σy	stat.x̄Diff
stat.χ^2	stat.FreqReg	stat.$\hat{p}$Diff	stat.Σy²	stat.x̄List
stat.c	stat.F̄row	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat.ȳ
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat.ŷ
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat.ŷList
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

Obs: Varje gång applikationen Listor och kalkylblad beräknar statistiska resultat kopierar den “stat.”-gruppvariabler till en “stat#.”-grupp där # är ett tal som ökas automatiskt. Detta låter dig bibehålla tidigare resultat medan du utför flera beräkningar.

stat.values

Katalog >

stat.values

Visar en matris över värdena beräknade för den/det senast utvärderade statistiska funktionen/kommandot.

Till skillnad från **stat.results** utelämnar **stat.values** namnen som är associerade med värdena.

Du kan kopiera ett värde och klistra in det på andra platser.

Se exemplet **stat.results**.

stDevPop()

Katalog >

stDevPop(*List***[,** *freqList***])⇒***uttryck*

Ger populationens standardavvikelse för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

I vinkelläget Radianer och i Auto-läge:

stDevPop({1,2,5,⁻6,3,⁻2})	3.59398
stDevPop({1.3,2.5,⁻6.4},{3,2,5})	4.11107

stDevPop() Katalog >

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

stDevPop(*Matrix1***[,** *freqMatrix***])⇒***matris*

Ger en radvektor med populationens standardavvikelser för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

$$\text{stDevPop}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right)$$
$$\begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right)$$
$$\begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

stDevSamp() Katalog >

stDevSamp(*List***[,** *freqList***])⇒***uttryck*

Ger urvalets standardavvikelse för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

$$\text{stDevSamp}(\{1,2,5,-6,3,-2\}) \qquad 3.937$$

$$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.33345$$

stDevSamp(*Matrix1***[,** *freqMatrix***])⇒***matris*

Ger en radvektor med urvalets standardavvikelser för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

$$\text{stDevSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right)$$
$$\begin{bmatrix} 4. & 3.60555 & 2. \end{bmatrix}$$

$$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right)$$
$$\begin{bmatrix} 2.7005 & 5.44695 \end{bmatrix}$$

Stop

Katalog >

Stop

Programmeringskommando: Avslutar programmet.

Stop är ej tillåtet i funktioner.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

i:=0	0
Define $prog1()$=Prgm For i,1,10,1 If i=5 Stop EndFor EndPrgm	*Done*
$prog1()$	*Done*
i	5

Store

Se → (store), på sidan 198.

string()

Katalog >

string(*Expr***)⇒***sträng*

Förenklar *Expr* och ger resultatet som en teckensträng.

string(1.2345)	"1.2345"
string(1+2)	"3"

subMat()

Katalog >

subMat(*Matrix1***[, *startRow*] [, *startCol*] [, *endRow*] [, *endCol*]) ⇒***matris***

Ger specificerad undermatris av *Matrix1*.

Förvalsinställningar: *startRow*=1, *startCol*=1, *endRow*=sista rad, *endCol*=sista kolumn.

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$
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)

Se Σ(), på sidan 190.

sum() Katalog >

sum(*List***[, *Start*[, *End*]])⇒***uttryck***

Ger summan av elementen i *List*.

Start och *end* är valfria. De specificerar ett område med element.

Varje tomt argument ger ett tomt resultat. Tomma element i *List* ignoreras. För mer information om tomma element, se på sidan 200.

$\text{sum}(\{1,2,3,4,5\})$ 15

$\text{sum}(\{a,2\cdot a,3\cdot a\})$ "Error: Variable is not defined"

$\text{sum}(\text{seq}(n,n,1,10))$ 55

$\text{sum}(\{1,3,5,7,9\},3)$ 21

sum(*Matrix1***[, *Start*[, *End*]])⇒***matris***

Ger en radvektor som innehåller summorna av elementen i kolumnerna i *Matrix1*.

Start och *end* är valfria. De specificerar ett område med rader.

Varje tomt argument ger ett tomt resultat. Tomma element i *Matrix1* ignoreras. För mer information om tomma element, se på sidan 200.

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\right)$ $\begin{bmatrix}5&7&9\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right)$ $\begin{bmatrix}12&15&18\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$ $\begin{bmatrix}11&13&15\end{bmatrix}$

sumIf() Katalog >

sumIf(*List*,*Criteria*[, *SumList*])⇒***värde***

Ger den totala kumulerade summan av alla element i *List* som uppfyller specificerade *Criteria*. Du kan också specificera en alternativ lista, *sumList*, för att ge elementen som skall kumuleras.

List kan vara ett uttryck, en lista eller en matris. *SumList*, om specificerad, måste ha samma dimension(er) som *List*.

Criteria kan vara:

- Ett värde, ett uttryck eller en sträng. Som exempel ackumulerar **34** endast de element i *List* som förenklas till värdet 34.
- Ett booleskt uttryck som innehåller symbolen **?** fungerar som platshållare för varje element. Som exempel ackumulerar **?<10** endast de element i *List* som är lägre än 10.

$\text{sumIf}(\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5)$ 12.859874482

$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$ 70

sumIf() Katalog >

När ett *List*-element uppfyller *Criteria* adderas elementet till den ackumulerade summan. Om du inkluderar *sumList* adderas i stället motsvarande element från *sumList* till summan.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för *List* och *sumList*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 200.

Obs: Se även **countIf()**, på sidan 30.

sumSeq() Se Σ(), på sidan 190.

system() Katalog >

system(*Value1* [**,** *Value2* [**,** *Value3* [**,** ...]]]**)**

Ger ett ekvationssystem formaterat som en lista. Du kan också skapa ett system med en mall.

T

T(transponera) Katalog >

*Matrix1***T**⇒*matris*

Ger den komplexkonjugerade transponeringen av *Matrix1*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@t`.

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

tan() trig tangent

tan(*Value1***)**⇒*värde*

tan(*List1***)**⇒*lista*

tan(*Value1***)** ger tangensen för argumentet.

I vinkelläget Grader:

tan() trig tangent

tan(*List1***)** ger en lista på tangensen för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i antingen grader, nygrader eller radianer beroende på det aktuella vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

$\tan\left(\left(\frac{\pi}{4}\right)^r\right)$	1.
$\tan(45)$	1.
$\tan(\{0,60,90\})$	$\{0.,1.73205,\text{undef}\}$

I vinkelläget Nygrader:

$\tan\left(\left(\frac{\pi}{4}\right)^r\right)$	1.
$\tan(50)$	1.
$\tan(\{0,50,100\})$	$\{0.,1.,\text{undef}\}$

I vinkelläget Radianer:

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45°)$	1.
$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right)$	$\{0.,1.73205,0.,1.\}$

tan(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med tangensen för *squareMatrix1*. Detta är inte detsamma som att beräkna tangensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

$$\begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

$\tan^{-1}$() trig tangent

$\tan^{-1}$(*Value1***)⇒***värde*

$\tan^{-1}$(*List1***)⇒***lista*

$\tan^{-1}$(*Value1***)** ger den vinkel vars tangens är *Value1*.

$\tan^{-1}$(*List1***)** ger en lista på den inversa tangensen för varje element i *List1*.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

$\tan^{-1}()$ — trig tangent

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arctan(...)`.

I vinkelläget Radianer:

$\tan^{-1}(\{0,0.2,0.5\})$	$\{0,0.197396,0.463648\}$

**$\tan^{-1}$(*squareMatrix1*)⇒*kvadratMatris*

Ger matrisen med invers tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna invers tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

$$\begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

tanh() — Katalog >

tanh(*Value1***)⇒***värde*

tanh(*List1***)⇒***lista*

tanh(*Value1***)** ger den hyperboliska tangensen för argumentet.

tanh(*List1***)** ger en lista på den hyperboliska tangensen för varje element i *List1*.

$\tanh(1.2)$	0.833655
$\tanh(\{0,1\})$	$\{0.,0.761594\}$

tanh(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med hyperbolisk tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

$\tanh^{-1}()$ — Katalog >

**$\tanh^{-1}$(*Value1*)⇒*värde*

**$\tanh^{-1}$(*List1*)⇒*lista*

I Rektangulärt komplext format:

tanh⁻¹() Katalog >

tanh⁻¹(*Value1***)** ger argumentets inversa hyperboliska tangens.

tanh⁻¹(*List1***)** ger en lista på invers hyperbolisk tangens för varje element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `arctanh(...)`.

tanh⁻¹(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med invers hyperbolisk tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

$\tanh^{-1}(0)$ 0.

$\tanh^{-1}(\{1,2.1,3\})$

$\{\text{undef}, 0.518046-1.5708\cdot i, 0.346574-1.570$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

$$\begin{bmatrix} -0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.9473 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901 \end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

tCdf() Katalog >

tCdf(*lowBound***,***upBound***,***df***)⇒***tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten för Student-*t*-fördelning mellan *lowBound* och *upBound* för den specificerade frihetsgraden *df*.

För P(X ≤ *upBound*), sätt *lowBound* = −9**E**999.

Text Katalog >

Text*frågeSträng*[, *DispFlagga*]

Programmeringskommando: Pausar programmet och visar teckensträngen *frågeSträng* i en dialogruta.

När användaren väljer **OK** fortsätter exekveringen av programmet.

Definiera ett program som pausar för att visa vart och ett av fem slumptal i en dialogruta.

Inom mallen Prgm...EndPrgm template, fullborda varje rad genom att trycka på ↵ i stället för enter. På datorns tangentbord, håll ned **Alt** och tryck på **Enter**.

Text

Katalog >

Det valfria argumentet *flagga* kan vara ett valfritt uttryck.

- Om *DispFlagga* utelämnas eller beräknas till **1** sparas textmeddelandet i Räknarens historik.
- Om *DispFlag* beräknas till **0** sparas textmeddelandet inte i historiken.

Om programmet behöver ett svar som skrivs in av användaren, se **Request**, på sidan 126 eller **RequestStr**, på sidan 128.

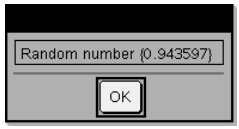
Obs: Du kan använda detta kommando inom ett användardefinierat program, men inte inom en funktion.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:=”Random number “ &
string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

Kör programmet:

```
text_demo()
```

Exempel på en dialogruta:



Then

Se If, på sidan 69.

tInterval

Katalog >

tInterval *List*[**,***Freq*[**,***CLevel*]]

(Indatalista)

tInterval $\bar{x}$**,***sx***,***n***[,***CLevel***]**

(Summary stats indata)

Beräknar ett *t*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall för ett okänt populationsmedelvärde
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.df	Frihetsgrader
stat.σx	Standardavvikelse för urvalet
stat.n	Längden på datasekvenser med urvalsmedelvärde

tInterval_2Samp

Katalog >

tInterval_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*CLevel*[,*Pooled*]]]]

(Indatalista)

tInterval_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*CLevel*[,*Pooled*]]

(Summary stats indata)

Beräknar ett 2-sampel *t*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Pooled=**1** slår samman varianser, *Pooled*=**0** slår inte samman varianser.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\bar{x}$1-$\bar{x}$2	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.df	Frihetsgrader
stat.$\bar{x}$1, stat.$\bar{x}$2	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.σx1, stat.σx2	Urvalets standardavvikelser för *List 1* och *List 2*

Resultatvariabel	Beskrivning
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.sp	Den sammanslagna standardavvikelsen. Beräknas när *Pooled* = YES.

tPdf() Katalog >

tPdf(*XVal***,***df***)⇒***tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen (pdf) för Student-*t*-fördelningen vid ett specificerat *x*-värde med den specificerade frihetsgraden *df*.

trace() Katalog >

trace(*squareMatrix***)⇒***värde*

Ger spåret (summan av alla elementen på huvuddiagonalen) av *squareMatrix*.

$\text{trace}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right)$ 15

$a:=12$ 12

$\text{trace}\left(\begin{bmatrix}a&0\\1&a\end{bmatrix}\right)$ 24

Try Katalog >

Try

block1

Else

block2

EndTry

Exekverar *block1* såvida inte ett fel uppstår. Programexekveringen överförs till *block2* om ett fel uppstår i *block1*. Systemvariabeln *errCode* innehåller felkoden som låter programmet utföra återhämtning efter fel. För en lista på felkoder, se "*Felkoder och meddelanden*" (på sidan 207).

```
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
```

block1 och *block2* kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Exempel 2

För att se kommandona **Try**, **ClrErr** och **PassErr** i arbete, mata in programmet eigenvals() som visas till höger. Kör programmet genom att exekvera vart och ett av följande uttryck.

$$eigenvals\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

Obs: Se även **ClrErr**, på sidan 22 och **PassErr**, på sidan 111.

```
Definiera eigenvals(a,b)=Prgm
© Programmet eigenvals(A,B) visar
egenvärdena för A·B
Try
  Disp "A= ",a
  Disp "B= ",b
  Disp " "
  Disp "Eigenvalues 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

Katalog >

tTest μ*0*,*List*[,*Freq*[,*Hypoth*]]

(Indatalista)

tTest μ*0*,$\bar{x}$,*sx*,*n*,[*Hypoth*]

(Summary stats indata)

tTest

Katalog >

Utför ett hypotestest för ett okänt populationsmedelvärde, μ, när populationens standardavvikelse, σ, är okänd. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Testa H_0: μ = μ0, mot en av följande:

För H_a: μ < μ0, ställ *Hypoth*<0

För H_a: μ ≠ μ0 (förinställning), ställ *Hypoth*=0

För H_a: μ > μ0, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 200).

Resultatvariabel	Beskrivning
stat.t	($\bar{x}$ − μ0) / (stdev / sqrt(n))
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen i *List*
stat.sx	Urvalets standardavvikelse för datasekvensen
stat.n	Urvalets storlek

tTest_2Samp

Katalog >

tTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*[,*Pooled*]]]]

(Indatalista)

tTest_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*Hypoth*[,*Pooled*]]

(Summary stats indata)

Beräknar ett 2-sampel *t*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Testa H_0: μ1 = μ2, mot en av följande:

För H_a: μ1< μ2, ställ *Hypoth*<0

tTest_2Samp

Katalog >

För H_a: μ1≠ μ2 (förinställning), ställ *Hypoth*=0

För H_a: μ1> μ2, ställ *Hypoth*>0

Pooled=**1** slår samman varianser,

Pooled=**0** slår inte samman varianser.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.t	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos t-statistiken
stat.$\bar{x}$1, stat.$\bar{x}$2	Medelvärden hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.n1, stat.n2	Storlek på urvalen
stat.sp	Den sammanslagna standardavvikelsen. Beräknad när *Pooled*=1.

tvmFV()

Katalog >

tvmFV(*N*,*I*,*PV*,*Pmt*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒*värde*

tvmFV(120,5,0,⁻500,12,12) 77641.1

Finansiell funktion som beräknar det framtida värdet på pengar.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 163. Se även **amortTbl()**, på sidan 7.

tvmI()

Katalog >

tvmI(*N*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒*värde*

tvmI(240,100000,⁻1000,0,12,12) 10.5241

Finansiell funktion som beräknar räntesatsen per år.

tvmI() Katalog >

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 163. Se även **amortTbl()**, på sidan 7.

tvmN() Katalog >

tvmN(*I***,***PV***,***Pmt***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])** ⇒*värde*

Finansiell funktion som beräknar antalet betalningsperioder.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 163. Se även **amortTbl()**, på sidan 7.

```
tvmN(5,0,-500,77641,12,12)          120.
```

tvmPmt() Katalog >

tvmPmt(*N***,***I***,***PV***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])** ⇒*värde*

Finansiell funktion som beräknar beloppet på varje betalning.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 163. Se även **amortTbl()**, på sidan 7.

```
tvmPmt(60,4,30000,0,12,12)          -552.496
```

tvmPV() Katalog >

tvmPV(*N***,***I***,***Pmt***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])** ⇒*värde*

Finansiell funktion som beräknar nuvärdet.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 163. Se även **amortTbl()**, på sidan 7.

```
tvmPV(48,4,-500,30000,12,12)        -3426.7
```

TVM-argument*	Beskrivning	Datatyp
N	Antal betalningsperioder	reellt tal

TVM-argument*	Beskrivning	Datatyp
I	Årlig räntesats	reellt tal
PV	Nuvärde	reellt tal
Pmt	Betalningsbelopp	reellt tal
FV	Framtida värde	reellt tal
PpY	Antal betalningar per år, förinställning = 1	heltal > 0
CpY	Ränteperioder per år, förinställning = 1	heltal > 0
$PmtAt$	Betalning som skall betalas i slutet (end) eller i början (beginning) av varje period, förinställning = end	heltal (0 = end, 1 = beginning)

* Dessa argumentnamn avseende tidsjusterat pengavärde påminner om namnen på TVM-variablerna (till exempel, **tvm.pv** och **tvm.pmt**) som används av Finance Solver i applikationen *Calculator*. Finansiella funktioner lagrar dock inte sina argumentvärden eller resultat i TVM-variablerna.

TwoVar

Katalog >

TwoVar X, Y[, [$Freq$] [, $Category$, $Include$]]

Utför tvåvariabelstatistik. En sammanfattning av resultaten visas i variabeln $stat.results$. (Se på sidan 148.)

Alla listor utom $Include$ måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

$Freq$ är en frivillig lista på frekvensvärden. Varje element i $Freq$ specificerar frekvensen för varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

$Category$ är en lista på numeriska eller strängkategorikoder för motsvarande X- och Y-data.

$Include$ är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. Ett tomt element i någon av listorna *X1* till och med *X20* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 200.

Resultatvariabel	**Beskrivning**
stat.$\bar{x}$	Medelvärde av x-värden
stat. x	Summa av x-värden
stat. x2	Summa av x2-värden
stat.sx	Standardavvikelse för x (sampling)
stat. x	Standardavvikelse för x (population)
stat.n	Antal datapunkter
stat.$\bar{y}$	Medelvärde av y-värden
stat. y	Summa av y-värden
stat. y^2	Summa av y2-värden
stat.sy	Standardavvikelse för y (sampling)
stat. y	Populationens standardavvikelse för y
stat. xy	Summa av x ·y-värden
stat.r	Korrelationskoefficient
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.MinY	Minsta y-värde
stat.Q_1Y	Undre kvartil för y
stat.MedY	Median för y
stat.Q_3Y	Övre kvartil för y

Resultatvariabel	Beskrivning
stat.MaxY	Största y-värde
stat. $(x\text{-})^2$	Kvadratsumma för avvikelser från medelvärdet på x
stat. $(y\text{-})^2$	Kvadratsumma för avvikelser från medelvärdet på y

U

unitV() Katalog >

unitV(*Vector1***)⇒***vektor*

Ger en radenhets- eller kolumnenhetsvektor beroende på formen hos *Vector1*.

Vector1 måste vara antingen en enradsmatris eller en enkolumnsmatris.

unitV([1 2 1])	[0.408248 0.816497 0.408248]
unitV([1; 2; 3])	[0.267261; 0.534522; 0.801784]

unLock Katalog >

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

unLock*Var***.**

Låser upp den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.

Se **Lock**, på sidan 87 och **getLockInfo()**, på sidan 65.

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() Katalog >

varPop(*List***[,** *freqList***])⇒***uttryck*

Ger populationsvariansen för *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element.

varPop({5,10,15,20,25,30})	72.9167

varPop() Katalog >

Om ett element i endera listan är tomt ignoreras detta element och även motsvarande element i den andra listan ignoreras. För mer information om tomma element, se på sidan 200.

varSamp() Katalog >

varSamp(*List***[,** *freqList***])⇒***uttryck*

Ger sampelvariansen för *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element.

Om ett element i endera listan är tomt ignoreras detta element och även motsvarande element i den andra listan ignoreras. För mer information om tomma element, se på sidan 200.

$$\text{varSamp}(\{1,2,5,-6,3,-2\}) \qquad \frac{31}{2}$$

$$\text{varSamp}(\{1,3,5\},\{4,6,2\}) \qquad \frac{68}{33}$$

varSamp(*Matrix1***[,** *freqMatrix***])⇒***matris*

Ger en radvektor som innehåller sampelvariansen för varje kolumn i *Matrix1*.

Varje *freqMatrix*-element räknar antalet konsekutiva förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader.

Om ett element i endera matrisen är tomt ignoreras detta element och även motsvarande element i den andra matrisen ignoreras. För mer information om tomma element, se på sidan 200.

$$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right) \qquad \begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$$

$$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix},\begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right) \qquad \begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$$

W

Wait Katalog >

Wait *tidISekunder*

För att vänta 4 sekunder:

Wait 4

Wait

Katalog >

Fördröjer exekvering för en period som varar *tidISekunder* sekunder.

Wait är speciellt användbart i ett program som behöver en kort fördröjning för att begärda data ska bli tillgängliga.

Argumentet *tidISekunder* måste vara ett uttryck som förenklas till ett decimalvärde i intervallet 0 till 100. Kommandot avrundar detta värde till närmaste 0,1 sekunder.

För att avbryta en **Wait** som pågår,

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Du kan använda kommandot **Wait** i ett användardefinierat program, men inte inom en funktion.

För att vänta 1/2 sekund:

Wait 0.5

För att vänta 1,3 sekunder med hjälp av variabeln *sekantal*:

sekantal:=1.3
Wait sekantal

I detta exempel tänds en grön lysdiod i 0,5 sekunder och släcks sedan.

Send "SET GREEN 1 ON"
Wait 0.5
Send "SET GREEN 1 OFF"

warnCodes ()

Katalog >

warnCodes(*Expr1***,** *StatusVar***)⇒***uttryck*

Utvärderar uttrycket *Expr1*, ger resultatet och lagrar eventuellt genererade varningskoder i listvariabeln *StatusVar*. Om inga varningar genereras tilldelar denna funktion *StatusVar* en tom lista.

Expr1 kan vara ett valfritt giltigt matematiskt uttryck i TI-Nspire™ eller TI-Nspire™ CAS. Du kan inte använda ett kommando eller en tilldelning som *Expr1*.

StatusVar måste vara ett giltigt variabelnamn.

För en lista på varningskoder och tillhörande meddelanden, se på sidan 216.

```
⚠ warnCodes(det([1.23456E-999]),warn)
                                1.23456E-999
warn                                {10029}
```

when() Katalog >

when(*Condition***,** *trueResult* [**,** *falseResult*] [**,** *unknownResult*]**)** ⇒*uttryck*

Ger *trueResult*, *falseResult* eller *unknownResult* beroende på om *Condition* är sant, falskt eller okänt. Ger indata om det finns alltför få argument för att specificera ett lämpligt resultat.

Utelämna både *falseResult* och *unknownResult* för att skapa ett uttryck som är definierat endast i området där *Condition* är sant.

$\text{when}(x<0,x+3)\|x=5$	undef

Använd ett **undef** *falseResult* för att definiera ett uttryck som plottas endast i ett intervall.

when() är användbar för att definiera rekursiva funktioner.

$\text{when}(n>0,n\cdot factoral(n-1),1)\rightarrow factoral(n)$	*Done*
$factoral(3)$	6
$3!$	6

While Katalog >

While *Condition*

Block

EndWhile

Exekverar påståendena i *Block* så länge *Condition* är sant.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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
```

xor

Katalog >

BooleskUttr1 **xor** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **xor** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **xor** *BooleskMatris2* ger *Boolesk matris*

Ger resultatet sant om *BooleanExpr1* är sant och *BooleanExpr2* är falskt, eller vice versa.

Ger resultatet falskt om båda argumenten är sanna eller falska. Ger ett förenklat booleskt uttryck om något av argumenten inte kan lösas om till sant eller falskt.

true xor true	false
5>3 xor 3>5	true

Obs: Se **eller**, på sidan 109.

Integer1 **xor** *Integer2* ⇒ *heltal*

Jämför två reella heltal bit för bit med en **xor**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om en bit (men inte båda) är 1. Resultatet blir 0 om båda bitarna är 0 eller 1. Det erhållna värdet representerar bitresultaten och visas enligt det inställda basläget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▶**Base2**, på sidan 16.

Obs: Se **eller**, på sidan 109.

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

0h7AC36 xor 0h3D5F	0h79169

I binärt basläge:

0b100101 xor 0b100	0b100001

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

zInterval

Katalog >

zInterval σ,*List*[,*Freq*[,*CLevel*]]

(Indatalista)

zInterval σ,x̄,*n* [,*CLevel*]

(Summary stats indata)

Beräknar ett *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall för ett okänt populationsmedelvärde
stat.x̄	Urvalsmedelvärde för datasekvensen från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.sx	Standardavvikelse för urvalet
stat.n	Längden på datasekvenser med urvalsmedelvärde
stat.σ	Känd populationsstandardavvikelse för datasekvensen *List*

zInterval_1Prop

Katalog >

zInterval_1Prop *x*,*n* [,*CLevel*]

Beräknar ett 1-proportion *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

x är ett icke negativt heltal.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\hat{p}$	Den beräknade proportionen lyckade försök
stat.ME	Felmarginal
stat.n	Antalet samplingar i datasekvens

zInterval_2Prop

Katalog >

zInterval_2Prop *x1*,*n1*,*x2*,*n2*[,*CLevel*]

Beräknar ett 2-proportion *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

x1 och *x2* är icke negativa heltal.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\hat{p}$Diff	Den beräknade skillnaden mellan proportioner
stat.ME	Felmarginal
stat.$\hat{p}$1	Proportionsuppskattning för första urvalet
stat.$\hat{p}$2	Proportionsuppskattning för andra urvalet
stat.n1	Urvalets storlek i datasekvens ett
stat.n2	Urvalets storlek i datasekvens två

zInterval_2Samp

Katalog >

zInterval_2Samp σ_1,σ_2,*List1*,*List2*[,*Freq1*[,*Freq2*,[*CLevel*]]]]

(Indatalista)

zInterval_2Samp σ_1,σ_2,$\bar{x}$*1*,*n1*,$\bar{x}$*2*,*n2*[,*CLevel*]

zInterval_2Samp

Katalog >

(Summary stats indata)

Beräknar ett 2-sampel *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\bar{x}$1-$\bar{x}$2	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.$\bar{x}$1, stat.$\bar{x}$2	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.σx1, stat.σx2	Urvalets standardavvikelser för *List 1* och *List 2*
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.r1, stat.r2	Kända populationsstandardavvikelser för datasekvenserna *List 1* och *List 2*

zTest

Katalog >

zTest μ*0*,σ,*List*,[*Freq*[,*Hypoth*]]

(Indatalista)

zTest μ*0*,σ,$\bar{x}$,*n*[,*Hypoth*]

(Summary stats indata)

Utför ett *z*-test med frekvensen *freqlist*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Testa H_0: μ = μ0, mot en av följande:

För H_a: μ < μ0, ställ *Hypoth*<0

För H_a: μ ≠ μ0 (förinställning), ställ *Hypoth*=0

För H_a: μ > μ0, ställ *Hypoth*>0

zTest

Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.z	$(\bar{x} - \mu 0) / (\sigma / \text{sqrt}(n))$
stat.P Value	Lägsta sannolikhet vid vilken nollhypotesen kan förkastas
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen i *List*
stat.sx	Urvalets standardavvikelse för datasekvensen. Ges endast för *Data*-inmatningen.
stat.n	Urvalets storlek

zTest_1Prop

Katalog >

zTest_1Prop *p0*,*x*,*n*[,*Hypoth*]

Beräknar ett 1-proportion *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

x är ett icke negativt heltal.

Testa H_0: $p = p0$ mot en av följande:

För H_a: $p > p0$, ställ *Hypoth*>0

För H_a: $p \neq p0$ *(förinställning)*, ställ *Hypoth*=0

För H_a: $p < p0$, ställ *Hypoth*<0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.p0	Hypotetisk populationsproportion
stat.z	Standardnormalvärde beräknat för proportionen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\hat{p}$	Uppskattad urvalsproportion
stat.n	Urvalets storlek

zTest_2Prop Katalog >

zTest_2Prop *x1*,*n1*,*x2*,*n2*[,*Hypoth*]

Beräknar ett 2-proportion *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

x1 och *x2* är icke negativa heltal.

Testa H_0: $p1 = p2$ mot en av följande:

För H_a: $p1 > p2$, ställ *Hypoth*>0

För H_a: $p1 \neq p2$ *(förinställning)*, ställ *Hypoth*=0

För H_a: $p < p0$, ställ *Hypoth*<0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan proportioner
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\hat{p}1$	Proportionsuppskattning för första urvalet
stat.$\hat{p}2$	Proportionsuppskattning för andra urvalet
stat.$\hat{p}$	Uppskattning av sammanslagna urvalsproportioner
stat.n1, stat.n2	Antal samplingar i försöksomgång 1 och 2

zTest_2Samp Katalog >

zTest_2Samp σ_1,σ_2, *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

(Indatalista)

zTest_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*Hypoth*]

(Summary stats indata)

Beräknar ett 2-sampel *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 148.)

Testa H_0: $\mu1 = \mu2$, mot en av följande:

För H_a: $\mu1 < \mu2$, ställ *Hypoth*<0

För H_a: μ1 ≠ μ2 (förinställning), ställ *Hypoth*=0

För H_a: μ1 > μ2, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 200).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\bar{x}$1, stat.$\bar{x}$2	Medelvärden hos urvalet i datasekvenserna i *List1* och *List2*
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List1* och *List2*
stat.n1, stat.n2	Storlek på urvalen

Symboler

+ (addera) — [+] tangent

Value1 + *Value2*⇒*värde*

Ger summan av de två argumenten.

56	56
56+4	60
60+4	64
64+4	68
68+4	72

List1 + *List2*⇒*lista*

Matrix1 + *Matrix2*⇒*matris*

Ger en lista (eller matris) som innehåller summorna av motsvarande element i *List1* och *List2* (eller *Matrix1* och *Matrix2*).

Argumenten måste ha samma dimensioner.

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\{22,3.14159,1.5708\}$
$\left\{10,5,\frac{\pi}{2}\right\}\rightarrow l2$	$\{10,5,1.5708\}$
$l1+l2$	$\{32,8.14159,3.14159\}$

Value + *List1*⇒*lista*

List1 + *Value*⇒*lista*

Ger en lista på summorna av *Value* och varje element i *List1*.

$15+\{10,15,20\}$	$\{25,30,35\}$
$\{10,15,20\}+15$	$\{25,30,35\}$

Value + *Matrix1*⇒*matris*

Matrix1 + *Value*⇒*matris*

Ger en matris med *Value* adderat till varje element i diagonalen av *Matrix1*. *Matrix1* måste vara kvadratisk.

Obs: Använd **.+** (punkt plus) för att lägga till ett uttryck till varje element.

$20+\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\3 & 24\end{bmatrix}$

−(subtrahera) — [−] tangent

Value1 − *Value2*⇒*värde*

Ger *Value1* minus *Value2*.

$6-2$	4
$\pi-\frac{\pi}{6}$	2.61799

List1 − *List2*⇒*lista*

Matrix1 − *Matrix2*⇒*matris*

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,{}^{-}1.85841,0.\}$
$[3\ \ 4]-[1\ \ 2]$	$[2\ \ 2]$

−(subtrahera) — [−] tangent

Subtraherar varje element i *List2* (eller *Matrix2*) från motsvarande element i *List1* (eller *Matrix1*) och ger resultatet.

Argumenten måste ha samma dimensioner.

Value − List1⇒lista

List1 − Value⇒lista

$15-\{10,15,20\}$	$\{5,0,-5\}$
$\{10,15,20\}-15$	$\{-5,0,5\}$

Subtraherar varje element i *List1* från *Value* eller subtraherar *Value* från varje element i *List1* och ger en lista på resultaten.

Value − Matrix1⇒matris

Matrix1 − Value⇒matris

$20-\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}19 & -2\\-3 & 16\end{bmatrix}$

Value − Matrix1 ger en matris över *Value* gånger enhetsmatrisen minus *Matrix1*. *Matrix1* måste vara kvadratisk.

Matrix1 − Value ger en matris över *Value* gånger enhetsmatrisen subtraherad från *Matrix1*. *Matrix1* måste vara kvadratisk.

Obs: Använd **.−** (punkt minus) för att subtrahera ett uttryck från varje element.

·(multiplicera) — [×] tangent

Value1 ·Value2⇒värde

$2\cdot 3.45$	6.9

Ger produkten av de två argumenten.

List1·List2⇒lista

$\{1.,2,3\}\cdot\{4,5,6\}$	$\{4,10,18\}$

Ger en lista som innehåller produkterna av motsvarande element i *List1* och *List2*.

Listorna måste ha samma dimensioner.

Matrix1 ·Matrix2⇒matris

$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}.\begin{bmatrix}7 & 8\\7 & 8\\7 & 8\end{bmatrix}$	$\begin{bmatrix}42 & 48\\105 & 120\end{bmatrix}$

Ger en matris över produkten av *Matrix1* och *Matrix2*.

Antalet kolumner i *Matrix1* måste vara lika med antalet rader i *Matrix2*.

·(multiplicera) — ×tangent

Value ·*List1*⇒*lista*

List1 ·*Value*⇒*lista*

Ger en lista på produkterna av *Value* och varje element i *List1*.

$\pi\cdot\{4,5,6\}$ → $\{12.5664,15.708,18.8496\}$

Value ·*Matrix1*⇒*matris*

Matrix1 ·*Value*⇒*matris*

Ger en matris över produkterna av *Value* och varje element i *Matrix1*.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\cdot 0.01$ → $\begin{bmatrix}0.01 & 0.02\\0.03 & 0.04\end{bmatrix}$

$6\cdot\text{identity}(3)$ → $\begin{bmatrix}6 & 0 & 0\\0 & 6 & 0\\0 & 0 & 6\end{bmatrix}$

Obs: Använd **.**·(punkt multiplicera) för att multiplicera ett uttryck med varje element.

/ (dividera) — ÷tangent

Value1 / *Value2*⇒*värde*

Ger kvoten av *Value1* dividerat med *Value2*.

$\frac{2}{3.45}$ → .57971

Obs: Se även **Fraction template**, på sidan 1.

List1 / *List2*⇒*lista*

Ger en lista på kvoterna av *List1* dividerat med *List2*.

Listorna måste ha samma dimensioner.

$\frac{\{1.,2,3\}}{\{4,5,6\}}$ → $\left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$

Value / *List1* ⇒ *lista*

List1 / *Value* ⇒ *lista*

Ger en lista på kvoterna av *Value* dividerat med *List1* eller*List1* dividerat med *Value*.

$\frac{6}{\{3,6,\sqrt{6}\}}$ → $\{2,1,2.44949\}$

$\frac{\{7,9,2\}}{7\cdot 9\cdot 2}$ → $\left\{\frac{1}{18},\frac{1}{14},\frac{1}{63}\right\}$

Value / *Matrix1*⇒ *matris*

Matrix1 / *Value* ⇒ *matris*

Ger en matris över kvoterna av *Matrix1*/*Value*.

$\frac{[7\ \ 9\ \ 2]}{7\cdot 9\cdot 2}$ → $\begin{bmatrix}\frac{1}{18} & \frac{1}{14} & \frac{1}{63}\end{bmatrix}$

Obs: Använd **. /** (punkt dividera) för att dividera ett uttryck med varje element.

Value1 ^ *Value2* ⇒ *värde*

List1 **^** *List2* ⇒ *lista*

Ger det första argumentet upphöjt till det andra argumentets potens.

Obs: Se även **Exponent template**, på sidan 1.

Ger, för en lista, elementen i *List1* upphöjda till potensen för motsvarande element i *List2*.

I det reella området använder potenser i bråkform som har reducerade exponenter med udda nämnare den reella delen kontra principaldelen för komplext läge.

4^2 → 16

$\{2,4,6\}^{\{1,2,3\}}$ → $\{2,16,216\}$

Value ^ *List1* ⇒ *lista*

Ger *Value* upphöjt till potensen för elementen i *List1*.

$\pi^{\{1,2,-3\}}$ → $\{3.14159,9.8696,0.032252\}$

List1 ^ *Value* ⇒ *lista*

Ger elementen i *List1* upphöjda till potensen för *Value*.

$\{1,2,3,4\}^{-2}$ → $\left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$

squareMatrix1 **^** *integer* ⇒ *matris*

Ger *squareMatrix1* upphöjd till potensen för *integer* (heltal).

squareMatrix1 måste vara en kvadratisk matris.

Om *integer* = ⁻1 beräknas den inversa matrisen.

Om *integer* < ⁻1 beräknas den inversa matrisen till en lämplig positiv potens.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^2$ → $\begin{bmatrix}7 & 10\\15 & 22\end{bmatrix}$

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-1}$ → $\begin{bmatrix}-2 & 1\\\frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-2}$ → $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\\frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x^2 (kvadrat) — [x²] tangent

Value1^2 ⇒ *värde*

Ger kvadraten på argumentet.

List1^2 ⇒ *lista*

Ger en lista med kvadraterna på elementen i *List1*.

squareMatrix1^2 ⇒ *matris*

Ger en matris över kvadraten på *squareMatrix1*. Detta är inte detsamma som att beräkna kvadraten på varje element. Använd .^2 för att beräkna kvadraten på varje element.

4^2	16
$\{2,4,6\}^2$	$\{4,16,36\}$
$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}^2$	$\begin{bmatrix}40&64&88\\49&79&109\\58&94&130\end{bmatrix}$
$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}.^\wedge 2$	$\begin{bmatrix}4&16&36\\9&25&49\\16&36&64\end{bmatrix}$

.+ (punkt addera) — [.] [+] tangenter

Matrix1 **.+** *Matrix2* ⇒ *matris*

Value **.+** *Matrix1* ⇒ *matris*

Matrix1 **.+** *Matrix2* ger en matris som är summan av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Value **.+** *Matrix1* ger en matris som är summan av *Value* och varje element i *Matrix1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix}.+\begin{bmatrix}10&30\\20&40\end{bmatrix}$	$\begin{bmatrix}11&32\\23&44\end{bmatrix}$
$5.+\begin{bmatrix}10&30\\20&40\end{bmatrix}$	$\begin{bmatrix}15&35\\25&45\end{bmatrix}$

.− (punkt subtrahera) — [.] [−] tangenter

Matrix1 **.−** *Matrix2* ⇒ *matrixs*

Value **.−***Matrix1* ⇒ *matris*

Matrix1 **.−***Matrix2* ger en matris som är skillnaden mellan varje par av motsvarande element i *Matrix1* och *Matrix2*.

Value **.−***Matrix1* ger en matris som är skillnaden mellan *Value* och varje element i *Matrix1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix}.-\begin{bmatrix}10&20\\30&40\end{bmatrix}$	$\begin{bmatrix}-9&-18\\-27&-36\end{bmatrix}$
$5.-\begin{bmatrix}10&20\\30&40\end{bmatrix}$	$\begin{bmatrix}-5&-15\\-25&-35\end{bmatrix}$

. ·(punkt multiplicera) [.] [×] tangenter

Matrix1 .· *Matrix2* ⇒ *matris*

Value .· *Matrix1* ⇒ *matris*

Matrix1 .· *Matrix2* ger en matris som är produkten av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Value .· *Matrix1* ger en matris som innehåller produkterna av *Value* och varje element i *Matrix1*.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} .\cdot \begin{bmatrix}10 & 20\\30 & 40\end{bmatrix} \qquad \begin{bmatrix}10 & 40\\90 & 160\end{bmatrix}$$

$$5 .\cdot \begin{bmatrix}10 & 20\\30 & 40\end{bmatrix} \qquad \begin{bmatrix}50 & 100\\150 & 200\end{bmatrix}$$

. / (punkt dividera) [.] [÷] tangenter

Matrix1 . / *Matrix2* ⇒ *matris*

Value . / *Matrix1*⇒*matris*

Matrix1 . / *Matrix2* ger en matris som är kvoten av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Value . / *Matrix1* ger en matris som är kvoten av *Value* och varje element i *Matrix1*.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} ./ \left(\begin{bmatrix}10 & 20\\30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{10} & \frac{1}{10}\\\frac{1}{10} & \frac{1}{10}\end{bmatrix}$$

$$5 ./ \left(\begin{bmatrix}10 & 20\\30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{2} & \frac{1}{4}\\\frac{1}{6} & \frac{1}{8}\end{bmatrix}$$

.^ (punkt potens) [.] [^] tangenter

Matrix1 .^ *Matrix2* ⇒ *matris*

Value . ^ *Matrix1* ⇒ *matris*

Matrix1 .^ *Matrix2* ger en matris där varje element i *Matrix2* är exponenten för motsvarande element i *Matrix1*.

Value .^ *Matrix1* ger en matris där varje element i *Matrix1* är exponenten för *Value*.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} .^\wedge \begin{bmatrix}0 & 2\\3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 4\\27 & \frac{1}{4}\end{bmatrix}$$

$$5 .^\wedge \begin{bmatrix}0 & 2\\3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 25\\125 & \frac{1}{5}\end{bmatrix}$$

-(negation) [(−)] tangent

-*Value1* ⇒ *värde*

-*List1* ⇒ *lista*

-*Matrix1* ⇒ *matris*

-2.43	-2.43
-{-1,0.4,1.2E19}	{1.,-0.4,-1.2E19}

⁻(negation) (–) tangent

Ger argumentets negation.

Ger, för en lista eller matris, alla elementen negerade.

Om argumentet är ett binärt eller hexadecimalt heltal ger negationen tvåkomplementet.

```
I binärt basläge:
```

Viktigt: Noll, inte bokstaven O.

```
⁻0b100101
0b11111111111111111111111111111111▸
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

% (procent) ctrl 📖 tangenter

Value1 % ⇒ *värde*

List1 **%** ⇒ *lista*

Matrix1 **%** ⇒ *matris*

Ger $\frac{argument}{100}$

Ger, för en lista eller matris, en lista eller matris med varje element dividerat med 100.

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

13%	0.13

{{1,10,100}}%	{0.01,0.1,1.}

= (lika med) = tangent

Expr1 **=** *Expr2*⇒*Booleskt uttryck*

List1 **=** *List2*⇒ *Boolesk lista*

Matrix1 **=** *Matrix2*⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms vara lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms inte vara lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Exempel på funktion som använder matchande testsymboler: =, ≠, <, ≤, >, ≥

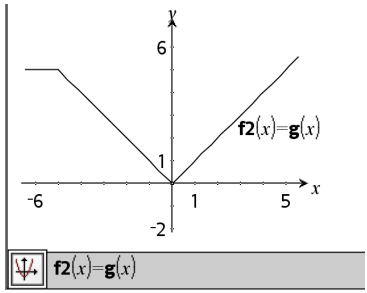
```
Define g(x)=Func
        If x≤⁻5 Then
        Return 5
        ElseIf x>⁻5 and x<0 Then
        Return ⁻x
        ElseIf x≥0 and x≠10 Then
        Return x
        ElseIf x=10 Then
        Return 3
        EndIf
        EndFunc
                                Done
```

= (lika med)

[=] tangent

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Resultat från plottning av g(x)



≠ (inte lika med)

[ctrl] [=] tangenter

Expr1 ≠ *Expr2* ⇒ *Booleskt uttryck*

List1 ≠ *List2* ⇒ *Boolesk lista*

Matrix1 ≠ *Matrix2* ⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms inte vara lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **/=**

Se exemplet "=" (lika med).

< (mindre än)

[ctrl] [=] tangenter

Expr1 < *Expr2* ⇒ *Booleskt uttryck*

List1 < *List2* ⇒ *Boolesk lista*

Matrix1 < *Matrix2* ⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms vara mindre än *Expr2*.

Se exemplet "=" (lika med).

< (mindre än)

ctrl = tangenter

Ger resultatet falskt om *Expr1* bestäms vara större än eller lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

≤ (mindre än eller lika med)

ctrl = tangenter

$Expr1 \le Expr2 \Rightarrow Booleskt\ uttryck$

$List1 \le List2 \Rightarrow Boolesk\ lista$

$Matrix1 \le Matrix2 \Rightarrow Boolesk\ matris$

Ger resultatet sant om *Expr1* bestäms vara mindre än eller lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara större än *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **<=**

Se exemplet "=" (lika med).

> (större än)

ctrl = tangenter

$Expr1 > Expr2 \Rightarrow Booleskt\ uttryck$

$List1 > List2 \Rightarrow Boolesk\ lista$

$Matrix1 > Matrix2 \Rightarrow Boolesk\ matris$

Ger resultatet sant om *Expr1* bestäms vara större än *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara mindre än eller lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Se exemplet "=" (lika med).

> (större än)

ctrl = tangenter

Ger, för listor och matriser, jämförelser element för element.

≥ (större än eller lika med)

ctrl = tangenter

Expr1 ≥ *Expr2* ⇒ *Booleskt uttryck*

List1 ≥ *List2* ⇒ *Boolesk lista*

Matrix1 ≥ *Matrix2* ⇒ *Boolesk matris*

Ger resultatet falskt om *Expr1* bestäms vara större än eller lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara mindre än *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **>=**

Se exemplet "=" (lika med).

⇒ (logisk implikation)

ctrl = knappar

BooleskUttr1 ⇒ *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 ⇒ *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 ⇒ *BooleskMatris2* ger *Boolesk matris*

Heltal1 ⇒ *Heltal2* ger *Heltal*

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}

Beräknar uttrycket **not <argument1> or <argument2>** och ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **=>**

⇔ (logisk dubbel implikation, XNOR)

ctrl = knappar

BooleskUttr1 ⇔ *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 ⇔ *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 ⇔ *BooleskMatris2* ger *Boolesk matris*

Heltal1 ⇔ *Heltal2* ger *Heltal*

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}

Ger negation av en **XOR** Boolesk operation på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **<=>**

! (fakultet)

?!▸ tangent

Value1! ⇒ *värde*

List1! ⇒ *lista*

Matrix1! ⇒ *matris*

5!	120
{{5,4,3}}!	{120,24,6}
$\left\{\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right\}!$	$\begin{bmatrix}1 & 2\\6 & 24\end{bmatrix}$

Ger argumentets fakultet.

Ger, för en lista eller matris, en lista eller matris med elementens fakulteter.

& (lägg till)

ctrl 📖 tangenter

String1 & *String2* ⇒ *sträng*

"Hello "&"Nick"	"Hello Nick"

Ger en textsträng som är *String2* bifogad till *String1*.

d(*Uttr1*, *Var*[, *Ordning*]) | *Var=Värde⇒värde*

d(*Uttr1*, *Var*[, *Ordning*])⇒*värde*

d(*Lista1*, *Var*[, *Ordning*])⇒*lista*

d(*Matris1*, *Var*[, *Ordning*])⇒*matris*

Förutom när den första syntaxen används måste du lagra ett numeriskt värde i variabeln *Var* innan du beräknar d**()**. Se exempel.

d() kan användas för att numeriskt, med automatiska deriveringsmetoder, beräkna första- och andraderivatan i en punkt.

Ordning, om inkluderad, måste vara =**1** eller **2**. Det förinställda värdet är **1**.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva `derivative(...)`.

Obs: Algoritmen **d()** har en begränsning: den arbetar rekursivt genom det ej förenklade uttrycket och beräknar det numeriska värdet för förstaderivatan (och andraderivatan om tillämpligt) samt beräknar varje deluttryck, vilket kan leda till ett oväntat resultat.

Studera exemplet till höger. Förstaderivatan av x·(x^2+x)^(1/3) för x=0 är lika med 0. Men eftersom förstaderivatan av deluttrycket (x^2+x)^(1/3) är odefinierat för x=0, och detta värde används för att beräkna derivatan av hela uttrycket, anger **d()** resultatet som odefinierat och visar ett varningsmeddelande.

Om du stöter på denna begränsning, verifiera lösningen grafiskt. Du kan också prova att använda **centralDiff()**.

$\frac{d}{dx}(|x|)|x=0$ undef

$x:=0:\frac{d}{dx}(|x|)$ undef

$x:=3:\frac{d}{dx}(\{x^2,x^3,x^4\})$ $\{6,27,108\}$

$\frac{d}{dx}\left(x\cdot(x^2+x)^{\frac{1}{3}}\right)|x=0$ undef

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

∫() (integral) Katalog >

∫(*Uttr1*, *Var*, *Undre*, *Övre*) ⇒ *värde*

Ger integralen av *Uttr1* med avseende på *Var* från *Undre* till *Övre*. Kan användas för att numeriskt beräkna den bestämda integralen med samma metod som nInt().

$$\int_0^1 x^2\,dx \qquad 0.333333$$

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`integral(...)`**.

Obs: Se även **nInt()**, på sidan 103 och **Mallen för bestämd integral**, på sidan 6.

√() (kvadratrot) ctrl x² tangenter

√ (*Value1*)⇒*värde*

√ (*List1*)⇒*lista*

Ger kvadratroten ur argumentet.

$$\sqrt{4} \qquad 2$$

$$\sqrt{\{9,2,4\}} \qquad \{3,1.41421,2\}$$

Ger, för en lista, kvadratrötterna ur alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`sqrt(...)`**

Obs: Se även **Square root template**, på sidan 1.

Π() (prodSeq) Katalog >

Π(*Expr1*, *Var*, *Low*, *High*)⇒*uttryck*

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`prodSeq(...)`**.

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

$$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right) \qquad \left\{\frac{1}{120},120,32\right\}$$

Utvärderar *Expr1* för varje värde på *Var* från *Low* till *High* och ger produkten av resultaten.

Obs: Se även **Product template (Π)**, på sidan 5.

Π() (prodSeq) Katalog >

Π(*Expr1*, *Var*, *Low*, *Low*−1)⇒1

Π(*Expr1*, *Var*, *Low*, *High*) ⇒**1/**Π(*Expr1*, *Var*, *High*+*1*, *Low*−*1*) om *High* < *Low*−*1*

Produktformlerna som används har härletts från följande referens:

Ronald L. Graham, Donald E. Knuth och Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\prod_{k=4}^{3}(k) \qquad 1$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right) \qquad 6$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right) \qquad \frac{1}{4}$$

Σ() (sumSeq) Katalog >

Σ(*Expr1*, *Var*, *Low*, *High*)⇒*uttryck*

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`sumSeq(...)`**.

Utvärderar *Uttr1* för varje värde på *Var* från *Låg* till *Hög* och ger summan av resultaten.

Obs: Se även **Sum template**, på sidan 5.

Σ(*Expr1*, *Var*, *Low*, *Low*−*1*)⇒0

Σ(*Expr1*, *Var*, *Low*, *High*) ⇒⁻Σ(*Expr1*, *Var*, *High*+*1*, *Low*−1) om *High* < *Low*−*1*

Summaformlerna som används har härletts från följande referens:

Ronald L. Graham, Donald E. Knuth och Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\sum_{n=1}^{5}\left(\frac{1}{n}\right) \qquad \frac{137}{60}$$

$$\sum_{k=4}^{3}(k) \qquad 0$$

$$\sum_{k=4}^{1}(k) \qquad -5$$

$$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k) \qquad 4$$

ΣInt() Katalog >

ΣInt(*NPmt1*, *NPmt2*, *N*, *I*, *PV* ,[*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*]**)** ⇒*värde*

ΣInt(*NPmt1*,*NPmt2*,*amortTable***)**⇒*värde*

Amorteringsfunktion som beräknar räntesumman under ett specificerat område av betalningar.

NPmt1 och *NPmt2* definierar start och slut på betalningsområdet.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 163.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

ΣInt(*NPmt1,NPmt2,amortTable***)** beräknar räntesumman baserat på amorteringstabellen *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 7.

Obs: Se även ΣPrn() nedan och **Bal()**, på sidan 15.

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() Katalog >

ΣPrn(*NPmt1*, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*]**)**⇒*värde*

ΣPrn(*NPmt1*,*NPmt2*,*amortTable***)**⇒*värde*

Amorteringsfunktion som beräknar kapitalsumman under ett specificerat område av betalningar.

ΣPrn(1,3,12,4.75,20000,,12,12) -4916.28

ΣPrn() Katalog >

NPmt1 och *NPmt2* definierar start och slut på betalningsområdet.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 163.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

ΣPrn(*NPmt1,NPmt2,amortTable***)** beräknar summan av betalt kapital baserat på amorteringstabellen *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 7.

Obs: Se även ΣInt() ovan och **Bal()**, på sidan 15.

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

(indirection) ctrl tangenter

*varNameString*

Avser variabeln vars namn är *varNameString*. Detta låter dig använda strängar för att skapa variabelnamn inom en funktion.

xyz:=12 12

#("x"&"y"&"z") 12

Skapar eller avser variabeln xyz.

10→*r* 10

"r"→*s1* "r"

#*s1* 10

Ger värdet på variabeln (r) vars namn är lagrat i variabeln s1.

E (grundpotensform)

EE tangent

*mantissa*E*exponent*

Skriver in ett tal i grundpotensform. Talet tolkas som *mantissa* × $10^{exponent}$.

Tips: Om du vill skriva in en potens av 10 utan att orsaka ett decimalt resultat, använd 10^*integer*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@E`. Skriv exempelvis `2.3@E4` för att mata in 2.3E4.

23000.	23000.
2300000000.+4.1E15	4.1E15
$3\cdot 10^4$	30000

g (nygrad)

π▸ tangent

*Expr1*g⇒*uttryck*

*List1*g⇒*lista*

*Matrix1*g⇒*matris*

Denna funktion ger dig ett sätt att specificera en vinkel i nygrader när du är i läge Grader eller Radianer.

Multiplicerar i vinkelläget Radianer *Expr1* med π/200.

Multiplicerar i vinkelläget Grader *Expr1* med g/100.

Ger i läget Nygrader *Expr1* oförändrat.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva `@g`.

I vinkelläge Grader, Nygrader eller Radianer:

$\cos(50^g)$	0.707107
$\cos(\{0,100^g,200^g\})$	$\{1,0.,-1.\}$

r (radian)

π▸ tangent

*Value1*r ⇒*värde*

*List1*r ⇒*lista*

*Matrix1*r ⇒*matris*

Denna funktion ger dig ett sätt att specificera en vinkel i radianer när du är i läge Grader eller Nygrader.

I vinkelläge Grader, Nygrader eller Radianer:

$\cos\left(\frac{\pi}{4^r}\right)$	0.707107
$\cos\left(\left\{0^r,\left(\frac{\pi}{12}\right)^r,-(\pi)^r\right\}\right)$	$\{1,0.965926,-1.\}$

ʳ (radian)

π▸ tangent

Multiplicerar i vinkelläget Grader argumentet med 180/π.

Ger i vinkelläget Radianer argumentet oförändrat.

Multiplicerar i vinkelläget Nygrader argumentet med 200/π.

Tips: Använd ʳ om du vill framtvinga radianer i en funktionsdefinition oavsett det inställda läget när funktionen används.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@r**.

° (grader)

π▸ tangent

Value1°⇒värde

List1°⇒lista

Matrix1°⇒matris

Denna funktion ger dig ett sätt att specificera en vinkel i grader när du är i läge Nygrader eller Radianer.

Multiplicerar i vinkelläget Radianer argumentet med π/180.

Ger i vinkelläget Grader argumentet oförändrat.

Multiplicerar i vinkelläget Nygrader argumentet med 10/9.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@d**.

I vinkelläge Grader, Nygrader eller Radianer:

$\cos(45°)$	0.707107

I vinkelläget Radianer:

°, ', " (grad/minut/sekund)

ctrl 📖 tangenter

dd°mm'ss.ss"⇒uttryck

*dd*Ett positivt eller negativt tal

*mm*Ett icke negativt tal

*ss.ss*Ett icke negativt tal

I vinkelläget Grader:

25°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

°, ', '' (grad/minut/sekund)

ctrl 📖 tangenter

Ger $dd+(mm/60)+(ss.ss/3600)$.

Med detta bas-60 inmatningsformat kan du:

- Skriva in en vinkel i grader/minuter/sekunder oavsett det inställda vinkelläget.
- Skriva in tid som timmar/minuter/sekunder.

Obs: Avsluta ss.ss med två apostrofer (''), inte med citationstecken ("').

∠ (vinkel)

ctrl 📖 tangenter

$[Radius,\angle\theta_Angle]\Rightarrow vektor$

(polär indata)

$[Radius,\angle\theta_Angle,Z_Coordinate]\Rightarrow vektor$

(cylindrisk indata)

$[Radius,\angle\theta_Angle,\angle\theta_Angle]\Rightarrow vektor$

(sfärisk indata)

Ger koordinater som en vektor beroende på det inställda vektorformatläget: rektangulär, cylindrisk eller sfärisk.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva `@<`.

I vinkelläget Radianer och med vektorformatet inställt på:

rektangulär

$[5\ \ \angle 60°\ \ \angle 45°]$

$[1.76777\ \ 3.06186\ \ 3.53553]$

cylindrisk

$[5\ \ \angle 60°\ \ \angle 45°]$

$[3.53553\ \ \angle 1.0472\ \ 3.53553]$

sfärisk

$[5\ \ \angle 60°\ \ \angle 45°]$

$[5.\ \ \angle 1.0472\ \ \angle 0.785398]$

$(Magnitude\ \angle\ Angle)\Rightarrow complexValue$

(polär indata)

Matar in ett komplext värde i (r∠θ) polär form. *Angle* tolkas enligt det inställda vinkelläget.

I vinkelläget Radianer och i Rektangulärt komplext format:

_ (understrykningstecken som ett tomt element)

Se “Tomma element” (på sidan 200).

10^()

Katalog >

10^ (*Value1***)⇒***värde*

10^ (*List1***)⇒***lista*

Ger 10 upphöjd till argumentets potens.

Ger, för en lista, 10 upphöjd till potensen för elementen i *List1*.

$10^{1.5}$ → 31.6228

10^(*squareMatrix1***)⇒***kvadratMatris*

Ger 10 upphöjd till potensen för *squareMatrix1*. Detta är inte detsamma som att beräkna 10 upphöjd till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

$$10^{\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}} \rightarrow \begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

^⁻¹(inverterat värde)

Katalog >

Value1 **^⁻¹⇒***värde*

List1 **^⁻¹⇒***lista*

Ger argumentets inverterade värde.

Ger, för en lista, de inverterade värdena på elementen i *List1*.

$(3.1)^{-1}$ → 0.322581

squareMatrix1 **^⁻¹⇒***kvadratMatris*

Ger inversen av *squareMatrix1*.

squareMatrix1 måste vara en icke singulär kvadratisk matris.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \rightarrow \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

| (operatorn begränsning)

ctrl 📖 tangenter

Uttr **|** *BoolesktUttr1* [**and***BoolesktUttr2*]...

Uttr **|** *BoolesktUttr1* [**or***BoolesktUttr2*]...

$x+1|x=3$ → 4

$x+55|x=\sin(55)$ → 54.0002

| (operatorn begränsning)

ctrl 📖 **tangenter**

("|")-symbolen begränsning fungerar som en binär operator. Operanden till vänster om | är ett uttryck. Operanden till höger om | specificerar ett eller flera relationer som är avsedda att påverka förenklingen av uttrycket. Flera relationer efter | måste förbindas med ett logiskt "**and**" eller "**or**"-operatorer.

Operatorn begränsning ger tre bastyper av funktionalitet:

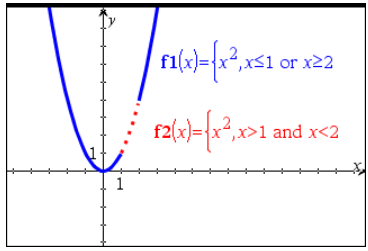
- Substitutioner
- Intervallbegränsningar
- Uteslutningar

Substitutioner är i form av en likhet såsom x=3 eller y=sin(x). För bästa effektivitet bör den vänstra sidan vara en enkel variabel. *Uttr* | *Variabel* = *värde* ersätter *värde* vid varje förekomst av *Variabel* i *Uttr*.

$x^3-2\cdot x+7\rightarrow f(x)$	*Done*
$f(x)\|x=\sqrt{3}$	8.73205

Intervallbegränsningar tar formen av en eller flera olikheter som förbinds med logiska "**and**" eller "**or**"-operatorer. Intervallbegränsningar medger också förenklingar som annars kan vara ogiltiga eller ej beräkningsbara.

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x)$	0.
$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x)\|x>0 \text{ and } x<5$	3.



Uteslutningar använder jämförelseoperatorn "inte lika med" (/= eller ≠) för att utesluta ett specifikt värde från övervägning.

→ (lagra)

ctrl var tangent

Value → *Var*

List → *Var*

Matrix → *Var*

Expr → *Function*(*Param1*,...)

List → *Function*(*Param1*,...)

Matrix → *Function*(*Param1*,...)

Om variabeln *Var* inte finns så skapas den och initialiseras till *Value*, *List* eller *Matrix*.

Om variabeln *Var* redan finns, och inte är låst eller skyddad, ersätts dess innehåll med *Value*, *List* eller *Matrix*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **=:** som kortkommando. Skriv exempelvis **`pi/4 =: minvar`**.

$\frac{\pi}{4} \to myvar$	0.785398
$2\cdot\cos(x) \to y1(x)$	*Done*
$\{1,2,3,4\} \to lst5$	$\{1,2,3,4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \to matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → *str1*	"Hello"

:= (tilldela)

ctrl tangenter

Var := *Value*

Var := *List*

Var := *Matrix*

Function(*Param1*,...) := *Expr*

Function(*Param1*,...) := *List*

Function(*Param1*,...) := *Matrix*

Om variabeln *Var* inte finns så skapas *Var* och initialiseras till *Value*, *List* eller *Matrix*.

Om *Var* redan finns, och inte är låst eller skyddad, ersätts dess innehåll med *Value*, *List* eller *Matrix*.

$myvar := \frac{\pi}{4}$	.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"

© (kommentar)

ctrl 📖 tangenter

© [*text*]

© hanterar *text* som en kommentarsrad så att du kan kommentera funktioner och program som du skapar.

© kan vara i början eller var som helst på raden. Allting till höger om ©, till slutet av raden, utgör kommentaren.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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 tangenter, 0 H tangenter

0b *binaryNumber*

0h *hexadecimalNumber*

Betecknar ett binärt respektive ett hexadecimalt tal. För att skriva in ett binärt eller hexadecimalt tal måste du använda prefixet 0b eller 0h oavsett det inställda basläget. Utan prefix behandlas ett tal som ett decimalt tal (bas 10).

Resultaten visas enligt det inställda basläget.

I decimalt basläge:

```
0b10+0hF+10                     27
```

I binärt basläge:

```
0b10+0hF+10                     0b11011
```

I hexadecimalt basläge:

```
0b10+0hF+10                     0h1B
```

Tomma element

Vid analys av verkliga data kanske du inte alltid har en komplett datauppsättning. TI-Nspire™ tillåter tomma dataelement så att du kan fortsätta med de nästan kompletta uppgifterna i stället för att behöva börja om från början eller kassera ofullständiga fall.

Du finner ett exempel på data som inbegriper tomma element i kapitlet Listor och kalkylblad under "*Plotta kalkylbladsdata*."

Med funktionen delVoid() kan du ta bort tomma element från en lista. Med funktionen isVoid() kan du testa förekomst av tomma element. För mer information, se delVoid(), på sidan 40 och isVoid(), på sidan 76.

Obs: För att manuellt mata in ett tomt element i ett matematiskt uttryck, skriv in "_" eller nyckelordet **`tomrum`**. Nyckelordet **`tomrum`** omvandlas automatiskt till symbolen "_" när uttrycket utvärderas. För att skriva in "_" på handenheten, tryck på [ctrl] [␣].

Beräkningar som innehåller tomma element

Flertalet beräkningar som inbegriper en tom inmatning producerar ett tomt resultat. Se specialfall nedan.

$\lvert_\rvert$	_
gcd(100,_)	_
3+_	_
{5,_,10}−{3,6,9}	{2,_,1}

Lista argument som innehåller tomma element

Följande funktioner och kommandon ignorerar (hoppar över) tomma element som påträffas i listargument:

count, **countIf**, **cumulativeSum**, **freqTable▸lista**, **frekvens**, **max**, **medelvärde**, **median**, **produkt**, **stDevPop**, **stDevSamp**, **summa**, **sumIf**, **varPop** och **varSamp** samt regressionsberäkningar, **OneVar**, **TwoVar** och **FiveNumSummary** statistik, konfidensintervaller och stat-tester

sum({2,_,3,5,6.6})	16.6
median({1,2,_,_,_,3})	2
cumulativeSum({1,2,_,4,5})	{1,3,_,7,12}
cumulativeSum([1 2; 3 _; 5 6])	[1 2; 4 _; 9 8]

SortA och **SortD** flyttar alla tomma element inom det första argumentet till botten.

{5,4,3,_,1} → *list1*	{5,4,3,_,1}
{5,4,3,2,1} → *list2*	{5,4,3,2,1}
SortA *list1*,*list2*	*Done*
list1	{1,3,4,5,_}
list2	{1,3,4,5,2}

Lista argument som innehåller tomma element

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD *list1*,*list2*	*Done*
list1	$\{5,3,2,1,_\}$
list2	$\{5,3,2,1,4\}$

I regressionsberäkningar introducerar ett tomrum i en X- eller Y-lista ett tomrum i motsvarande element för residualen.

$l1:=\{1,2,3,4,5\}$: $l2:=\{2,_,3,5,6.6\}$	$\{2,_,3,5,6.6\}$
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	$\{1.,_,1.,1.,1.\}$

En utelämnad kategori i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
$cat:=\{"M","M","F","F"\}$: $incl:=\{"F"\}$	$\{"F"\}$
LinRegMx *l1*,*l2*,1,*cat*,*incl*	*Done*
stat.Resid	$\{_,_,0.,0.\}$
stat.XReg	$\{_,_,4.,5.\}$
stat.YReg	$\{_,_,5.,6.6\}$
stat.FreqReg	$\{_,_,1.,1.\}$

En frekvens på 0 i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
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	$\{1.,_,1.,1.\}$

Kortkommandon för att mata in matematiska uttryck

Med kortkommandon kan du mata in element i matematiska uttryck genom att skriva i stället för att använda Katalogen eller Symbolpaletten. För att exempelvis mata in uttrycket $\sqrt{6}$ kan du skriva **sqrt(6)** på inmatningsraden. När du trycker på enter ändras uttrycket **sqrt(6)** till $\sqrt{6}$. Vissa kortkommandon kan användas från både handenheten och datorns tangentbord. Andra är i första hand användbara från datorns tangentbord.

Från handenheten eller datorns tangentbord

För att mata in detta:	**Skriv detta kortkommando:**
π	pi
θ	theta
∞	**infinity**
$\leq$	**<=**
$\geq$	**>=**
$\neq$	**/=**
$\Rightarrow$ (logisk implikation)	=>
$\Leftrightarrow$ (logisk dubbel implikation, XNOR)	<=>
$\rightarrow$ (lagra operator)	**=:**
| | (absolutbelopp)	**abs(...)**
$\sqrt{}$**()**	**sqrt(...)**
Σ() (mallen Summa)	**sumSeq(...)**
Π() (mallen Produkt)	**prodSeq(...)**
sin^{-1}(), cos^{-1}(), ...	**arcsin(...)**, **arccos(...)**, ...
Δ**List()**	**deltaList(...)**

Från datorns tangentbord

För att mata in detta:	**Skriv detta kortkommando:**
i (imaginära enheten)	**@i**
e (naturlig logaritmbas e)	**@e**
E (grundpotensform)	**@E**
^{T} (transponera)	**@t**
^{r} (radianer)	**@r**

För att mata in detta:	**Skriv detta kortkommando:**
° (grader)	`@d`
g (nygrader)	`@g`
∠ (vinkel)	`@<`
▸ (omvandling)	`@>`
▸Decimal, **▸approxFraction ()**, och så vidare.	`@>Decimal`, `@>approxFraction()`, och så vidare.

EOS™-hierarki (Equation Operating System)

Detta avsnitt beskriver det ekvationsoperativsystem (EOS™) som används av TI-Nspire™ Inlärningsteknologi för matematik och naturvetenskap. Tal, variabler och funktioner matas in i en enkel och direkt sekvens. Programvaran EOS™ beräknar uttryck och ekvationer genom parentesgruppering och enligt de prioriteringsregler som beskrivs nedan.

Ordning vid utvärdering

Nivå	Operator
1	Parenteser ( ), hakparenteser [ ], klammerparenteser { }
2	Indirection (#)
3	Funktionsanrop
4	Postoperatorer: grader-minuter-sekunder (°,',"), fakultet (!), procent (%), radian (^{r}), index ([ ]), transponering (^{T})
5	Exponentberäkning, potensoperator (^)
6	Negation ($^-$)
7	Strängsammanlänkning (&)
8	Multiplikation (•), division (/)
9	Addition (+), subtraktion (-)
10	Likhetsförhållanden: lika med (=), inte lika med (≠ eller /=), mindre än (<), mindre än eller lika med (≤ eller <=), större än (>), större än eller lika med (≥ eller >=)
11	Logiskt **not**
12	Logiskt **and**
13	Logisk **or**
14	**xor**, **nor**, **nand**
15	Logisk implikation (⇒)
16	Logisk dubbel implikation, XNOR⇔
17	(“|”)-operatorn begränsning
18	Spara (→)

Parenteser, hakparenteser och klammerparenteser

Alla beräkningar inom ett par av parenteser, hakparenteser eller klammerparenteser utvärderas först. I exempelvis uttrycket 4(1+2) beräknar EOS™ först den del av uttrycket som är inom parentesen, 1+2, och multiplicerar sedan resultatet, 3, med 4.

Antalet inledande respektive avslutande parenteser, hakparenteser och klammerparenteser måste vara lika inom ett uttryck eller en ekvation. Annars visas ett felmeddelande som anger det element som saknas. Till exempel visar (1+2)/(3+4 felmeddelandet “Missing)”.

Obs: Eftersom TI-Nspire™ programvara ger dig möjlighet att definiera dina egna funktioner betraktas ett variabelnamn, följt av ett uttryck inom parentes, som ett "funktionsanrop" i stället för en indirekt multiplikation. Till exempel är a(b+c) funktionen a utvärderad med b+c. För att multiplicera uttrycket b+c med variabeln a, använd explicit multiplikation: a∗(b+c).

Indirection

Den indirekta operatorn (#) konverterar en sträng till ett variabel- eller funktionsnamn. Till exempel skapar #(“x”&”y”&”z”) variabelnamnet xyz. "Indirection" ger också möjlighet att skapa och modifiera variabler inne i ett program. Om exempelvis 10→r och “r”→s1 erhålls #s1=10.

Postoperatorer

Postoperatorer är operatorer som kommer direkt efter ett argument, t.ex. 5!, 25% eller 60°15' 45". Argument som följs av en postoperator utvärderas på den fjärde prioritetsnivån. I exempelvis uttrycket 4^3! utvärderas 3! först. Resultatet, 6, blir sedan exponenten för 4, vilket ger 4096.

Exponentberäkning

Exponentberäkning (^) och exponentberäkning element för element (.^) utvärderas från höger till vänster. Till exempel utvärderas uttrycket 2^3^2 på samma sätt som 2^(3^2). Man får resultatet 512. Detta är inte detsamma som (2^3)^2, vilket ger resultatet 64.

Negation

För att skriva in ett negativt tal, tryck på (−) följt av talet. Postoperationer och exponentberäkningar utförs före negationer. Till exempel är resultatet av ⁻x2 ett negativt tal och $^{-}9^2 = {}^{-}81$. Använd parenteser för att beräkna kvadraten på ett negativt tal. Till exempel ger $(^{-}9)^2$ resultatet 81.

Begränsning (“|”)

Argumentet som följer efter (“|”)-operatorn begränsning ger en uppsättning av begränsningar som påverkar utvärderingen av argumentet som föregår operatorn.

Konstanter och värden

Följande tabell listar konstanterna och deras värden som finns tillgängliga när enhetsomvandling utförs. De kan skrivas in manuellt eller väljas från listan **Konstanter** i **Verktyg > Enhetsomvandlingar** (Handenhet: Tryck 📖 **3**).

Konstant	Namn	Värde
_c	Ljusets hastighet	299792458 _m/_s
_Cc	Coulombs konstant	8987551787.3682 _m/_F
_Fc	Faradays konstant	96485.33289 _coul/_mol
_g	Tyngdkraftens acceleration	9.80665 _m/_s^2
_Gc	Gravitationskonstanten	6.67408E-11 _m^3/_kg/_s^2
_h	Plancks konstant	6.626070040E-34 _J _s
_k	Boltzmanns konstant	1.38064852E-23 _J/_°K
_μ0	Permeabilitet vid vakuum	1.2566370614359E-6 _N/_A^2
_μb	Bohrmagneton	9.274009994E-24 _J _m^2/_Wb
_Me	Elektronens vilomassa	9.10938356E-31 _kg
_Mμ	Myonmassa	1.883531594E-28 _kg
_Mn	Neutronens vilomassa	1.674927471E-27 _kg
_Mp	Protonens vilomassa	1.672621898E-27 _kg
_Na	Avogadros tal	6.022140857E23 /_mol
_q	Elektronens laddning	1.6021766208E-19 _coul
_Rb	Bohrradie	5.2917721067E-11 _m
_Rc	Molar gas constant (Allmänna gaskonstanten)	8.3144598 _J/_mol/_°K
_Rdb	Rydbergs konstant	10973731.568508/_m
_Re	Elektronradie	2.8179403227E-15 _m
_u	Atommassa	1.660539040E-27 _kg
_Vm	Molvolym	2.2413962E-2 _m^3/_mol
_ε0	Permittivitet vid vakuum	8.8541878176204E-12 _F/_m
_σ	Stefan-Boltzmanns konstant	5.670367E-8 _W/_m^2/_$°K^4$
_ϕ0	Magnetiskt flödeskvantum	2.067833831E-15 _Wb

Felkoder och meddelanden

När ett fel inträffar placeras dess felkod i variabeln *errCode*. Användardefinierade program och funktioner kan undersöka *errCode* för att bestämma orsaken till ett fel. För ett exempel på användningen av *errCode*, se exempel 2 under kommandot **Try**, på sidan 160.

Obs: Vissa feltillstånd avser endast TI-Nspire™ CAS-produkter medan andra endast avser TI-Nspire™-produkter.

Felkod	Beskrivning
10	A function did not return a value (En funktion gav inget värde)
20	A test did not resolve to TRUE or FALSE. (Ett test gav varken TRUE eller FALSE.) I regel kan odefinierade variabler inte jämföras. Som exempel orsakar testet “Om a<b” detta fel om a eller b är odefinierade när Om-påståendet exekveras.
30	Argument cannot be a folder name. (Argumentet får inte vara ett mappnamn.)
40	Argument error (Argumentfel)
50	Argument mismatch (Fel typ av argument) Två eller flera argument måste vara av samma typ.
60	Argument must be a Boolean expression or integer (Argumentet måste vara ett booleskt uttryck eller ett heltal)
70	Argument must be a decimal number (Argumentet måste vara ett decimaltal)
90	Argument must be a list (Argumentet måste vara en lista)
100	Argument must be a matrix (Argumentet måste vara en matris)
130	Argument must be a string (Argumentet måste vara en sträng)
140	Argument must be a variable name (Argumentet måste vara ett variabelnamn). Kontrollera att namnet: • inte börjar med en siffra • inte innehåller mellanslag eller specialtecken • inte innehåller understrykningstecken eller punkt på ogiltigt sätt • inte överskrider max. längd Se avsnittet Calculator (Räknare) i dokumentationen för mer information.
160	Argument must be an expression (Argumentet måste vara ett uttryck)
165	Batteries too low for sending or receiving (Batterierna är för svaga för sändning eller mottagning) Sätt i nya batterier före sändning eller mottagning.
170	Bound (Gräns)

Felkod	Beskrivning
	Den nedre gränsen måste vara mindre än den övre gränsen för att definiera sökintervallet.
180	Avsluta Tangenten [esc] eller [on] trycktes ned under en lång beräkning eller under programexekvering.
190	Circular definition (Cirkulär definition) Detta meddelande visas för att inte minnet skall ta slut under ändlös ersättning av variabelvärden i samband med förenkling. Till exempel orsakar a+1->a, där a är en odefinierad variabel, detta fel.
200	Invalid constraint (Ogiltig begränsning) Som exempel skulle solve(3x^2-4=0,x) | x<0 eller x>5 ge detta felmeddelande eftersom begränsningen är separerad av “eller” i stället för “och”.
210	Invalid data type (Ogiltig datatyp) Ett argument är av fel datatyp.
220	Dependent limit (Beroende gräns)
230	Dimension Ett index för en lista eller för en matris är inte giltigt. Om exempelvis listan {1,2,3,4} lagras i L1 är L1[5] ett dimensionsfel eftersom L1 endast innehåller fyra element.
235	Dimension mismatch. (Dimensioner överensstämmer inte.) Inte tillräckligt med element i listorna.
240	Dimension mismatch (Dimensioner överensstämmer inte) Två eller flera argument måste ha samma dimension. Till exempel är [1,2]+[1,2,3] ett dimensionsfel eftersom matriserna innehåller olika antal element.
250	Divide by zero (Division med noll)
260	Domain error (Områdesfel) Ett argument måste vara inom ett specificerat område. Exempelvis är rand(0) inte giltigt.
270	Duplicate variable name (Dubblerade variabelnamn)
280	Else and ElseIf invalid outside of If…EndIf block (Else och ElseIf är ogiltiga utanför If..EndIf-block)
290	EndTry is missing the matching Else statement (EndTry saknas i det matchande Else-påståendet)
295	Excessive iteration (För många iterationer)

Felkod	Beskrivning
300	Expected 2 or 3-element list or matrix (En 2- eller 3-elements lista eller matris förväntades)
310	Första argumentet till nSolve måste vara en ekvation i en variabel. Det får inte innehålla någon variabel utan värde förutom den variabel som används i beräkningen.
320	First argument of solve or cSolve must be an equation or inequality (Första argumentet till solve eller cSolve måste vara en ekvation eller en olikhet) Till exempel är solve(3x^2-4,x) ogiltigt eftersom första argumentet inte är en ekvation.
345	Inconsistent units (Enheterna är oförenliga)
350	Index out of range (Index ligger utanför giltigt område)
360	Indirection string is not a valid variable name (Indirection-strängen är inte ett giltigt variabelnamn)
380	Undefined Ans (Odefinierad Ans) Antingen skapades inte Ans av den föregående beräkningen eller också har ingen tidigare beräkning matats in.
390	Invalid assignment (Ogiltig tilldelning)
400	Invalid assignment value (Ogiltigt tilldelningsvärde)
410	Invalid command (Ogiltigt kommando)
430	Invalid for the current mode settings (Ogiltigt för de aktuella inställningarna)
435	Invalid guess (Ogiltig gissning)
440	Invalid implied multiply (Ogiltig indirekt multiplikation) Till exempel är x(x+1) ogiltig medan x*(x+1) har korrekt syntax. Detta är för att undvika förväxling mellan indirekt multiplikation och funktionsanrop.
450	Invalid in a function or current expression (Ogiltigt i en funktion eller i det aktuella uttrycket) Endast vissa kommandon är giltiga i en användardefinierad funktion.
490	Invalid in Try..EndTry block (Ogiltigt i Try..EndTry-block)
510	Invalid list or matrix (Ogiltig lista eller matris)
550	Invalid outside function or program (Ogiltigt utanför funktion eller program) Ett antal kommandon är inte giltiga utanför en funktion eller ett program. Exempelvis kan Local bara användas inne i en funktion eller ett program.
560	Invalid outside Loop..EndLoop, For..EndFor, or While..EndWhile blocks (Ogiltigt utanför Loop..EndLoop, For..EndFor, eller While..EndWhile-block) Exempelvis kan kommandot Exit bara användas innanför dessa slingor.

Felkod	Beskrivning
565	Invalid outside program (Ogiltig utanför program)
570	Invalid pathname (Ogiltigt sökvägsnamn) Till exempel är \var ogiltigt.
575	Invalid polar complex (Ogiltigt polärt komplext tal)
580	Invalid program reference (Ogiltig programreferens) Man får inte referera till Program inom funktioner eller uttryck såsom 1+p(x) där p är ett program.
600	Invalid table (Ogiltig tabell)
605	Invalid use of units (Ogiltig användning av enheter)
610	Invalid variable name in a Local statement (Ogiltigt variabelnamn i ett Local-påstående)
620	Invalid variable or function name (Ogiltigt variabel- eller funktionsnamn)
630	Invalid variable reference (Ogiltig variabelreferens)
640	Invalid vector syntax (Ogiltig syntax för vektor)
650	Link transmission (Länköverföring) En överföring mellan två enheter slutfördes inte. Kontrollera att anslutningskabeln är ordentligt ansluten i båda ändarna.
665	Matrix not diagonalizable (Matrisen är inte diagonaliserbar)
670	Low Memory (Ont om minne) 1. Ta bort lite data från detta dokument 2. Spara och stäng detta dokument Om dessa steg inte löser problemet, plocka ur batterierna och sätt i dem igen
672	Resource exhaustion (Resursutmattning)
673	Resource exhaustion (Resursutmattning)
680	Missing ((Saknar “ (“)
690	Missing) (Saknar “) “)
700	Missing " (Saknar “ " “)
710	Missing] (Saknar “] “)
720	Missing } (Saknar “ } “)
730	Missing start or end of block syntax (Saknar start eller slut i blockets syntax)

Felkod	Beskrivning
740	Missing Then in the If..EndIf block (Saknar “Then” i If..EndIf-blocket)
750	Name is not a function or program (Namnet är inte en funktion eller ett program)
765	No functions selected (Inga funktioner är valda)
780	No solution found (Ingen lösning funnen)
800	Non-real result (Resultatet är inte ett reellt tal) Om exempelvis programvaran har inställningen Real så är $\sqrt{(-1)}$ ogiltigt. För att medge komplexa resultat, ändra lägesinställningen “Real or Complex” till RECTANGULAR eller POLAR.
830	Overflow (För stort värde)
850	Program not found (Program hittades inte) En programreferens inne i ett annat program kunde ej hittas via angiven sökväg under exekveringen.
855	Rand type functions not allowed in graphing (Funktioner av slumptyp tillåts ej vid grafritning)
860	Recursion too deep (Rekursionen för djup)
870	Reserved name or system variable (Reserverat namn eller systemvariabel)
900	Argument error (Argumentfel) Median-median-modellen kunde inte användas på datauppsättningen.
910	Syntaxfel
920	Text not found (Text hittades inte)
930	Too few arguments (För få argument) Funktionen eller kommandot saknar ett eller flera argument.
940	Too many arguments (För många argument) Uttrycket eller ekvationen innehåller för många argument och kan inte utvärderas.
950	Too many subscripts (För många nedsänkta tecken)
955	Too many undefined variables (För många odefinierade variabler)
960	Variable is not defined (Variabel ej definierad) Inget värde är tillskrivet variabeln. Använd något av följande kommandon: • sto → • := • **Define (Definiera)**

Felkod	Beskrivning
	för att tilldela variabler värden.
965	Unlicensed OS (Olicensierat OS)
970	Variable in use so references or changes are not allowed (Variabeln används varför referenser eller ändringar ej tillåts)
980	Variable is protected (Variabeln är skyddad)
990	Invalid variable name (Ogiltigt variabelnamn) Kontrollera att namnet inte överskrider max. längd.
1000	Window variables domain (Fönstervariabeldomän)
1010	Zoom
1020	Internal error (Internt fel)
1030	Protected memory violation (Minnesskydd)
1040	Unsupported function. (Funktionen stöds ej.) Denna funktion kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1045	Unsupported operator. (Operatorn stöds ej.) Denna operator kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1050	Unsupported feature. (Funktionen stöds ej.) Denna operator kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1060	Input argument must be numeric. (Inmatade argument måste vara numeriska.) Endast inmatningar som innehåller numeriska värden är tillåtna.
1070	Trig function argument too big for accurate reduction (Trig-funktionens argument är för stort för en noggrann reduktion)
1080	Unsupported use of Ans. This application does not support Ans. (Användning av Ans stöds inte. Denna applikation stöder inte Ans.)
1090	Function is not defined. (Funktion ej definierad.) Använd något av följande kommandon: • **Define (Definiera)** • **:=** • sto → för att definiera en funktion.
1100	Non-real calculation (Icke-reell beräkning) Om exempelvis programvaran har inställningen Real så är $\sqrt{}(-1)$ ogiltigt. För att medge komplexa resultat, ändra lägesinställningen “Real or Complex” till RECTANGULAR eller POLAR.
1110	Invalid bounds (Ogiltiga gränser)

Felkod	Beskrivning
1120	No sign change (Ingen teckenändring)
1130	Argument cannot be a list or matrix (Argumentet får inte vara en lista eller en matris)
1140	Argument error (Argumentfel) Det första argumentet måste vara ett polynomuttryck i det andra argumentet. Om det andra argumentet utelämnas försöker programvaran välja ett förinställt argument.
1150	Argument error (Argumentfel) De första två argumenten måste vara polynomuttryck i det tredje argumentet. Om det tredje argumentet utelämnas försöker programvaran välja ett förinställt argument.
1160	Invalid library pathname (Ogiltigt sökvägsnamn för bibliotek) Ett sökvägsnamn måste vara i formen *xxx**yyy*, där: • Delen *xxx* kan ha 1 till 16 tecken. • Delen *yyy* kan ha 1 till 15 tecken. Se avsnittet Bibliotek i dokumentationen för mer information.
1170	Invalid use of library pathname (Ogiltig användning av sökvägsnamn för bibliotek) • Ett sökvägsnamn får inte ges ett värde med **Define**, **:=** eller sto **→**. • Ett sökvägsnamn får inte anges som en Local-variabel eller användas som en parameter i en funktions- eller programdefinition.
1180	Invalid library variable name. (Ogiltigt namn på biblioteksvariabel.) Kontrollera att namnet: • inte innehåller en punkt • inte börjar med ett understrykningstecken • inte överskrider 15 tecken Se avsnittet Bibliotek i dokumentationen för mer information.
1190	Biblioteksdokument kunde ej hittas: • Kontrollera att biblioteket är i MyLib-mappen. • Uppdatera bibliotek. Se avsnittet Bibliotek i dokumentationen för mer information.
1200	Biblioteksvariabel kunde ej hittas: • Kontrollera att biblioteksvariabeln finns i det första problemet i biblioteket. • Kontrollera att biblioteksvariabeln har definierats som LibPub eller LibPriv. • Uppdatera bibliotek. Se avsnittet Bibliotek i dokumentationen för mer information.

Felkod	Beskrivning
1210	Ogiltigt genvägsnamn till bibliotek. Kontrollera att namnet inte: • innehåller en punkt • börjar med ett understrykningstecken • överskrider 16 tecken • är ett reserverat namn Se avsnittet Bibliotek i dokumentationen för mer information.
1220	Områdesfel: Funktionerna tangentLine och normalLine stöder endast funktioner med reella värden.
1230	Områdesfel. Trigonometriska omvandlingsoperatorer stöds inte i vinkellägena Grader och Nygrader.
1250	Argumentfel Använd ett linjärt ekvationssystem. Exempel på ett linjärt ekvationssystem med variablerna x och y: 3x+7y=5 2y-5x=-1
1260	Argumentfel: Det första argumentet till nfMin eller nfMax måste vara ett uttryck i en variabel. Det får inte innehålla någon variabel utan värde förutom den variabel som används i beräkningen.
1270	Argumentfel Derivatans ordning måste vara lika med 1 eller 2.
1280	Argumentfel Använd ett polynom i expanderad form i en variabel.
1290	Argumentfel Använd ett polynom i en variabel.
1300	Argumentfel Koefficienterna i polynomet måste beräknas till numeriska värden.
1310	Argumentfel: En funktion kunde inte utvärderas för ett eller flera av dess argument.
1380	Argumentfel:

Felkod	Beskrivning
	Nästlade anrop till funktionen domän() är inte tillåtna.

Varningskoder och meddelanden

Du kan använda funktionen **warnCodes()** för att lagra de varningskoder som genereras genom att utvärdera ett uttryck. Denna tabell listar varje numerisk varningskod och tillhörande meddelande.

För ett exempel på lagring av varningskoder, se **warnCodes()**, på sidan 168.

Varningskod	Meddelande
10000	Operationen kan ge falska lösningar.
10001	Derivering av en ekvation kan ge en falsk ekvation.
10002	Tvivelaktig lösning
10003	Tvivelaktig noggrannhet
10004	Operationen kan förlora lösningar.
10005	cSolve kan ange fler nollställen.
10006	Solve kan ange fler nollställen.
10007	Flera lösningar kan finnas. Försök att ange lämpliga nedre och övre gränser och/eller en gissning. Exempel som använder solve(): • solve(Ekvation, Var=Gissning)|undrGräns<Var<övrGräns • solve(Ekvation, Var)|undrGräns<Var<övrGräns • solve(Ekvation, Var=Gissning)
10008	Resultatets definitionsområde kan vara mindre än inmatningens definitionsområde.
10009	Resultatets definitionsområde kan vara större än inmatningens definitionsområde.
10012	Icke-reell beräkning
10013	∞^0 eller undef^0 ersattes med 1
10014	undef^0 ersattes med 1
10015	1^∞ eller 1^undef ersattes med 1
10016	1^undef ersattes med 1
10017	För stort värde ersattes med ∞ eller ⁻∞
10018	Operationen kräver och ger ett 64-bitars värde.
10019	Resursutmattning, förenklingen kan vara ofullständig.
10020	Trig-funktionens argument är för stort för en noggrann reduktion.
10021	Inmatningen innehåller en odefinierad parameter. Resultatet kanske inte är giltigt för samtliga möjliga parametervärden.

Varningskod	Meddelande
10022	Att ange lämpliga övre och undre gränser kan ge en lösning.
10023	Skalär har multiplicerats med enhetsmatrisen.
10024	Resultat erhållet med ungefärlig aritmetik.
10025	Ekvivalens kan inte verifieras i EXAKT läge.
10026	Begränsning kan ignoreras. Specificera begränsning med formen "\" 'Variable MathTestSymbol Constant' eller en sammansättning av dessa former, till exempel 'x<3 och x>-12'

Allmän information

Hjälp-funktion online

education.ti.com/eguide

Välj ditt land för ytterligare produktinformation.

Kontakta TI support

education.ti.com/ti-cares

Välj ditt land för teknisk och andra supportresurser.

Service- och garanti-information

education.ti.com/warranty

Välj ditt land för information om längden och villkoren för garanti.

Begränsad garanti. Denna garanti påverkar inte dina lagstadgade rättigheter.

Innehållsförteckning

'

', minutnotation 194

–

–, subtrahera[*] 177

!

!, fakultet 187

"

", sekundnotation 194

#

#, indirection 192
#, indirection, operator 205

%

%, procent 183

&

&, lägg till 187

*

*, multiplicera 178

.

.-, punkt subtraktion 181
.*, punkt multiplikation 182
./, punkt division 182
.^, punkt potens 182
.+, punkt addition 181

:

:=, tilldela 198

^

$^{-1}$, inverterat värde 196
^, potens 180

|

|, operatorn begränsning 196

+

+, addera 177

/

/, dividera[*] 179

≠

≠, inte lika med[*] 184

=

=, lika med 183

>

>, större än 185

∏

∏, produkt[*] 189

∑

∑( ), summa[*] 190
∑Int( ) 191
∑Prn( ) 191

√

√, kvadratrot[*] 189

∫

∫, integral[*] 189

≤

≤, mindre än eller lika med 185

≥

≥, större än eller lika med 186

▸

▸, konvertera till nygradvinkel[Grad] 69
▸approxFraction( ) 13
▸Base10, visa som decimalt heltal[Bas 10] 17
▸Base16, visa som hexadecimal[Bas 16] 18
▸Base2, visa som binär[Bas 2] 16
▸Cylind, visa som cylindrisk vektor [Cylind] 35
▸DD, visa som decimal vinkel[DD] 36
▸Decimal, visa resultat som decimal [Decimal] 36
▸DMS, visa som grad/minut/sekund [DMS] 43
▸Polar, visa som polär vektor[Polär] 113
▸Rad, omvandla till radianer 120
▸Rect, visa som ortogonal vektor 123
▸Sphere, visa som sfärisk vektor[sfär] 147

→

→, lagra 198

⇒

⇒ , logisk implikation[*] 186, 202

⇔

⇔ , logisk dubbel implikation[*] 187

©

©, kommentar 199

°

°, grader/minuter/sekunder[*] 194
°, gradnotation[*] 194

0

0b, binär indikator 199
0h, hexadecimal indikator 199

1

10^( ), tiopotens 196

2

2-sampel F Test 58

A

abs( ), absolutbelopp 7
absolutbelopp
mall 3-4
addera, + 177
amorteringstabell, amortTbl( ) 7, 15
amortTbl( ), amorteringstabell 7, 15
and, Booleska operatorer 8
andraderivata
mall för 6
angle( ), vinkel 9
annars, Else 69
ANOVA 2-vägs, 2-vägs variansanalys 10
ANOVA, 1-vägs variansanalys 9
Ans, sista svar 12
antal dagar mellan datum, dbd( ) 35
approx( ), ungefärlig 12
approxRational( ) 13
arccos() 13
arccosh() 13
arccosinus, $\cos^{-1}$( ) 26
arccot() 13
arccoth() 13
arccsc() 13
arccsch() 14
arcsec() 14
arcsech() 14
arcsin() 14
arcsinh() 14
arcsinus, $\sin^{-1}$( ) 143
arctan() 14
arctangens, $\tan^{-1}$( ) 154
arctanh() 14
argument i TVM-funktioner 163
augment( ), sammanfoga/slå ihop 14
avgRC( ), genomsnittlig
ändringsfrekvens 15

avrunda, round( ) 132
avsluta
om, EndIf 69
avsluta om, EndIf 69
avsluta, Exit 49

B

Begär 126
beräkning, ordning vid 204
bestämd integral
mall 6
bibliotek
skapa genvägar till objekt 78
binomCdf( ) 18, 74
binomPdf( ) 19
binär
indikator, 0b 199
visa, ▸Base2 16
blandade bråk, använda propFrac(›
med 116
Booleska operatorer
⇒ 186, 202
⇔ 187
and 8
eller 109
icke 105
negerad 99
nor 104
xor 170
bråk
mall 1
propFrac 116

C

Cdf( ) 53
ceiling( ), tak 19
centralDiff( ) 19
char( ), teckensträng 20
$\chi^2$2way 20
χ^2Cdf( ) 21
χ^2GOF 21
χ^2Pdf( ) 22
ClearAZ 22
ClrErr, rensa fel 22
colAugment 23
colDim( ), matriskolumndimension 23
colNorm( ), matrixskolumnnorm 23
conj( ), komplexkonjugat 24
constructMat( ), konstruera matris 24
corrMat( ), korrelationsmatris 25
$\cos^{-1}$, arccosinus 26
cos( ), cosinus 25
$\cosh^{-1}$( ), hyperbolisk arccosinus 28
cosh( ), hyperbolisk cosinus 27
cosinus, cos( ) 25
$\cot^{-1}$( ), arccotangens 29
cot( ), cotangens 28
cotangens, cot( ) 28
$\coth^{-1}$( ), hyperbolisk arccotangens 29
coth( ), hyperbolisk cotangens 29
count( ), räkna poster i en lista 30
countif( ), villkorligt räkna poster i en
lista 30
cPolyRoots() 31
crossP( ), vektorprodukt 31
$\csc^{-1}$( ), invers cosekant 32
csc( ), cosekant 32
csch^{-1}( ), invers hyperbolisk cosekant 33
csch( ), hyperbolisk cosekant 32
CubicReg, kubisk regression 33
cumulativeSum( ), kumulativ summa 34
Cycle, cykel 34
cykel, Cycle 34

D

d( ), förstaderivata 188
days between dates (dagar mellan
datum), dbd( ) 35
dbd( ), days between dates (dagar
mellan datum) 35
decimal
heltalsvisning, ▸Base10 17
vinkelvisning, ▸DD 36
Define 36
Define LibPriv 38
Define LibPub 38

define, Define 36
Define, define 36
defining
private function or program ... 38
public function or program 38
dela heltal, intDiv() 73
delmatris, subMat( ) 153
deltaList() 39
DelVar, ta bort variabel 39
delVoid( ), ta bort tomma element . 40
derivata
förstaderivata, d( ) 188
numerisk derivata, nDeriv( ) ... 102
numerisk derivata, nDerivative( ) 101
det( ), matrisdeterminant 40
diag( ), diagonal matris 40
dim( ), dimension 41
dimension, dim( ) 41
Disp, visa data 41, 135
DispAt 41
dividera, / 179
dotP( ), skalärprodukt 44

E

e exponent
mall 2
e till en potens, e^( ) 44, 50
E, exponent 193
e^( ), e till en potens 44
eff), konvertera nominell till effektiv
ränta 45
effektiv ränta, eff( ) 45
egentligt bråk, propFrac 116
egenvektor, eigVc( ) 45
egenvärde, eigVl( ) 46
eigVc( ), egenvektor 45
eigVl( ), egenvärde 46
Ekvationsoperativsystem (EOS) 204
ekvationssystem (2 ekvationer)
mall 3
ekvationssystem (N ekvationer)
mall 3
eller (boolesk), eller 109
eller, Boolesk operator 109
else if, ElseIf 46
ElseIf, else if 46
end
for, EndFor 55
funktion, EndFunc 58
end function, EndFunc 58
EndTry, slut försök 159
EndWhile, slut medan 169
enhetsvektor, unitV( ) 166
envariabelstatistik, OneVar 108
EOS (Ekvationsoperativsystem) 204
euler( ), Euler function 47
Exit, avsluta 49
exp( ), e till en potens 50
exponent, E 193
exponenter
mall 1
exponentiell regession, ExpReg 51
expr( ), sträng till uttryck 51
ExpReg, exponentiell regession 51

F

factor( ), faktor 52
faktor, factor( ) 52
fakultet, ! 187
fel och felsökning
flytta fel, PassErr 111
rensa fel, ClrErr 22
Fill, fylla matris 53
finansiella funktioner, tvmFV( ) 162
finansiella funktioner, tvmI( ) 162
finansiella funktioner, tvmN( ) 163
finansiella funktioner, tvmPmt( ) ... 163
finansiella funktioner, tvmPV( ) 163
FiveNumSummary 53
floor( ), golv 54
flytta fel, PassErr 111
For 55
for, For 55
For, for 55
format( ), formatsträng 55
formatsträng, format( ) 55
fpart( ), funktionsdel 56

freqTable( ) 56
frequency( ) 57
Frobenius norm, norm( ) 104
Func, funktion 58
Func, programfunktion 58
functions
user-defined 36
funktioner
del, fpart( ) 56
programfunktion, Func 58
funktioner och variabler
kopiera 24
förstaderivata
mall för 5
försök, Try 159

G

g, nygrader 193
gcd( ), största gemensamma delare 59
genomsnittlig ändringsfrekvens, avgRC( ) 15
geomCdf( ) 59
geomPdf( ) 60
Get 60
getDenom( ), hämta/ge nämnare 61
getKey() 61
getLangInfo( ), hämta/ge språkinformation 65
getLockInfo( ), testar låsstatus hos en variabel eller variabelgrupp 65
getMode( ), hämta lägesinställningar 66
getNum( ), hämta/ge tal 67
GetStr 67
getType( ), get type of variable 67
getVarInfo( ), hämta/ge variabelinformation 68
golv, floor( ) 54
Goto, gå till 69
grader/minuter/sekunder 194
gradnotation, ° 194
grupper, låsa och låsa upp 87, 166
grupper, testa låsstatus 65
gå till, Goto 69

H

heltal, int() 72
heltalsdel, iPart( ) 75
hexadecimal
indikator, 0h 199
visa, ▸Base16 18
hyperbolisk
arccosinus, $\cosh^{-1}$( ) 28
arcsinus, $\sinh^{-1}$( ) 144
arctangens, $\tanh^{-1}$( ) 155
cosinus, cosh( ) 27
sinus, sinh( ) 144
tangens, tanh( ) 155
hämta/ge
nämnare, getDenom( ) 61
tal, getNum( ) 67
variabelinformation, getVarInfo( ) 65, 68
höger, right( ) 129
höger, right() 73

I

icke, Booleska operatorer 105
identitetsmatris, identity( ) 69
identity( ), identitetsmatris 69
If, om 69
ifFn( ) 71
imag( ), imaginärdel 71
imaginärdel, imag() 71
indirection, # 192
indirection, operator (#) 205
inom sträng, inString() 72
inString( ), inom sträng 72
inställningar, hämta aktuella 66
int( ), heltal 72
intDiv( ), dela heltal 73
inte lika med, ≠ 184
integral, ∫ 189
interpolate( ), interpolera 73
invers kumulativ normalfördelning, invNorm( ) 75
invers, $^{-1}$ 196
inverterat värde, $^{-1}$ 196

invF( ) 74
invNorm( ), invers kumulativ normalfördelning 75
invt( ) 75
Invχ²( ) 73
iPart( ), heltalsdel 75
irr( ), internränta internal rate of return, irr( ) 75
isPrime(), primtest 76
isVoid( ), testa om sant 76

K

kombinationer, nCr( ) 100
Kommandot Wait 167
kommentar, © 199
komplext konjugat, conj( ) 24
konstruera matris, constructMat( ) 24
konvertera 4Grad 69
kopiera variabel eller funktion, CopyVar 24
korrelationsmatris, corrMat( ) 25
kortkommandon 202
kortkommandon, tangentbord 202
kubisk regression, CubicReg 33
kumulativ summa, cumulativeSum( ) 34
kvadratisk regression, QuadReg 117
kvadratrot mall 1
kvadratrot, √( ) 147, 189
kvartär regression, QuartReg 119

L

lagra symbol, & 198
Lbl, märka 77
lcm, minsta gemensamma multipel 77
left( ), vänster 77
LibPriv 38
LibPub 38
libShortcut( ), skapa genvägar till biblioteksobjekt 78
lika med, = 183
linjär regression, LinRegAx 79
linjär regression, LinRegBx 78, 80
LinRegBx, linjär regression 78
LinRegMx, linjär regression 79
LinRegtIntervals, linjär regression 80
LinRegtTest 82
linSolve() 83
Δlist( ), listskillnad 84
list▸mat( ), lista till matris 84
lista till matris, list▸mat( ) 84
lista, räkna poster 30
lista, villkorligt räkna poster 30
listor
kumulativ summa, cumulativeSum( ) 34
lista till matris, list▸mat( ) 84
matris till lista, mat▸list( ) 91
maximum, max( ) 92
med tomma element 200
minimum, min( ) 95
mittsträng, mid( ) 94
ny, newList( ) 102
produkt, product( ) 116
sammanfoga/slå ihop, augment( ) 14
skalärprodukt, dotP( ) 44
skillnad, @list( ) 84
skillnader i en lista, @list( ) 84
sortera fallande, SortD 147
sortera stigande, SortA 146
summering, sum( ) 152
vektorprodukt, crossP( ) 31
ln( ), naturlig logaritm 84
LnReg, logaritmisk regression 85
Local, lokal variabel 86
Log mall 2
logaritmer 84
logaritmisk regression, LnReg 85
logisk dubbel implikation, ⇔ 187
logisk implikation, ⇒ 186, 202
Logistic, logistisk regression 88
LogisticD, logistisk regression 89
logistisk regression, Logistic 88

logistisk regression, LogisticD 89
lokal variabel, Local 86
lokal, Local 86
Loop, slinga 90
LU, matris undre-övre uppdelning .. 91
låsa upp variabler eller variabelgrupper 166
låsa variabler eller variabelgrupper .. 87
Låsa, låsa variabel eller variabelgrupp 87
lägen
inställning, setMode( ) 138
lägesinställningar, getMode( ) 66
lägg till, & 187
längd på sträng 41

M

mallar
absolutbelopp 3-4
andraderivata 6
bestämd integral 6
bråk 1
e exponent 2
ekvationssystem (2 ekvationer) . 3
ekvationssystem (N ekvationer) 3
exponent 1
förstaderivata 5
kvadratrot 1
Log 2
matris (1 × 2) 4
matris (2 × 1) 4
matris (2 × 2) 4
matris (m × n) 4
n:te rot 1
produkt (P) 5
stegvis funktion (2 steg) 2
stegvis funktion (N steg) 3
summa (G) 5
mat▸list( ), matris till lista 91
matris (1 × 2)
mall 4
matris (2 × 1)
mall 4
matris (2 × 2)
mall 4
matris (m × n)
mall 4
matris till lista, mat▸list( ) 91
matriser
delmatris, subMat( ) 153
determinant, det( ) 40
diagonal, diag( ) 40
dimension, dim( ) 41
egenvektor, eigVc( ) 45
egenvärde, eigVl( ) 46
fylla, Fill 53
identitet, identity( ) 69
kolumndimension, colDim( ) ... 23
kolumnnorm, colNorm( ) 23
kumulativ summa, cumulativeSum( ) 34
lista till matris, list▸mat( ) 84
matris till lista, mat▸list( ) 91
maximum, max( ) 92
minimum, min( ) 95
ny, newMat( ) 102
produkt, product( ) 116
punkt addition, .+ 181
punkt division, .P 182
punkt multiplikation, .* 182
punkt potens, .^ 182
punkt subtration, .N 181
QR-faktorisering, QR 117
radaddition, rowAdd( ) 132
raddimension, rowDim( ) 133
radmultiplikation och addition, mRowAdd( ) 97
radnorm rowNorm( ) 133
radoperation, mRow( ) 96
reducerad radtrappstegsform, rref( ) 133
sammanfoga/slå ihop, augment( ) 14
slump, randMat( ) 122
summering, sum( ) 152
transponera, T 153
trappstegsform, ref( ) 124
undermatris, subMat( ) 151
undre-övre uppdelning, LU 91

växla matrisrad, rowSwap( ) ... 133
max( ), maximum ... 92
maximum, max( ) ... 92
mean( ), medelvärde ... 92
med, ... 196
medan, While ... 169
medelvärde, mean( ) ... 92
median( ), median ... 93
median, median( ) ... 93
medium-medium-linjeregression, MedMed ... 93
MedMed, medium-medium-linjeregression ... 93
mid( ), mittsträng ... 94
min( ), minimum ... 95
mindre än eller lika med, { ... 185
minimum, min( ) ... 95
minsta gemensamma multipel, lcm . 77
minutnotation, ... 194
mirr( ), modifierad internränta ... 95
mittsträng, mid( ) ... 94
mod( ), modul ... 96
modifierad internränta, mirr( ) ... 95
modul, mod( ) ... 96
mRow( ), matrisradoperation ... 96
mRowAdd( ), matrisradmultiplikation och addition ... 97
Multipelt linjärt regression-t-test ... 98
multiplicera, * ... 178
MultReg ... 97
MultRegIntervals( ) ... 97
MultRegTests( ) ... 98
märka, Lbl ... 77

N

n:te rot
mall ... 1
nand, Boolesk operator ... 99
naturlig logaritm, ln( ) ... 84
nCr( ), kombinationer ... 100
nDerivative( ), numerisk derivata ... 101
negation, skriva in negativa tal ... 205
nettovärde, npv( ) ... 106
newList( ), ny lista ... 102
newMat( ), ny matris ... 102
nfMax( ), numeriskt funktionsmaximum ... 102
nfMin( ), numeriskt funktionsminimum ... 102
nInt( ), numerisk integral ... 103
nom), konvertera effektiv till nominell ränta ... 103
nominell ränta, nom( ) ... 103
nor, Booleska operatorer ... 104
norm( ), Frobenius norm ... 104
normCdf( ) ... 105
normPdf( ) ... 105
nPr( ), permutationer ... 106
npv( ), nettovärde ... 106
nSolve( ), numerisk lösning ... 107
numeric
derivata, nDeriv( ) ... 102
numerisk
derivata, nDeriv( ) ... 102
derivata, nDerivative( ) ... 101
integral, nInt( ) ... 103
lösning, nSolve( ) ... 107
ny
lista, newList( ) ... 102
matris, newMat( ) ... 102
nygradsnotation, g ... 193
när, when( ) ... 169

O

objekt
skapa genvägar till bibliotek ... 78
om, If ... 69
omvandla
▸Rad ... 120
OneVar, envariabelstatistik ... 108
operatorer
ordning vid beräkning ... 204
operatorn begränsning "|" ... 196
operatorn begränsning, ordning vid utvärdering ... 204
ord( ), numerisk teckenkod ... 110
ortogonalvektorvisning, ▸Rect ... 123

fördelningsfunktioner
binomCdf( ) 18, 74
binomPdf( ) 19
χ²2way( ) 20
invNorm( ) 75
invt( ) 75
Invχ²( ) 73
normCdf( ) 105
normPdf( ) 105
χ²Cdf( ) 21
poissCdf( ) 112
poissPdf( ) 112
tCdf( ) 156
tPdf( ) 159
χ²GOF( ) 21
χ²Pdf( ) 22

P

P▸Rx( ), rektangulär x-koordinat 110
P▸Ry( ), rektangulär y-koordinat 111
PassErr, flytta fel 111
Pdf( ) 56
permutationer, nPr( ) 106
piecewise( ) 112
poissCdf( ) 112
poissPdf( ) 112
polyEval( ), utvärdera polynom 113
polynom
slump, randPoly( ) 122
utvärdera, polyEval( ) 113
PolyRoots() 114
polär
visa vektor, ▸Polar 113
polära
koordinater, R▸Pθ( ) 120
potens, ^ 180
potensregression,
PowerReg 114, 126, 128, 156
PowerReg, potensregression 114
Prgm, definiera program 115
primtalstest, isPrime( ) 76
procent, % 183
prodSeq() 116
product( ), produkt 116
produkt (P)
mall 5
produkt, product( ) 116
produkt, Π( ) 189
program och programmering
försök, Try 159
rensa fel, ClrErr 22
slut försök, EndTry 159
slut program, EndPrgm 115
visa I/O- skärm, Disp 135
visa I/O-fönster, Disp 41
programmera
definiera program, Prgm 115
flytta fel, PassErr 111
visa data, Disp 41
programmering
visa data, Disp 135
programs
defining private library 38
defining public library 38
propFrac, egentligt bråk 116
punkt
addition, .+ 181
division, .P 182
multiplikation, .* 182
potens, .^ 182
subtraktion, .N 181

Q

QR-faktorisering, QR 117
QR, QR-faktorisering 117
QuadReg, kvadratisk regression 117
QuartReg, kvartär regression 119

R

R, radian 193
R▸Pr( ), polära koordinater 120
R▸Pθ( ), polära koordinater 120
radian, R 193
rand( ), slumptal 121
randBin, slumptal 121
randInt( ), slumpheltal 121

randMat( ), slumpmatris 122
randNorm( ), slumpnorm 122
randPoly( ), slumppolynom 122
randSamp( ) 122
RandSeed, slumptalsfrö 123
real( ), reell 123
reducerad radtrappstegsform, rref() 133
reell, real() 123
ref( ), trappstegsform 124
RefreshProbeVars 125
regression
exponentiell, ExpReg 51
kubisk, CubicReg 33
regressioner
kvadratisk, QuadReg 117
kvartär, QuartReg 119
linjär regression, LinRegAx 79
linjär regression, LinRegBx 78, 80
logaritmisk, LnReg 85
Logistic 88
logistisk, Logistic 89
medium-medium-linje, MedMed 93
MultReg 97
potensregression, PowerReg 114, 126, 128, 156
sinus, SinReg 145
rektangulär x-koordinat, P▸Rx( ) 110
rektangulär y-koordinat, P▸Ry( ) 111
remain( ), rest 126
rensa
fel, ClrErr 22
RequestStr 128
rest, remain( ) 126
resultat, statistik 148
resultvärden, statistik 149
retur, Return 129
Return, retur 129
right( ), höger 129
right, right( ) 47, 168
rk23( ), Runge Kutta-funktion 129
rotate( ), rotera 131
rotera, rotate( ) 131
round( ), avrunda 132
rowAdd( ), matrisradaddition 132
rowDim( ), matrisraddimension 133
rowNorm( ), matrisradnorm 133
rowSwap( ), växla matrisrad 133
rref( ), reducerad trappstegsform 133
räkna poster i en lista villkorligt , countif( ) 30
räkna poster i en lista, count( ) 30

S

sammanfoga/slå ihop, augment( ) 14
samtidiga ekvationer, simult( ) 141
sannolik täthet, normPdf( ) 105
sannolikhet för normalfördelning, normCdf( ) 105
sannolikhet för student-t-fördelning, tCdf( ) 156
sec⁻¹( ), invers sekansfunktion 134
sec( ), sekansfunktion 134
sech⁻¹( ), invers hyperbolisk sekansfunktion 135
sech( ), hyperbolisk sekansfunktion 135
sekundnotation, " 194
seq( ), talföljd 136
seqGen( ) 136
seqn( ) 137
sequence, seq( ) 136-137
setMode( ), ställ in läge 138
sfärisk vektorvisning, ▸Sphere 147
shift( ), skifta 140
sign( ), tecken 141
simult( ), samtidiga ekvationer 141
sin⁻¹( ), arcsinus 143
sin( ), sinus 142
sinh⁻¹( ), hyperbolisk arcsinus 144
sinh( ), hyperbolisk sinus 144
SinReg, sinusregression 145
sinus, sin( ) 142
sinusregression, SinReg 145
skalär
produkt, dotP( ) 44
skifta, shift( ) 140
slinga, Loop 90
slump
matris, randMat( ) 122

norm, randNorm( ) 122
polynom, randPoly( ) 122
slumpfrö, RandSeed 123
slump sampel 122
slut
försök, EndTry 159
medan, EndWhile 169
program, EndPrgm 115
slinga, EndLoop 90
slut medan, EndWhile 169
slut slinga, EndLoop 90
SortA, sortera stigande 146
SortD, sortera fallande 147
sortera
fallande, SortD 147
stigande, SortA 146
språk
hämta språkinformation 65
sqrt( ), kvadratrot 147
standardavvikelse, stdDev( ) 149-150, 166
stat.results 148
stat.values 149
statistik
envariabelstatistik, OneVar 108
fakultet, ! 187
kombinationer, nCr( ) 100
medelvärde, mean( ) 92
median, median( ) 93
permutationer, nPr( ) 106
slump norm, randNorm( ) 122
slumptalsfrö, RandSeed 123
standardavvikelse, stdDev( ) 149-150, 166
tvåvariabelresultat, TwoVar 164
varians, variance( ) 167
stdDevPop( ), standardavvikelse för population 149
stdDevSamp( ), standardavvikelse för urval 150
stegvis funktion (2 steg)
mall 2
stegvis funktion (N steg)
mall 3
Stoppkommando 151
string( ), uttryck till sträng 151
strings
right, right( ) 47, 168
sträng
dimension, dim( ) 41
längd 41
strängar
använda för att skapa variabelnamn 205
format, format( ) 55
formatera 55
höger, right( ) 73, 129
indirection, # 192
inom, inString 72
lägg till, & 187
mittsträng, mid( ) 94
numerisk teckenkod, ord( ) 110
rotera, rotate( ) 131
skifta, shift( ) 140
sträng till uttryck, expr( ) 51
teckensträng, char( ) 20
uttryck till sträng, string( ) 151
vänster, left( ) 77
ställ in
läge, setMode( ) 138
större än eller lika med, | 186
större än, > 185
största gemensamma delare, gcd( ) 59
subMat( ), delmatris 153
subMat( ), undermatris 151
substitution med "|"-operatorn 196
subtrahera, - 177
sum( ), summering 152
sumIf( ) 152
summa (G)
mall 5
summa av kapitalbetalningar 191
summa av räntebetalningar 191
summa, ∑( ) 190
summering, sum( ) 152
sumSeq() 153
svar (sista), Ans 12

T

t-test, tTest 160
T, transponera 153
ta bort
    tomma element från lista 40
    variabel, DelVar 39
tak, ceiling( ) 19, 31
talföljd, seq( ) 136
$\tan^{-1}$( ), arctangens 154
tan( ), tangens 153
tangens, tan( ) 153
$\tanh^{-1}$( ), hyperbolisk arctangens ... 155
tanh( ), hyperbolisk tangens 155
tCdf( ), studentt
    fördelningssannolikhet 156
tecken
    numerisk kod, ord( ) 110
    sträng, char( ) 20
tecken, sign( ) 141
teckensträng, char( ) 20
test for void, isVoid( ) 76
Test_2S, 2-sample F test 58
Text, kommando 156
tidsjusterat pengavärde, antal
    betalningsperioder 163
tidsjusterat pengavärde,
    betalningsbelopp 163
tidsjusterat pengavärde, framtida
    värde 162
tidsjusterat pengavärde, nuvärde .. 163
tidsjusterat pengavärde, ränta 162
TInterval, t konfidensintervall 157
tInterval_2Samp, 2sampel t-
    konfidensintervall 158
tiopotens, 10^( ) 196
tomma element 200
tomma element, ta bort 40
tPdf( ), studentt täthetsfunktion ... 159
trace( ) 159
transponera, T 153
trappstegsform, ref() 124
Try, felhanteringskommando 159
Try, försök 159
tTest, t-test 160
tTest_2Samp, 2-sampel t-test 161
TVM-argument 163
tvmFV( ) 162
tvmI( ) 162
tvmN( ) 163
tvmPmt( ) 163
tvmPV( ) 163
TwoVar, tvåvariabelresultat 164
tvåvariabelresultat, TwoVar 164
täthetsfunktion för student-t, tPdf( ) 159

U

undermatris, subMat( ) 151
ungefärlig, approx( ) 12
unitV( ), enhetsvektor 166
unLock, låsa upp variabel eller
    variabelgrupp 166
user-defined functions 36
user-defined functions and
    programs 38
uteslutning med "|"-operatorn 196
uttryck
    sträng till uttryck, expr( ) 51
utvärdera polynom, polyEval( ) 113

V

variabel
    skapa namn från en
        teckensträng 205
variabler
    lokal, Local 86
    rensa alla enstaka bokstäver ... 22
    ta bort, DelVar 39
variabler och funktioner
    kopiera 24
variabler, låsa och låsa upp 65, 87, 166
varians, variance( ) 167

W

warnCodes( ), Warning codes 168
varningskoder och meddelanden .. 216
varPop( ) 166
varSamp( ), sampelvarians 167

vektorer
enhet, unitV( ) 166
skalärprodukt, dotP( ) 44
vektorprodukt, crossP( ) 31
visa cylindrisk vektor, ▸Cylind .. 35
vektorprodukt, crossP( ) 31
when( ), när 169
While, medan 169
vinkel, angle( ) 9
visa cylindrisk vektor, ▸Cylind 35
visa data, Disp 41, 135
visa som
binär, ▸Base2 16
cylindrisk vektor, ▸Cylind 35
decimal vinkel, ▸DD 36
decimalt heltal, ▸Base10 17
grad/minut/sekund, ▸DMS 43
hexadecimal, ▸Base16 18
ortogonal vektor, ▸Rect 123
polär vektor, ▸Polar 113
sfärisk vektor, ▸Sphere 147
visning i grad/minut/sekund, ▸DMS . 43
void, testa om 76
vänster, left( ) 77

X

x^2, kvadrat 181
XNOR 187
xor, Booleskt exklusivt eller 170

Z

zInterval, z konfidensintervall 171
zInterval_1Prop, 1-proportion z-konfidensintervall 171
zInterval_2Prop, 2-proportion z-konfidensintervall 172
zInterval_2Samp, 2-sampel z-konfidensintervall 172
zTest 173
zTest_1Prop, 1-proportion z-test ... 174
zTest_2Prop, 2-proportion z-test ... 175
zTest_2Samp, 2-sampel z-test 175