

TI-Nspire™ CX

Referenshandbok

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.

© 2021 Texas Instruments Incorporated

De faktiska produkterna kan variera något från de visade bilderna.

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	92
N	100
O	108
P	111
Q	118
R	121
S	135
T	154
U	167
V	167
W	168
X	171
Z	172
Symboler	178
TI-Nspire™ CX II-Kommandon för att rita	201
Grafikprogrammering	201
Grafikskärm	201
Standardvy och inställningar	202
Felmeddelanden på grafikskärmen	203
Ogiltiga kommandon i grafikläge	203
C	204
D	205
F	208
G	210
P	211
S	213
U	215

Tomma element	216
Kortkommandon för att mata in matematiska uttryck	218
EOS™-hierarki (Equation Operating System)	220
TI-Nspire CX II - TI-Basic programmeringsfunktioner	222
Autoindentering i Programeditor	222
Förbättrade felmeddelanden för TI-Basic	222
Konstanter och värden	225
Felkoder och meddelanden	226
Varningskoder och meddelanden	235
Allmän information	237
Hjälp-funktion online	237
Kontakta TI support	237
Service- och garanti-information	237
Innehållsförteckning	238

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



Obs: Se även / (dela), på sidan 180.

Exempel:

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

Mall för Exponent

[^] tangent



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 181.

Exempel:

$$2^3 = 8$$

Mall för Kvadratrot

[ctrl][x²] tangenter



Obs: Se även √() (kvadratrot), på sidan 190.

Exempel:

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

Mall för N:te rot

[ctrl][^] tangenter



Obs: Se även root(), på sidan 132.

Exempel:



Mall för N:te rot

ctrl **^** **tangenter**

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

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

e exponent mall

e^x **tangent**

e

Exempel:

Basen för den naturliga logaritmen e upphöjd till

$$e^1 \quad 2.71828182846$$

Obs: Se även $e^{\wedge}()$, på sidan 44.

Mall för Log

ctrl **10^x** **tangenter**

log

Exempel:

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

$$\log_{10}(2.) \quad 0.5$$

Obs: Se även $\log()$, på sidan 88.

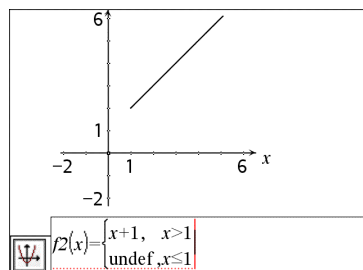
Stegvis mall (2 steg)

Katalog > **tab**

Exempel:

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 $\text{stegvis}()$, på sidan 113.



Stegvis mall (N steg)

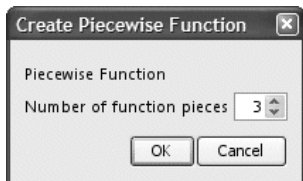
Katalog >



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

Exempel:

Se exemplet på Stegvis mall (2 steg).



Obs: Se även `stegvis()`, på sidan 113.

Mall för System med 2 ekvationer

Katalog >



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 154.

Exempel:

$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=-\frac{5}{2}$$
$$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$$
$$x=-\frac{3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$$

Mall för System med N ekvationer

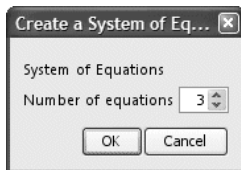
Katalog >



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

Exempel:

Se exemplet på mall för Ekvationssystem (2 ekvationer).



Obs: Se även `system()`, på sidan 154.

Mall för Absolutbelopp

Katalog >



Obs: Se även `abs()`, på sidan 7.

Exempel:

Mall för Absolutbelopp

Katalog > 

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

Mall för dd°mm'ss.ss''

Katalog > 

° ' "

Exempel:

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.

30°15'10"

0.528011

Matrismall (2 x 2)

Katalog > 

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

Exempel:

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 5$

$\begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$

Skapar en 2 x 2-matris.

Matrismall (1 x 2)

Katalog > 

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

Exempel:

$\text{crossP}(\begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \end{bmatrix})$

$\begin{bmatrix} 0 & 0 & -2 \end{bmatrix}$

Matrismall (2 x 1)

Katalog > 

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

Exempel:

$\begin{bmatrix} 5 \\ 8 \end{bmatrix} \cdot 0.01$

$\begin{bmatrix} 0.05 \\ 0.08 \end{bmatrix}$

Matrismall (m x n)

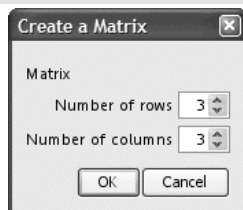
Katalog > 

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

Exempel:

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

$\begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$



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

Mall för Summa (Σ)

$$\sum_{i=0}^{} ()$$

Exempel:

$$\sum_{n=1}^7 (n) \quad 25$$

Obs: Se även $\Sigma()$ (**sumSeq**), på sidan 191.

Mall för Produkt (Π)

$$\prod_{i=0}^{} ()$$

Exempel:

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

Obs: Se även $\Pi()$ (**prodSeq**), på sidan 190.

Mall för förstaderivata

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

Exempel:

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

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

Obs: Se även **d()** (**derivata**), på sidan 189.

Andraderivata, mall

Katalog > 

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

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

Obs: Se även **d()** (**derivata**), på sidan 189.

Exempel:

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

Mall för Bestämd integral

Katalog > 

$$\int_a^b \square dx$$

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

Obs: Se även **nInt()**, på sidan 104.

Exempel:

$$\int_0^{10} x^2 dx \quad 333.333$$

Alfabetisk lista

Poster som inte är alfabetiska (till exempel, +, ! och >) listas i slutet av detta avsnitt och börjar, på sidan 178. 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(Value I)⇒värde

$\left\{ \left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\} \right\}$

{ 1.5708,1.0472 }

abs(List I)⇒lista

$|2-3 \cdot i|$

3.60555

abs(Matrix I)⇒matrix

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.

amortTbl()

Katalog >

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

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

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

Amorteringsfunktion som ger en matrix 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 164.

- 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.

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 $\Sigma\text{Int}()$ och $\Sigma\text{Prn}()$, se på sidan 192, och **bal()**, se på sidan 15.

and (och)

BooleanExpr1 **and**
BooleanExpr2 $\Rightarrow$ Booleskt uttryck

BooleanList1 **and** *BooleanList2* $\Rightarrow$ Boolesk lista

BooleanMatrix1 **and**
BooleanMatrix2 $\Rightarrow$ Boolesk matris

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

Integer1 **and** *Integer2* $\Rightarrow$ 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
--------------	---

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()

angle(*Value1*) $\Rightarrow$ *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
$\text{angle}(\{1+2\cdot i, 3+0\cdot i, 0-4\cdot i\})$	$\{1.10715, 0, -1.5708\}$

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

vinkel(*Matrix1*) $\Rightarrow$ *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

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 149.)

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	Sammanlagda (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*List1,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 149.)

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.PVallInteract	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	56 56
Ger resultatet på det senast beräknade uttrycket.	56+4 60
	60+4 64

approx()	Katalog > 
approx(ValueI)⇒tal	$\text{approx}\left(\frac{1}{3}\right)$ 0.333333
Visar resultatet av beräkningen av argumentet som ett uttryck med decimala värden, när så är möjligt, oavsett den aktuella inställningen av Auto eller Ungefärlig .	$\text{approx}\left(\left\{\frac{1}{3}, \frac{1}{9}\right\}\right)$ { 0.333333, 0.111111 }
Detta motsvarar att skriva in argumentet och trycka på ctrl enter .	$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$ { 0., -1. }
	$\text{approx}\left(\left[\sqrt{2} \quad \sqrt{3}\right]\right)$ [1.41421 1.73205]
	$\text{approx}\left(\left[\frac{1}{3} \quad \frac{1}{9}\right]\right)$ [0.333333 0.111111]
approx(ListI)⇒lista	$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$ { 0., -1. }
approx(MatrixI)⇒matrix	$\text{approx}\left(\left[\sqrt{2} \quad \sqrt{3}\right]\right)$ [1.41421 1.73205]
Ger en lista eller <i>matrix</i> där varje element har beräknats till ett decimalt värde, när så är möjligt.	

►approxFraction()**Katalog >** **Value ►approxFraction([Tol])⇒värde**

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

List ►approxFraction([Tol])⇒lista

$$0.8333333333333333 \blacktriangleright \text{approxFraction}(5.E-14)$$

Matrix ►approxFraction([Tol])⇒matrix

$$\frac{5}{6}$$

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

$$\{\pi, 1.5\} \blacktriangleright \text{approxFraction}(5.E-14)$$

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

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

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

$$\text{approxRational}(0.333, 5 \cdot 10^{-5}) \quad \frac{333}{1000}$$

approxRational(List[, Tol])⇒lista

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

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

approxRational(Matrix[, Tol])⇒matrix

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

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 $\text{csch}^{-1}()$, på sidan 33.
arcsec()	Se $\text{sec}^{-1}()$, på sidan 135.
arcsech()	Se $\text{sech}^{-1}()$, på sidan 136.
arcsin()	Se $\sin^{-1}()$, på sidan 144.
arcsinh()	Se $\sinh^{-1}()$, på sidan 145.
arctan()	Se $\tan^{-1}()$, på sidan 155.
arctanh()	Se $\tanh^{-1}()$, på sidan 156.

augment()	Katalog > 
augment(List1, List2) ⇒ lista	$\text{augment}(\{1, -3, 2\}, \{5, 4\}) \quad \{1, -3, 2, 5, 4\}$
Ger en ny lista med <i>List2</i> inlagd i slutet på <i>List1</i> .	
augment(Matrix1, Matrix2) ⇒ matris	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow m2 \quad \begin{bmatrix} 5 \\ 6 \end{bmatrix}$ $\text{augment}(m1, m2) \quad \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}$
Ger en ny matris med <i>Matrix2</i> fogad till <i>Matrix1</i> . När kommatecknet (,) används måste matriserna ha samma raddimensioner och <i>Matrix2</i> fogas till <i>Matrix1</i> som nya kolumner. Ändrar inte <i>Matrix1</i> eller <i>Matrix2</i> .	

avgRC()Katalog > 

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

$x:=2$	2
--------	---

$\text{avgRC}(x^2-x+2,x)$	3.001
---------------------------	-------

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

$\text{avgRC}(x^2-x+2,x,1)$	3.1
-----------------------------	-----

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

$\text{avgRC}(x^2-x+2,x,3)$	6
-----------------------------	---

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 "I" för variabeln.

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

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

B**bal()**Katalog > 

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

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

bal(*NPmt*,*amortTable*)⇒värde

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

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 164.

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 164.

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

- 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 192.

►Base2

Integer1 ►**Base2**⇒*heltal*

256►Base2

0b100000000

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

0h1F►Base2

0b11111

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,

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

-2⁶³ 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.

2⁶³ blir -2⁶³ och visas som
0h8000000000000000 i Hexadecimalt
basläge 0b100...000 (63 nollor) i Binärt
basläge

2⁶⁴ blir 0 och visas som 0h0 i Hexadecimalt
basläge 0b0 i Binärt basläge

-2⁶³ - 1 blir 2⁶³ - 1 och visas som
0h7FFFFFFFFFFFFFFF i Hexadecimalt
basläge 0b111...111 (64 ettor) i Binärt
basläge

►Base10

Integer1 ►Base10⇒heltal

0b10011►Base10	19
----------------	----

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

0h1F►Base10	31
-------------	----

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.

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

Integer1 ►Base16⇒*heltal*

256►Base16

0h100

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

0b111100001111►Base16

0hFOF

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.

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 $P(0 \leq X \leq upBound)$ ⇒*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 $P(X \leq upBound)$, sätt $lowBound=0$

binomPdf()Katalog > 

binomPdf(n,p) $\Rightarrow$ *lista*

binomPdf($n,p,XVal$) $\Rightarrow$ *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($ValueI$) $\Rightarrow$ *värde*

$\text{ceiling}(.456)$	1.
------------------------	----

Ger det närmaste heltal som är $\geq$ argumentet.

Argumentet kan vara ett reellt eller ett komplext tal.

Obs: Se även **floor()**.

ceiling($ListI$) $\Rightarrow$ *lista*

$\text{ceiling}(\{-3.1, 1, 2.5\})$	$\{-3., 1, 3.\}$
------------------------------------	------------------

ceiling($MatrixI$) $\Rightarrow$ *matris*

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

Ger en lista eller matris över taket för varje element.

centralDiff()Katalog > 

**centralDiff($UtrI, Var [=Värde]$
[, $Steg$])** $\Rightarrow$ *uttryck*

$\text{centralDiff}[\cos(x), x] x = \frac{\pi}{2}$	-1.
--	-----

**centralDiff($UtrI, Var$
[, $Steg$]) | $Var=Värde$** $\Rightarrow$ *uttryck*

centralDiff($UtrI, Var [=Värde]$

[,Lista])⇒*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()

char(*Integer*)⇒*tecken*

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

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

χ²2way

χ²2way *ObsMatrix*

chi22way *ObsMatrix*

Beräknar ett χ²-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 149.)

För information om effekten av tomma element i en matris, se "Tomma element" (på sidan 216).

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

 χ^2 Cdf()

$\chi^2\text{Cdf}(\text{lowBound}, \text{upBound}, \text{df}) \Rightarrow \text{tal}$ om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

$\text{chi2Cdf}(\text{lowBound}, \text{upBound}, \text{df}) \Rightarrow \text{tal}$ 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 $P(X \leq \text{upBound})$, sätt *lowBound* = 0.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

 χ^2 GOF

$\chi^2\text{GOF } \text{obsList}, \text{expList}, \text{df}$

$\text{chi2GOF } \text{obsList}, \text{expList}, \text{df}$

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 149.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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

 χ^2 Pdf()

$\chi^2\text{Pdf}(XVal,df) \Rightarrow tal$ om $XVal$ är ett tal,
lista om $XVal$ är en lista

$\text{chi2Pdf}(XVal,df) \Rightarrow tal$ 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 216).

ClearAZ

ClearAZ

$5 \rightarrow b$	5
-------------------	---

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

b	5
-----	---

ClearAZ	Done
---------	------

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 167.

b	"Error: Variable is not defined"
-----	----------------------------------

ClrErr

ClrErr

För ett exempel på **ClrErr**, se exempel 2 under kommandot **Try**, på sidan 161.

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

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 112 och **Try**, på sidan 160.

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

colAugment()

colAugment(*Matrix1*, *Matrix2*) $\Rightarrow$ *matrix*

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*.

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

colDim()

colDim(*Matrix*) $\Rightarrow$ *uttryck*

Ger antalet kolumner i *Matrix*.

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

Obs: Se även **rowDim**().

colNorm()

colNorm(*Matrix*) $\Rightarrow$ *uttryck*

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

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

colNorm()Katalog > 

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

conj()Katalog > 

conj(*Value1*) $\Rightarrow$ värde

$$\text{conj}(1+2 \cdot i) \qquad 1-2 \cdot i$$

conj(*List1*) $\Rightarrow$ lista

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

conj(*Matrix1*) $\Rightarrow$ matris

Ger argumentets komplexkonjugat.

Obs: Alla odefinierade variabler behandlas som reella variabler.

constructMat()Katalog > 

constructMat
(*Expr*, *Var1*, *Var2*, *numRows*, *numCols*)
 $\Rightarrow$ matris

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

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*.

CopyVarKatalog > 

CopyVar *Var1*, *Var2*

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

CopyVar *Var1*., *Var2*.

$$\text{Define } b(x) = x^2 \qquad \text{Done}$$

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

$$\begin{array}{l} \text{CopyVar } a, c: c(4) \qquad \frac{1}{4} \\ \text{CopyVar } b, c: c(4) \qquad 16 \end{array}$$

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.

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.mn*-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.

<i>aa.a</i> :=45	45																
<i>aa.b</i> :=6.78	6.78																
CopyVar <i>aa.,bb.</i>	<i>Done</i>																
getVarInfo()	<table><tr><td><i>aa.a</i></td><td>"NUM"</td><td>"0"</td><td>0</td></tr><tr><td><i>aa.b</i></td><td>"NUM"</td><td>"0"</td><td>0</td></tr><tr><td><i>bb.a</i></td><td>"NUM"</td><td>"0"</td><td>0</td></tr><tr><td><i>bb.b</i></td><td>"NUM"</td><td>"0"</td><td>0</td></tr></table>	<i>aa.a</i>	"NUM"	"0"	0	<i>aa.b</i>	"NUM"	"0"	0	<i>bb.a</i>	"NUM"	"0"	0	<i>bb.b</i>	"NUM"	"0"	0
<i>aa.a</i>	"NUM"	"0"	0														
<i>aa.b</i>	"NUM"	"0"	0														
<i>bb.a</i>	"NUM"	"0"	0														
<i>bb.b</i>	"NUM"	"0"	0														

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

Beräknar korrelationsmatrisen för den
 sammanfogade matrisen $[List1, List2, \dots, List20]$.

cos(Value1)⇒värde

I vinkelläget Grader:

$$\mathbf{cos}(List1) \Rightarrow lista$$

$\cos\left(\left(\frac{\pi}{4}\right)_r\right)$	0.707107
---	----------

cos(Value1) ger argumentets cosinus som ett värde.

$\cos(45)$	0.707107
------------	----------

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

$$\cos(\{0, 60, 90\}) \qquad \{1, 0.5, 0\}$$

I vinkelläget Nygrader:

cos() **tangent**

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.

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

I vinkelläget Radianer:

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

$$\cos(45^\circ) \quad 0.707107$$

cos(squareMatrixI)⇒kvadratMatrix

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

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

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

squareMatrixI 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.

I vinkelläget Radianer:

$$\cos\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$$

cos⁻¹() **tangent**

cos⁻¹(ValueI)⇒värde

I vinkelläget Grader:

cos⁻¹()**trig tangent****cos⁻¹(List1)**⇒listacos⁻¹(1)

0.

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

I vinkelläget Nygrader:

cos⁻¹(List1) ger en lista på invers cosinus för varje element i List1.cos⁻¹(0)

100.

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

I vinkelläget Radianer:

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

I vinkelläget Radianer och i Rektangulärt komplext format:

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.cos⁻¹ $\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$ $\begin{bmatrix} 1.73485+0.064606\cdot i & -1.49086+2.10514 \\ -0.725533+1.51594\cdot i & 0.623491+0.778369 \\ -2.08316+2.63205\cdot i & 1.79018-1.27182\cdot i \end{bmatrix}$

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

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

cosh()**Katalog >****cosh(Value1)**⇒värde

I vinkelläget Grader:

cosh(List1)⇒listacosh $\left(\left\{\frac{\pi}{4}\right\}r\right)$

1.74671E19

cosh(Value1) ger argumentets hyperboliska cosinus.**cosh(List1)** ger en lista på hyperbolisk cosinus för varje element i List1.**cosh(squareMatrix1)**⇒kvadratMatris

I vinkelläget Radianer:

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.

cosh()

Katalog > 

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

$$\cosh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 421.255 & 253.909 & 216.905 \\ 327.635 & 255.301 & 202.958 \\ 226.297 & 216.623 & 167.628 \end{bmatrix}$$

cosh⁻¹()

Katalog > 

cosh⁻¹(Value1) ⇒ värde

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

cosh⁻¹(List1) ⇒ lista

$$\cosh^{-1}(\{1, 2, 1, 3\}) = \{0, 1.37286, 1.76275\}$$

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:

$$\cosh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 2.52503+1.73485 \cdot i & -0.009241-1.49086 \cdot i & 0.486969-0.725533 \cdot i \\ 0.486969-0.725533 \cdot i & 1.66262+0.623491 \cdot i & -0.322354-2.08316 \cdot i \\ -0.322354-2.08316 \cdot i & 1.26707+1.79018 \cdot i & 1.26707+1.79018 \cdot i \end{bmatrix}$$

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

cot()

 tangent

cot(Value1) ⇒ värde

I vinkelläget Grader:

cot(List1) ⇒ lista

$$\cot(45) = 1.$$

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

cot()

 tangent

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda $^{\circ}$, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Nygrader:

$\cot(50)$	1.
------------	----

I vinkelläget Radianer:

$\cot(\{1,2,1,3\})$	$\{0.642093, -0.584848, -7.01525\}$
---------------------	-------------------------------------

$\cot^{-1}()$

 tangent

$\cot^{-1}(ValueI) \Rightarrow \text{värde}$

I vinkelläget Grader:

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

$\cot^{-1}(ListI) \Rightarrow \text{lista}$

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

I vinkelläget Nygrader:

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

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

I vinkelläget Radianer:

$\cot^{-1}(1)$	0.785398
----------------	----------

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

coth()

Katalog > 

$\coth(ValueI) \Rightarrow \text{värde}$

$\coth(1.2)$	1.19954
--------------	---------

$\coth(ListI) \Rightarrow \text{lista}$

$\coth(\{1,3,2\})$	$\{1.31304, 1.00333\}$
--------------------	------------------------

Ger hyperbolisk cotangens för *ValueI* eller en lista på hyperbolisk cotangens för alla element i *ListI*.

$\coth^{-1}()$

Katalog > 

$\coth^{-1}(ValueI) \Rightarrow \text{värde}$

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

$\coth^{-1}(ListI) \Rightarrow \text{lista}$

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

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 (...)**.

count()

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 216.

count(2,4,6)	3				
count({ 2,4,6 })	3				
count(2,{ 4,6 }, <table border="1" data-bbox="683 478 761 533"><tr><td>8</td><td>10</td></tr><tr><td>12</td><td>14</td></tr></table>)	8	10	12	14	7
8	10				
12	14				

countif()

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.

countif({ 1,3,"abc",undef,3,1 },3)	2
Räknar antalet element som är lika med 3.	
countif({ "abc","def","abc",3},"def")	1

Räknar antalet element som är lika med "def."

countIf()Katalog > 

- 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 216.

Obs: Se även **sumIf()**, på sidan 153 och **frequency()**, på sidan 57.

<code>countIf({1,3,5,7,9},?<5)</code>	2
--	---

Räknar 1 och 3.

<code>countIf({1,3,5,7,9},2<?<8)</code>	3
---	---

Räknar 3, 5 och 7.

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

<code>polyRoots(y^3+1,y)</code>	{-1}
---------------------------------	------

<code>cPolyRoots(y^3+1,y)</code>	{-1,0.5-0.866025 <i>i</i> ,0.5+0.866025 <i>i</i> }
----------------------------------	--

<code>polyRoots(x^2+2*x+1,x)</code>	{-1,-1}
-------------------------------------	---------

<code>cPolyRoots({1,2,1})</code>	{-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.

<code>crossP({0.1,2.2,-5},{1,-0.5,0})</code>	{-2.5,-5,-2.25}
--	-----------------

crossP()Katalog > **crossP**(*Vector1*, *Vector2*) $\Rightarrow$ *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 ($\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$)	$\begin{bmatrix} -3 & 6 & -3 \end{bmatrix}$
crossP ($\begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \end{bmatrix}$)	$\begin{bmatrix} 0 & 0 & -2 \end{bmatrix}$

csc() **tangent****csc**(*Value1*) $\Rightarrow$ *värde*

I vinkelläget Grader:

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

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

csc (45)	1.41421
-----------------	---------

I vinkelläget Nygrader:

csc (50)	1.41421
-----------------	---------

I vinkelläget Radianer:

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

csc⁻¹() **tangent****csc⁻¹**(*Value1*) $\Rightarrow$ *värde*

I vinkelläget Grader:

csc⁻¹(*List1*) $\Rightarrow$ *lista*

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

csc⁻¹ (1)	90.
-----------------------------	-----

I vinkelläget Nygrader:

csc⁻¹ (1)	100.
-----------------------------	------

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 Radianer:

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

csch()Katalog > **csch(ValueI)** $\Rightarrow$ värde

csch(3) 0.099822

csch(ListI) $\Rightarrow$ listacsch({1,2,1,4})
{0.850918,0.248641,0.036644}

Ger den hyperboliska cosekanten för *ValueI* eller en lista på den hyperboliska cosekanten för alla element i *ListI*.

csch⁻¹()Katalog > **csch⁻¹(Value)** $\Rightarrow$ värdecsch⁻¹(1) 0.881374**csch⁻¹(ListI)** $\Rightarrow$ listacsch⁻¹({1,2,1,3})
{0.881374,0.459815,0.32745}

Ger den inversa hyperboliska cosekanten för *ValueI* eller en lista på den inversa hyperboliska cosekanten för alla element i *ListI*.

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

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

Utför en tredjegrads regressionsanalys = $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*, på sidan 149.

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
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 <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

cumulativeSum()

cumulativeSum(List1) ⇒ lista

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

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

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.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
cumulativeSum(m1)	$\begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 12 \end{bmatrix}$

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 216.

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**).

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

Funktion som listar heltal från 1 till 100 utom 50.

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

►Cylind

Vector ►**Cylind**

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

Visar rad- eller kolumnvektorn i cylindrisk form $[r, \angle \theta, z]$.

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

$[2 \ 2 \ 3]$ ► Cylind	
	$[2.82843 \ \angle 0.785398 \ 3.]$

D**dbd()**

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.

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

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)

DDMM.YY (format som ofta används i Europa)

►DDKatalog >

Expr1 ►DD⇒värde

I vinkelläget Grader:

List1 ►DD⇒lista

(1.5°) ►DD	1.5°
$(45^\circ 22' 14.3'')$ ►DD	45.3706°
$(\{45^\circ 22' 14.3'', 60^\circ 0' 0''\})$ ►DD	$\{45.3706^\circ, 60^\circ\}$

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 Nygrader:

1►DD	$\frac{9}{10}^\circ$
------	----------------------

I vinkelläget Radianer:

(1.5) ►DD	85.9437°
-------------	----------

►DecimalKatalog >

Number1 ►Decimal⇒värde

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

List1 ►Decimal⇒värde

Matrix1 ►Decimal⇒värde

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

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

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 $\rightarrow$ *Function*(*Param1*,*Param2*).

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

Block

EndFunc

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

Block

EndPrgm

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

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

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

Define (Definiera)

Katalog > 

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**).

Obs för att mata in exemplet: Se avsnittet Räkna 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 g(x,y)=Prgm
  If x>y Then
    Disp x," greater than ",y
  Else
    Disp x," not greater than ",y
  EndIf
EndPrgm
```

Done

g(3,-7)

3 greater than -7

Done

Define LibPriv

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 37 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

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 37 och **Define LibPriv**, på sidan 38.

deltaList()

Se Δ List(), på sidan 84.

DelVar

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

$2 \rightarrow a$	2
-------------------	---

DelVar *Var*.

$(a+2)^2$	16
-----------	----

Tar bort den specificerade variabeln eller variabelgruppen från minnet.

DelVar <i>a</i>	Done
-----------------	------

$(a+2)^2$	"Error: Variable is not defined"
-----------	----------------------------------

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 167.

DelVar

Katalog > 

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.

<i>aa.a:=</i> 45	45									
<i>aa.b:=</i> 5.67	5.67									
<i>aa.c:=</i> 78.9	78.9									
getVarInfo()	<table><tr><td><i>aa.a</i></td><td>"NUM"</td><td>"0"</td></tr><tr><td><i>aa.b</i></td><td>"NUM"</td><td>"0"</td></tr><tr><td><i>aa.c</i></td><td>"NUM"</td><td>"0"</td></tr></table>	<i>aa.a</i>	"NUM"	"0"	<i>aa.b</i>	"NUM"	"0"	<i>aa.c</i>	"NUM"	"0"
<i>aa.a</i>	"NUM"	"0"								
<i>aa.b</i>	"NUM"	"0"								
<i>aa.c</i>	"NUM"	"0"								
DelVar <i>aa.</i>	<i>Done</i>									
getVarInfo()	"NONE"									

delVoid()

Katalog > 

delVoid(*List1*)⇒*lista*

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

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

För mer information om tomma element, se på sidan 216.

det()

Katalog > 

det(*squareMatrix*[, *Tolerance*])⇒*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*.

$\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	-2
$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow mat1$	$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix}$
$\det(mat1)$	0
$\det(mat1,1)$	1.E20

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

$$5E-14 \cdot \max(\dim(squareMatrix)) \cdot \text{rowNorm}(squareMatrix)$$

diag()Katalog > **diag(List)**⇒*matrix*

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

diag(rowMatrix)⇒*matrix***diag(columnMatrix)**⇒*matrix*

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

diag(squareMatrix)⇒*radMatrix*

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

$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$	$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$
diag(<i>Ans</i>)	$\begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$

squareMatrix måste vara kvadratisk.

dim()Katalog > **dim(List)**⇒*heltal*

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

Ger dimensionen på *List*.

dim(Matrix)⇒*lista*

dim($\begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{pmatrix}$)	{3,2}
--	-------

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

dim(String)⇒*heltal*

dim("Hello")	5
dim("Hello "&"there")	11

Ger antalet tecken i teckensträngen *String*.

DispKatalog > **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 exempel: Se avsnittet Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

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

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

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

Radnumret kan specificeras som ett uttryck.

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

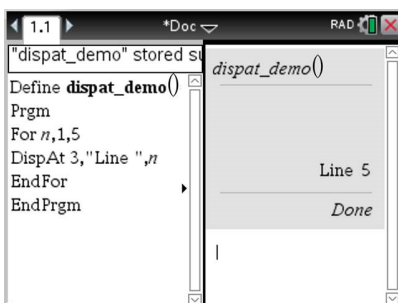
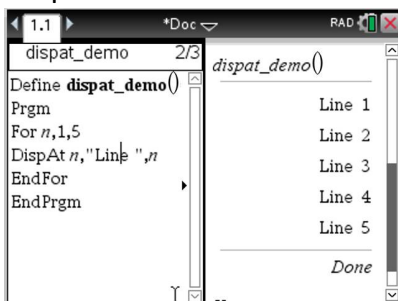
Detta kommando möjliggör kontrollpanelsliknande utdata från program där värdet av ett uttryck eller från en sensoravläsning uppdateras på samma rad.

DispAt och 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.

DispAt

Exempel



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

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*Value* ►DMS

I vinkelläget Grader:

List ►DMS

{45.371}►DMS 45°22'15.6"

Matrix ►DMS

{ {45.371,60} }►DMS { 45°22'15.6",60° }

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

Tolkar argumentet som en vinkel och visar motsvarande DMS-värde (DDDDDD°MM'SS.ss"). Se °, ', ", på sidan 195 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()

dotP(List1, List2)⇒uttryck

$\text{dotP}(\{1,2\},\{5,6\})$	17
--------------------------------	----

Ger "prick"-produkten av två listor.

dotP(Vector1, Vector2)⇒uttryck

$\text{dotP}([1\ 2\ 3],[4\ 5\ 6])$	32
------------------------------------	----

Ger "prick"-produkten av två vektorer.

Båda måste vara radvektorer eller båda måste vara kolumnvektorer.

E

e^()

e^(Value1)⇒värde

e^1	2.71828
-------	---------

Ger e upphöjt till potensen Value1.

e^{3^2}	8103.08
-----------	---------

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

Obs: Att trycka på  för att visa e^(skiljer sig från att trycka på tecknet  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

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

Ger e upphöjt till potensen för varje element i List1.

e^()**e^x**-tangent**e^(*squareMatrixI*)**⇒*kvadratMatrix*

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

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

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

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 104.

eff(5.75,12)	5.90398
--------------	---------

eigVc()**Katalog >** **eigVc(*squareMatrix*)**⇒*matrix*

I Rektangulärt komplext format:

Ger en matris som innehåller egenvektorer 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. Egenvektorer är normaliserade, vilket innebär att om $V = [x_1, x_2, \dots, x_n]$, så är:

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

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 egenvektorer beräknas med en Schur-faktorisering.

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

eigVc(m1)	
$\begin{bmatrix} -0.800906 & 0.767947 & (\\ 0.484029 & 0.573804+0.052258 \cdot i & 0.5738 \\ 0.352512 & 0.262687+0.096286 \cdot i & 0.2626 \end{bmatrix}$	

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

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 egenvektorer beräknas från den övre Hessenberg-matrisen.

I läget Rektangulärt komplext format:

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

$$\text{eigVl}(m1) \\ \{-4.40941, 2.20471 + 0.763006i, 2.20471 - 0.763006i\}$$

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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $g(x)$ =Func

If $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ Then

Return $-x$

ElseIf $x \geq 0$ and $x \neq 10$ 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 91.

EndPrgm

Se Prgm, på sidan 116.

EndTry

Se Try, på sidan 160.

EndWhile

Se While, på sidan 170.

euler ()

Katalog > 

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

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

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

Använder Eulers metod för att lösa
systemet

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

Differentialekvation:

$$y' = 0,001 \cdot y \cdot (100 - y) \text{ och } y(0) = 10$$

euler(0.001·y·(100-y), t, y, {0, 100}, 10, 1)				
0.	1.	2.	3.	4.
10.	10.9	11.8712	12.9174	14.042

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

Ekvationssystem:

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

$$\text{med } y1(0) = 2 \text{ och } y2(0) = 5$$

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.

$$\text{euler} \left(\begin{array}{c} \left\{ -y1+0.1 \cdot y1 \cdot y2, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1 \right\} \\ \left\{ 3 \cdot y2 - y1 \cdot y2 \right\} \end{array} \right) = \begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix}$$

$Expr$ 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 $\text{sign}(VarStep) = \text{sign}(VarMax - Var0)$ och lösningar ges vid $Var0 + i \cdot VarStep$ för alla $i=0, 1, 2, \dots$ sådana att $Var0 + i \cdot VarStep$ är i intervallet $[Var0, VarMax]$ (det kanske inte finns ett lösningsvärde vid $VarMax$).

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

eval ()

Hubb-meny

$\text{eval}(Expr) \Rightarrow string$

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

eval ()

Hubb-meny

eval() är giltig endast i TI-Innovator™ Hub 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.

<i>lum:=127</i>	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.

<i>fadein()</i>	Done
<i>n:=0.25</i>	0.25
<i>m:=8</i>	8
<i>n·m</i>	2.
Send "SET COLOR.BLUE ON TIME eval(n·m)"	Done
<i>iostr.SendAns</i>	"SET COLOR.BLUE ON TIME 2"

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

Obs: Se även **Get** (på sidan 60), **GetStr** (på sidan 67), och **Send** (på sidan 136).

Exit

Katalog >

Exit

Funktionslista:

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

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

exp()** -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.

exp(List1)⇒lista

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

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^1	2.71828
-------	---------

e^{3^2}	8103.08
-----------	---------

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

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

expr(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

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

Beräknar den exponentiella regressionen $y = a \cdot (b)^x$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot (b)^x$
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 <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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  och tryck på  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

(
lowBound
,upBound,dfNumer,dfDenom) $\Rightarrow$ number om
lowBound och *upBound* är tal, lista om
lowBound och *upBound* är listor

FCdf

(
lowBound
,upBound,dfNumer,dfDenom) $\Rightarrow$ 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 $P(X \leq \text{övrGräns})$, ange *undrGräns* = 0.

Fill

Fill *Value*, *matrixVar* $\Rightarrow$ *matrix*

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, <i>amatrix</i>	Done
<i>amatrix</i>	$\begin{bmatrix} 1.01 & 1.01 \\ 1.01 & 1.01 \end{bmatrix}$

Fill *Value*, *listVar* $\Rightarrow$ *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, <i>alist</i>	Done
<i>alist</i>	$\{1.01,1.01,1.01,1.01,1.01\}$

FiveNumSummary

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 149.)

X representerar en lista på aktuella data.

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 216.

Resultatvariabel	Beskrivning
stat.MinX	Minsta x-värde
stat.Q ₁ X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q ₃ X	Övre kvartil för x
stat.MaxX	Största x-värde

floor()

floor(Value I) $\Rightarrow$ heltal

floor(-2.14) -3.

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

Argumentet kan vara ett reellt eller ett komplext tal.

floor(List I) $\Rightarrow$ lista

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

floor(Matrix I) $\Rightarrow$ matris

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

Ger en lista eller matris med golvvärden för varje element.

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

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

Done

Local *tempsum,step,i*

0 → *tempsum*

1 → *step*

For *i*,1,100,*step*

tempsum + *i* → *tempsum*

EndFor

EndFunc

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"

G[n][c]: Samma som fast format, men separerar också siffror 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 enskilt tecken som specificerar vad som skall ersättas för baspunkten.

fPart()

Katalog >

fPart(*Expr I*) $\Rightarrow$ *uttryck*

fPart(-1.234)	-0.234
---------------	--------

fPart(*List I*) $\Rightarrow$ *lista*

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

fPart(*Matrix I*) $\Rightarrow$ *matrix*

Ger argumentets bråkdel.

Ger, för en lista eller matrix, elementens bråkdelar.

Argumentet kan vara ett reellt eller ett komplext tal.

FPdf()

Katalog >

FPdf(*XVal*, *dfNumer*, *dfDenom*) $\Rightarrow$ *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

(*List I*, *freqIntegerList*) $\Rightarrow$ *lista*

Ger en lista som innehåller elementen från *List I* 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 }

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 216.

frequency()

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 { $? \leq b(1)$, $b(1) < ? \leq b(2)$, ..., $b(n-1) < ? \leq b(n)$, $b(n) > ?$ }. 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, $? \leq b(1)$), countIf(list, $b(1) < ? \leq b(2)$), ..., countIf(list, $b(n-1) < ? \leq b(n)$), 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 216.

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

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

<i>datalist</i> ={1,2,e,3,π,4,5,6,"hello",7}
{1,2,2.71828,3,3.14159,4,5,6,"hello",7}
frequency(<i>datalist</i> ,{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 *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 149.)

eller $H_a: \sigma_1 > \sigma_2$, sätt *Hypoth*>0

För $H_a: \sigma_1 \neq \sigma_2$ (förinställning), sätt *Hypoth*=0

För $H_a: \sigma_1 < \sigma_2$, sätt *Hypoth*<0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.F	Beräknad $\hat{U}$ -statistik för datasekvensen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.dfNumer	täljare, frihetsgrader = $n_1 - 1$
stat.dfDenom	nämnare, frihetsgrader = $n_2 - 1$
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>
stat.x1_bar stat.x2_bar	Medelvärden hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>
stat.n1, stat.n2	Storlek på urvalen

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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $g(x)=$ Func

Done

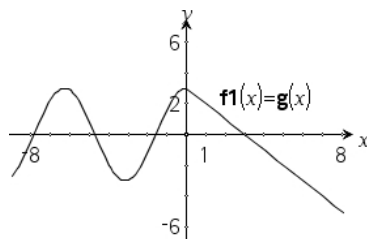
If $x < 0$ ThenReturn $3 \cdot \cos(x)$

Else

Return $3-x$

EndIf

EndFunc

Resultat från plottning av $g(x)$ **G****gcd()****gcd**(*Value1*, *Value2*) $\Rightarrow$ uttryck $\text{gcd}(18, 33)$

3

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 bråk i flyttalsform.

gcd(*List1*, *List2*) $\Rightarrow$ lista $\text{gcd}(\{12, 14, 16\}, \{9, 7, 5\})$ {3, 7, 1}

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

gcd(*Matrix1*, *Matrix2*) $\Rightarrow$ matris
$$\text{gcd}\left(\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}, \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}\right) \quad \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

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

geomCdf()

geomCdf(*p*, *lowBound*, *upBound*) $\Rightarrow$ tal om *lowBound* och *upBound* är tal, lista om

lowBound och *upBound* är listor

geomCdf(*p*,*upBound*) för $P(1 \leq X \leq upBound) \Rightarrow tal$ 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 $P(X \leq upBound)$, sätt *lowBound* = 1.

geomPdf()

geomPdf(*p*,*XVal*) $\Rightarrow tal$ 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(*promptString*,] *var*[, *statusVar*]

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

Programmeringskommando: Hämtar ett värde från en ansluten TI-Innovator™ Hub 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 <i>lightval</i>	Done
<i>lightval</i>	0.347922

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

Get "READ BRIGHTNESS", <i>lightval</i>	Done
<i>lightval</i>	0.378441

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 136.

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

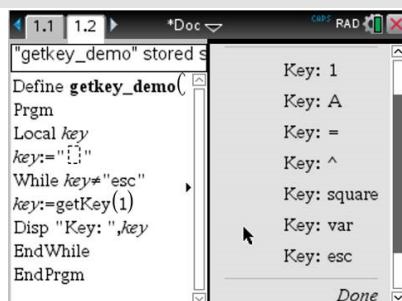
getKey()

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/emulator tangent	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*

getLangInfo()

"en"

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"

getLockInfo()

Katalog > 

getLockInfo(Var) ⇒ *värde*

a:=65

65

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

Lock a

Done

getLockInfo(a)

1

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

a:=75

"Error: Variable is locked."

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

DelVar a

"Error: Variable is locked."

Unlock a

Done

a:=75

75

DelVar a

Done

Se **Lock**, på sidan 87 och **unLock**, på sidan 167.

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 139.

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

Lägets namn	Lägesheltal	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 Komplex)	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**(*FractionI*) $\Rightarrow$ *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*]Se **Get** för exempel.

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 136.

getType()Katalog > **getType**(*var*) $\Rightarrow$ *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"
<i>DelVar temp</i>	<i>Done</i>
$\text{getType}(temp)$	"NONE"

getVarInfo() $\Rightarrow$ *matris* eller *sträng*

getVarInfo(LibNameString) $\Rightarrow$ *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()	<table><tr><td>x</td><td>"NUM"</td><td>"{"</td><td>"</td><td>1</td></tr><tr><td>y</td><td>"LIST"</td><td>"LibPriv"</td><td>"</td><td>0</td></tr><tr><td>z</td><td>"FUNC"</td><td>"LibPub"</td><td>"</td><td>0</td></tr></table>	x	"NUM"	"{"	"	1	y	"LIST"	"LibPriv"	"	0	z	"FUNC"	"LibPub"	"	0
x	"NUM"	"{"	"	1												
y	"LIST"	"LibPriv"	"	0												
z	"FUNC"	"LibPub"	"	0												
getVarInfo(tmp3)	"Error: Argument must be a string"															
getVarInfo(tmp3)	<table><tr><td>[volcy12</td><td>"NONE"</td><td>"LibPub"</td><td>"</td><td>0]</td></tr></table>	[volcy12	"NONE"	"LibPub"	"	0]										
[volcy12	"NONE"	"LibPub"	"	0]												

$a:=1$		1															
$b:=[1\ 2]$		$[1\ 2]$															
$c:=[1\ 3\ 7]$		$[1\ 3\ 7]$															
$vs:=\text{getVarInfo}()$	<table><tr><td>a</td><td>"NUM"</td><td>"{"</td><td>"</td><td>0</td></tr><tr><td>b</td><td>"MAT"</td><td>"{"</td><td>"</td><td>0</td></tr><tr><td>c</td><td>"MAT"</td><td>"{"</td><td>"</td><td>0</td></tr></table>	a	"NUM"	"{"	"	0	b	"MAT"	"{"	"	0	c	"MAT"	"{"	"	0	
a	"NUM"	"{"	"	0													
b	"MAT"	"{"	"	0													
c	"MAT"	"{"	"	0													
$vs[1]$	$[1\ \text{"NUM"}\ \text{"{"}\ \text{"}\ 0]$																
$vs[1,1]$		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 <i>temp,i</i>	
$0 \rightarrow temp$	
$1 \rightarrow i$	
Lbl <i>top</i>	
$temp+i \rightarrow temp$	
If $i < 10$ Then	
$i+1 \rightarrow i$	
Goto <i>top</i>	
EndIf	
Return <i>temp</i>	
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:

$(1.5) \blacktriangleright \text{Grad}$	$(1.66667)^{\circ}$
---	---------------------

I vinkelläget Radianer:

$(1.5) \blacktriangleright \text{Grad}$	$(95.493)^{\circ}$
---	--------------------

/

identity()**Katalog** > **identity**(*Integer*) ⇒ *matrix*

Ger identitetsmatrisen med dimensionen *Integer*.

Integer måste vara positivt heltal.

identity(4)	<table><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

If BooleanExpr 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.

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	Done
If $x < 0$ Then	
Return $\neg x$	
Else	
Return x	
EndIf	
EndFunc	

$g(12)$	12
$g(-12)$	12

Define $g(x)$ =Func	
If $x < -5$ Then	
Return 5	
ElseIf $x > 5$ and $x < 0$ Then	
Return $\neg x$	
ElseIf $x \geq 0$ and $x \neq 10$ Then	
Return x	
ElseIf $x = 10$ Then	
Return 3	
EndIf	
EndFunc	

	Done
$g(-4)$	4
$g(10)$	3

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.

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

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

imag(*ValueI*) ⇒ *värde*

Ger argumentets imaginärdel.

imag(*ListI*) ⇒ *lista*

Ger en lista på elementens imaginärdelar.

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

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

imag()

Katalog > 

imag(*MatrixI*) $\Rightarrow$ *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 193.

inString()

Katalog > 

inString(*srcString*, *subString*[, *Start*]) $\Rightarrow$
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*.

$\text{inString}(\text{"Hello there"}, \text{"the"})$	7
$\text{inString}(\text{"ABCEFG"}, \text{"D"})$	0

int()

Katalog > 

int(*Value*) $\Rightarrow$ *heltal*

int(*ListI*) $\Rightarrow$ *lista*

int(*MatrixI*) $\Rightarrow$ *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
komplex tal.

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

$\text{int}(-2.5)$	-3.
$\text{int}\left(\begin{bmatrix} -1.234 & 0 & 0.37 \end{bmatrix}\right)$	$\begin{bmatrix} -2. & 0 & 0. \end{bmatrix}$

intDiv()

[Katalog >](#) 

intDiv(*Number1*, *Number2*) $\Rightarrow$ *heltal*

intDiv(*List1*, *List2*) $\Rightarrow$ *lista*

intDiv(*Matrix1*, *Matrix2*) $\Rightarrow$ *matrix*

Ger heltalsdelen med tecken av (*Number1* $\div$ *Number2*).

Ger, för listor och matriser, heltalsdelen med tecken av (*argument1* $\div$ *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*) $\Rightarrow$ *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:

$y' = -3 \cdot y + 6 \cdot t + 5$ och $y(0) = 5$

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

$xvalueList := seq(i, 0, 10, 0.5)$	
$\{0, 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6., 6.5\}$	
$xlist := mat \blacktriangleright list(rk[1])$	$\{0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.\}$
$ylist := mat \blacktriangleright list(rk[2])$	$\{5., 3.19499, 5.00394, 6.99957, 9.00593, 10.9978\}$
$yprimeList := -3 \cdot y + 6 \cdot t + 5 y=ylist \text{ och } t=xlist$	$\{-10., 1.41503, 1.98819, 2.00129, 1.98221, 2.006\}$
$interpolate(xvalueList, xlist, ylist, yprimeList)$	
$\{5., 2.67062, 3.19499, 4.02782, 5.00394, 6.00011\}$	

invχ2()

[Katalog >](#) 

invχ2(*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(*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
(*CumulativeProb*,*NumTrials*,*Prob*,
OutputForm) $\Rightarrow$ 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(*CumulativeProb*,*Prob*,
NumSuccess,*OutputForm*) $\Rightarrow$ skalär eller matris

Exempel: Monika övar målskick 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?

invBinomN()Katalog > 

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*).

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

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

OutputForm=1, visar resultatet som en matris.

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*) $\Rightarrow$ *heltal*

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

iPart(*Matrix1*) $\Rightarrow$ *matris*

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

Ger argumentets heltalsdel.

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

Argumentet kan vara ett reellt eller ett komplext tal.

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*.

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 96.

<i>list1</i> := { 6000, -8000, 2000, -3000 }	{ 6000, -8000, 2000, -3000 }
<i>list2</i> := { 2, 2, 2, 1 }	{ 2, 2, 2, 1 }
irr (5000, <i>list1</i> , <i>list2</i>)	-4.64484

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 Räkna 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 <i>nextprim</i> (<i>n</i>) = Func	Done
Loop	
<i>n</i> + 1 → <i>n</i>	
If isPrime (<i>n</i>)	
Return <i>n</i>	
EndLoop	
EndFunc	
<i>nextprim</i> (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.

<i>a</i> := _	_
isVoid (<i>a</i>)	true
isVoid ({ 1, _ , 3 })	{ false, true, false }

För mer information om tomma element, se på sidan 216.

L

Lbl

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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $g()$ = Func	Done
Local <i>temp</i> , <i>i</i>	
$0 \rightarrow temp$	
$1 \rightarrow i$	
Lbl <i>top</i>	
$temp + i \rightarrow temp$	
If $i < 10$ Then	
$i + 1 \rightarrow i$	
Goto <i>top</i>	
EndIf	
Return <i>temp</i>	
EndFunc	
$g()$	55

lcm()

lcm(*Number1*, *Number2*) $\Rightarrow$ *uttryck*

$lcm(6,9)$	18
$lcm\left(\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right)$	$\left\{\frac{2}{3}, 14, 80\right\}$

lcm(*List1*, *List2*) $\Rightarrow$ *lista***lcm**(*Matrix1*, *Matrix2*) $\Rightarrow$ *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.

left()

left(*sourceString*[, *Num*]) $\Rightarrow$ *sträng*

$left("Hello", 2)$	"He"
--------------------	------

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

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

left(*List1*[, *Num*])⇒*lista*

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.

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

libShortcut()

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*. Läger 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

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

Utför den linjära regressionsanalysen $y = a + b \cdot x$ på listorna X och Y med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a + b \cdot x$
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 <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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

Beräknar den linjära regressionen $y = m \cdot x + b$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $m \cdot x + b$
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 <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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 enskild observation och ett nivå-C-konfidensintervall för medelvärde på svaret.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a + b \cdot x$
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

Resultatvariabel	Beskrivning
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. $\hat{y}$	$a + b \cdot XVal$

LinRegtTest

Katalog > 

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

Utför en linjär regressionsanalys på listorna X och Y och ett t -test på lutnings 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 $H_a: \beta \neq 0$ och $\rho \neq 0$ (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 149.)

För information om effekten av tomma
element i en lista, se "Tomma element" (på
sidan 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a + b \cdot x$
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

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

linSolve(*LinjärEkv1* och *LinjärEkv2* och ..., *Var1*, *Var2*, ...) ⇒ lista

$$\text{linSolve}\left(\left\{\begin{array}{l} 2 \cdot x = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{array}\right\}, \{x, y\}\right) \quad \left\{\frac{3}{2}, \frac{1}{6}\right\}$$

linSolve({*LinjärEkv1*, *LinjärEkv2*, ...}, *Var1*, *Var2*, ...) ⇒ lista

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

linSolve(*SystemAvLinjäraEkv*, {*Var1*, *Var2*, ...}) ⇒ lista

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

linSolve(*LinjärEkv1* och *LinjärEkv2* och ..., {*Var1*, *Var2*, ...}) ⇒ lista

linSolve({*LinjärEkv1*, *LinjärEkv2*, ...}, {*Var1*, *Var2*, ...}) ⇒ lista

Ger en lista på lösningar för variablerna *Var1*, *Var2*, ...

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

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

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(*List* [, *elementsPerRow*])⇒*matris*

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(...)**.

list►mat({ 1,2,3 })	[1 2 3]						
list►mat({ { 1,2,3,4,5 },2 })	<table> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>5</td><td>0</td></tr> </table>	1	2	3	4	5	0
1	2						
3	4						
5	0						

ln()

  **tangenter**

ln(*Value I*)⇒*värde*

ln(2.) 0.693147

ln(*List I*)⇒*lista*

Ger argumentets naturliga logaritm.

Om det komplexa formatläget är Real:

Ger, för en lista, elementens naturliga logaritmer.

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(*squareMatrix I*)⇒*kvadratMatris*

Ger matrisen med naturlig logaritm för *squareMatrix I*. Detta är inte detsamma som att beräkna den naturliga logaritmen för varje element. För information om beräkningsmetoden, se **cos()**.

squareMatrix I 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·i
-0.266891-2.08316·i	1.12436+1.79018·i

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

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

Utför en en logaritmisk regressionsanalys $y = a + b \cdot \ln(x)$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a + b \cdot \ln(x)$
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 <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>

Resultatvariabel	Beskrivning
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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.

Obs för att mata in exemplet: Se avsnittet Räkna 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

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 167 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

$$\log_{10} (2.) \quad 0.30103$$

log(*List1*[,*Value2*])⇒lista

$$\log_4 (2.) \quad 0.5$$

Ger bas-*Value2*-logaritmen för det första argumentet.

$$\log_3 (10) - \log_3 (5) \quad 0.63093$$

Obs: Se även **Log template**, på sidan 2.

Ger, för en lista, bas-*Value2*-logaritmen för elementen.

Om det komplexa formatläget är Real:

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

"Error: Non-real calculation"

Om det andra argumentet utelämnas används 10 som bas.

Om det komplexa formatläget är Rectangular:

$$\log_{10} (\{-3, 1.2, 5\})$$
$$\{0.477121 + 1.36438 \cdot i, 0.079181, 0.69897\}$$

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.

I vinkelläget Radianer och i Rektangulärt komplext format:

$$\log_{10} \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 0.795387 + 0.753438 \cdot i & 0.003993 - 0.64747 \cdot i \\ 0.194895 - 0.315095 \cdot i & 0.462485 + 0.27079 \cdot i \\ -0.115909 - 0.904706 \cdot i & 0.488304 + 0.77746 \cdot i \end{bmatrix}$$

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

Om basargumentet utelämnas används 10 som bas.

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 regressionsanalysen $y = (c / (1 + a \cdot e^{-bx}))$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

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 <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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 149.)

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 .

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 216).

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 <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>

Resultatvariabel	Beskrivning
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

Loop

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 Räkna-re i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Katalog > 

```

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

```

Done

rollcount()

16

rollcount()

3

LU

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*.

$lMatris \cdot uMatris = pMatris \cdot matrix$

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*.

Katalog > 

6

12

18

5

14

31

3

8

18

$\rightarrow m1$

6

12

18

5

14

31

3

8

18

LU *m1*,*lower*,*upper*,*perm*

lower

upper

perm

1

0

0

$\frac{5}{6}$

1

0

$\frac{1}{2}$

$\frac{1}{2}$

1

6

12

18

0

4

16

0

0

1

1

0

0

0

1

0

0

0

1

- Om du använder   eller ställer in

Alfabetisk lista 91

Auto eller Ungefärlig på Approximate utförs beräkningarna med flyttalsaritmetik.

- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som:
 $5E-14 \cdot \max(\dim(Matrix)) \cdot \text{rowNorm}(Matrix)$

Faktoriseringsalgoritmen **LU** använder partiell pivotering med radutbyten.

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 ($\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$)	$\{1, 2, 3\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
mat▶list (<i>m1</i>)	$\{1, 2, 3, 4, 5, 6\}$

max()

Katalog > 

max(*Value1*, *Value2*)⇒*uttryck*

max (2.3, 1.4)	2.3
-----------------------	-----

max(*List1*, *List2*)⇒*lista*

max ($\{1, 2\}, \{-4, 3\}$)	$\{1, 3\}$
--------------------------------------	------------

max(*Matrix1*, *Matrix2*)⇒*matrix*

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(*List*)⇒*uttryck*

max ($\{0, 1, -7, 1.3, 0.5\}$)	1.3
---	-----

Ger maximelementet i *list*.

max(*Matrix1*)⇒*matrix*

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

Ger en radvektor som innehåller maximelementet för varje kolumn i *Matrix1*.

max()Katalog > 

Tomma element ignoreras. För mer information om tomma element, se på sidan 216.

Obs: Se även **min()**.

mean()Katalog > 

mean(List[,freqList])⇒*uttryck*

Ger medelvärde för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

mean(MatrixI[,freqMatrix])⇒*matrix*

Ger en radvektor med medelvärdena för alla kolumner i *MatrixI*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *MatrixI*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 216.

$\text{mean}(\{0.2, 0.1, -0.3, 0.4\})$	0.26
$\text{mean}(\{1, 2, 3\}, \{3, 2, 1\})$	$\frac{5}{3}$

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*.

median(MatrixI[,freqMatrix])⇒*matrix*

Ger en radvektor som innehåller medianerna för kolumnerna i *MatrixI*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *MatrixI*.

$\text{median}(\{0.2, 0.1, -0.3, 0.4\})$	0.2
--	-----

$\text{median}\left(\begin{bmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{bmatrix}\right)$	$\begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$
---	--

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 216.

MedMed

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

Beräknar median-median-linjen $y = (m \cdot x + b)$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Ekvation för median-median-linje $m \cdot x + b$
stat.m, stat.b	Modellkoefficienter
stat.Resid	Residualer från median-median-linjen

Resultatvariabel	Beskrivning
stat.XReg	Lista på datapunkter i den modifierade <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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(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(sourceStringList, Start[, Count])⇒lista

Ger *Count*-strängarna från stränglistan *sourceStringList* och börjar med elementnumret *Start*.

mid("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	" "

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{ }

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

min()

Katalog > 

min(Value1, Value2)⇒uttryck

min(List1, List2)⇒lista

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

min(*Matrix1*, *Matrix2*) $\Rightarrow$ *matrix*

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(*List*) $\Rightarrow$ *uttryck*

Ger minimelementet för *List*.

min(*Matrix1*) $\Rightarrow$ *matrix*

Ger en radvektor som innehåller minimelementet för varje kolumn i *Matrix1*.

Obs: Se även **max()**.

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

$$\min\left(\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}\right) \quad \begin{bmatrix} -4 & -3 & 0.3 \end{bmatrix}$$

mirr()

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.

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 76.

$$\begin{aligned} list1 &:= \{6000, -8000, 2000, -3000\} \\ &\quad \{6000, -8000, 2000, -3000\} \\ list2 &:= \{2, 2, 2, 1\} & \{2, 2, 2, 1\} \\ mirr(4.65, 12, 5000, list1, list2) & \quad 13.41608607 \end{aligned}$$

mod()Katalog > **mod**(*Value1*, *Value2*) $\Rightarrow$ uttryck $\text{mod}(7,0)$ 7**mod**(*List1*, *List2*) $\Rightarrow$ lista $\text{mod}(7,3)$ 1**mod**(*Matrix1*, *Matrix2*) $\Rightarrow$ matris $\text{mod}(-7,3)$ 2

Ger det första argumentet modulo det andra argumentet definierat av identiteterna:

 $\text{mod}(7,-3)$ -2 $\text{mod}(-7,-3)$ -1 $\text{mod}(\{12,-14,16\},\{9,7,-5\})$ $\{3,0,-4\}$ $\text{mod}(x,0) = x$ $\text{mod}(x,y) = x - y \text{ floor}(x/y)$

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 127

mRow()Katalog > **mRow**(*Value*, *Matrix1*, *Index*) $\Rightarrow$ matris

Ger en kopia av *Matrix1* med varje element i rad *Index* i *Matrix1* multiplicerat med *Value*.

$$\text{mRow}\left(\frac{-1}{3}, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 2\right) \quad \begin{bmatrix} 1 & 2 \\ -1 & \frac{-4}{3} \end{bmatrix}$$
mRowAdd()Katalog > **mRowAdd**(*Value*, *Matrix1*, *Index1*, *Index2*) $\Rightarrow$ matris

Ger en kopia av *Matrix1* med varje element i rad *Index2* i *Matrix1* ersatt med:

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

$\text{Value} \cdot \text{rad Index1} + \text{rad Index2}$

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

Beräknar den multipla linjära regressionen i lista Y på listorna $X1, X2, \dots, X10$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.b0, stat.b1, ...	Regressionskoefficienter
stat.R ²	Koefficient för multipel bestämning
stat. $\hat{y}$ List	$\hat{y}$ List = $b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Residualer från regressionsanalysen

MultRegIntervals

MultRegIntervals $Y, X1[,X2,X3,\dots$
 $[,X10]]], XValList[,CLevel]$

Beräknar ett prognostiserat y -värde, ett nivå- C -prediktionsintervall för en enskilda observation och ett nivå- C -konfidensintervall för medelvärde på svaret.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat. $\hat{y}$	En punktu uppskattning: $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ för <i>XValList</i>
stat.dfError	Fel hos frihetsgrader
stat.CLower, stat.CUpper	Konfidensintervall för ett medelvärde på svaret

Resultatvariabel	Beskrivning
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelvärdet på svaret
stat.LowerPred, stat.UpperrPred	Prediktionsintervall för en enskilda observation
stat.MEPred	Prediktionsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.bList	Lista på regressionskoefficienter, $\{b_0, b_1, b_3, \dots\}$
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 149.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Utdata

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.F	Global F -teststatistik
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

Resultatvariabel	Beskrivning
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.yList	$\hat{y}$ List = $b_0 + b_1 \cdot x_1 + \dots$
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

  **knappar**

BoolesktUttr1 **nand** *BoolesktUttr2* ger
Booleskt uttryck

BooleskLista1 **nand** *BooleskLista2* ger
Boolesk lista

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* $\Rightarrow$ *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*) $\Rightarrow$ *uttryck*

För heltal *Value1* och *Value2* med *Value1* $\geq$ *Value2* $\geq$ 0 är **nCr()** antalet kombinationer av *Value1* tagna *Value2* åt gången. (Detta kallas också en binomial koefficient.)

$\text{nCr}(z,3) _{z=5}$	10
$\text{nCr}(z,3) _{z=6}$	20

nCr(*Value*, 0) $\Rightarrow$ 1

nCr(*Value*, *negInteger*) $\Rightarrow$ 0

nCr(*Value*, *posInteger*) $\Rightarrow$ *Value* ·
(*Value* - 1) ...
(*Value* - *posInteger* + 1) / *posInteger*!

nCr(*Value*, *nonInteger*) $\Rightarrow$ *expression*! /
(*Value* - *nonInteger*)! · *nonInteger*!

nCr(*List1*, *List2*) $\Rightarrow$ *lista*

Ger en lista på kombinationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.

$\text{nCr}(\{5,4,3\}, \{2,4,2\})$	$\{10,1,3\}$
------------------------------------	--------------

nCr(*Matrix1*, *Matrix2*) $\Rightarrow$ *matrix*

Ger en matris över kombinationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.

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

nDerivative()Katalog > 

nDerivative(*Uttr1*, *Var*=*Värde*
[,*Ordning*]) $\Rightarrow$ *värde*

nDerivative(*Uttr1*, *Var*[,*Ordning*]) |
Var=*Värde* $\Rightarrow$ *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**.

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 $x \cdot (x^2 + x)^{1/3}$ 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.

Om du stöter på denna begränsning, verifiera lösningen grafiskt. Du kan också prova att använda **centralDiff()**.

$nDerivative(x , x=1)$	1
$nDerivative(x , x) _{x=0}$	undef
$nDerivative(\sqrt{x-1}, x) _{x=1}$	undef

$nDerivative\left(x \cdot (x^2 + x)^{\frac{1}{3}}, x, 1\right) _{x=0}$	undef
$centralDiff\left[x \cdot (x^2 + x)^{\frac{1}{3}}, x\right] _{x=0}$	0.000033

newList()Katalog > 

newList(*numElements*) $\Rightarrow$ *lista*

Ger en lista med dimensionen på *numElements*. Varje element är noll.

$newList(4)$	{0,0,0,0}
--------------	-----------

newMat()Katalog > **newMat(numRows, numColumns)**⇒*matris*

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

Ger en matris med nollor med dimensionen *numRows* gånger *numColumns*.

nfMax()Katalog > **nfMax(Expr, Var)**⇒*värde*

nfMax $\left(-x^2-2\cdot x-1,x\right)$	-1.
---	-----

nfMax(Expr, Var, undrGräns)⇒*värde*

nfMax $\left(0.5\cdot x^3-x-2,x,-5,5\right)$	5.
---	----

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.

nfMin()Katalog > **nfMin(Expr, Var)**⇒*värde*

nfMin $\left(x^2+2\cdot x+5,x\right)$	-1.
--	-----

nfMin(Expr, Var, undrGräns)⇒*värde*

nfMin $\left(0.5\cdot x^3-x-2,x,-5,5\right)$	-5.
---	-----

nfMin(Expr, Var, undrGräns, övrGräns)⇒*värde***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*) $\Rightarrow$ uttryck

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

Om integranden *Expr1* inte innehåller någon variabel utöver *Var*, och om *Lower* och *Upper* är konstanter, positiv ∞ eller negativ ∞ , ger **nInt()** en uppskattning av $\int$ (*Expr1*, *Var*, *Lower*, *Upper*). Denna uppskattning är ett vägt genomsnitt av vissa sampelvärden hos integranden i intervallet $\text{Lower} < \text{Var} < \text{Upper}$.

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.

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

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.

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

nom()

Katalog > 

nom(*effectiveRate*, *CpY*) $\Rightarrow$ värde

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

Finansiell funktion som konverterar den årliga effektiva räntan *effectiveRate* till en nominell ränta, given av *CpY* som antalet sammansatta ränteperioder per år.

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

Obs: Se även **eff()**, på sidan 45.

nor

  **knappar**

BoolesktUttr1 **nor** *BoolesktUttr2* 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* $\Rightarrow$ *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*) $\Rightarrow$ *uttryck*

$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	5.47723
--	---------

norm(*Vector*) $\Rightarrow$ *uttryck*

$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	2.23607
---	---------

Ger Frobenius norm.

$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	2.23607
--	---------

normCdf()

Katalog > 

normCdf(*lowBound*, *upBound*, μ , σ) $\Rightarrow$ *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 $P(X \leq upBound)$, sätt $lowBound = -9E999$.

$normPdf(XVal[, \mu[, \sigma]]) \Rightarrow tal$ 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 *BooleanExpr* $\Rightarrow$ Booleskt uttryck

Ger en sann, falsk eller förenklad form av argumentet.

not *Integer1* $\Rightarrow$ 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.

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).

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.

not {2≥3}	true
not 0hB0►Base16	0hFFFFFFFFFFFFFFF4F
not not 2	2

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

not 0h7AC36	0hFFFFFFFFFFFF853C9
-------------	---------------------

I binärt basläge:

0b100101►Base10	37
not 0b100101	
0b11111111111111111111111111111111►	
not 0b100101►Base10	-38

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

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	$nPr(z,3) z=5$	60
För heltal <i>Value1</i> och <i>Value2</i> med <i>Value1</i> ≥ <i>Value2</i> ≥ 0 är nPr() antalet permutationer av <i>Value1</i> saker tagna <i>Value2</i> åt gången.	$nPr(z,3) z=6$	120
	$nPr(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$
	$nPr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$
nPr(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	$nPr(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$
Ger en lista på permutationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.		
nPr(Matrix1, Matrix2)⇒matrix	$nPr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$
Ger en matrix över permutationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.		

npv()		Katalog > 
npv(InterestRate,CFO,CFList[,CFFreq])	$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
Finansiell funktion som beräknar nettovärdet, dvs. summan av aktuella värden för kassainflöden och kassautflöden. Ett positivt resultat för npv indikerar en vinstgivande investering.	$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
	$npv(10,5000,list1,list2)$	4769.91
<i>InterestRate</i> är räntan med vilken kassaflödena (kapitalanskaffningskostnaderna) diskonteras under en period.		
<i>CFO</i> är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.		
<i>CFList</i> är en lista på kassaflödesbelopp efter det initiala kassaflödet <i>CFO</i> .		

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()

nSolve(*Equation*,*Var*[=*Guess*]) $\Rightarrow$ *tal* eller *fel_sträng*

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x)$	3.84429
--	---------

nSolve(*Equation*,*Var*[=*Guess*],*lowBound*) $\Rightarrow$ *tal* eller *fel_sträng*

$\text{nSolve}(x^2 = 4, x = -1)$	-2.
----------------------------------	-----

$\text{nSolve}(x^2 = 4, x = 1)$	2.
---------------------------------	----

nSolve(*Equation*,*Var*[=*Guess*],*lowBound*,*upBound*) $\Rightarrow$ *tal* eller *fel_sträng*

Obs: Om det finns flera lösningar kan du använda en gissning för att lättare hitta en viss lösning.

nSolve(*Equation*,*Var*[=*Guess*]) | *lowBound* $\leq$ *Var* $\leq$ *upBound* $\Rightarrow$ *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.

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".

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x) x < 0$	-8.84429
--	----------

$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r} = 26, r\right) r > 0 \text{ and } r < 0.25$	0.006886
--	----------

$\text{nSolve}(x^2 = -1, x)$	"No solution found"
------------------------------	---------------------

O

OneVar

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 149.)

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 216.

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

Resultatvariabel	Beskrivning
stat.Q ₁ X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q ₃ X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.SSX	Kvadratsumma av avvikelser från medelvärdet på x

or (eller)

Katalog > 

BoolesktUtr1 **or** *BoolesktUtr2* 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.

Integer1 **or** *Integer2* $\Rightarrow$ *heltal*

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.

```
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

I hexadecimalt basläge:

0h7AC36 or 0h3D5F	0h7BD7F
-------------------	---------

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.

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**.

ord()

ord(*String*) $\Rightarrow$ *integer*

ord(*List l*) $\Rightarrow$ *lista*

Ger den numeriska koden för det första tecknet i teckensträngen *String* eller en lista på de första tecknen i varje listelement.

<code>ord("hello")</code>	104
<code>char{104}</code>	"h"
<code>ord(char{24})</code>	24
<code>ord({"alpha", "beta"})</code>	{ 97, 98 }

P

P►Rx()

P►Rx(*rExpr*, *θExpr*) $\Rightarrow$ *uttryck*

I vinkelläget Radianer:

P►Rx(*rList*, *θList*) $\Rightarrow$ *lista*

`P►Rx{4,60°}` 2.

P►Rx(*rMatrix*, *θMatrix*) $\Rightarrow$ *matrix*

`P►Rx{{-3,10,1.3},{π/3,π/4,0}}`
 { -1.5, 7.07107, 1.3 }

Ger den ekvivalenta x-koordinaten för paret (*r*, *θ*).

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(*rValue*, *θValue*)⇒*värde*

I vinkelläget Radianer:

P►Ry(*rList*, *θList*)⇒*lista*

P►Ry($\{4, 60^\circ\}$) 3.4641

P►Ry(*rMatrix*, *θMatrix*)⇒*matrix*

P►Ry $\left(\left\{-3, 10, 1.3\right\}, \left\{\frac{\pi}{3}, \frac{\pi}{4}, 0\right\}\right)$
 $\{-2.59808, -7.07107, 0\}$

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 (...)**.

PassErr

PassErr

För ett exempel på **PassErr**, se exempel 2 under kommandot **Try**, på sidan 161.

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.

Obs: Se även **ClrErr**, på sidan 22 och **Try**, på sidan 160.

Kommentarer om inmatningen av exemplet: I applikationen Räkna kan du skriva in flerradiga definitioner genom att trycka på  i stället för  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 $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$	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 $P(0 \leq X \leq \text{upBound}) \Rightarrow$ 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 $P(X \leq \text{upBound})$, 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

$\begin{bmatrix} 1 & 3. \end{bmatrix}$ ►Polar	$\begin{bmatrix} 3.16228 & \angle 71.5651 \end{bmatrix}$
---	--

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

Visar *vector* i polär form [$r \angle \theta$]. 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 124.

complexValue ►Polar

Visar *complexVector* i polär form.

- Vinkelläget Grader ger $(r \angle \theta)$.
- 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 $(r \angle \theta)$ polär inmatning.

I vinkelläget Radianer:

$(3+4 \cdot i)$ ►Polar	$e^{0.927295 \cdot i} \cdot 5$
$\left(4 \angle \frac{\pi}{3}\right)$ ►Polar	$e^{1.0472 \cdot i} \cdot 4$

I vinkelläget Nygrader:

$(4 \cdot i)$ ►Polar	$(4 \angle 100.)$
----------------------	-------------------

I vinkelläget Grader:

$(3+4 \cdot i)$ ►Polar	$(5 \angle 53.1301)$
------------------------	----------------------

polyEval()

polyEval(List1, Expr1)⇒uttryck

$\text{polyEval}(\{1, 2, 3, 4\}, 2)$	26
--------------------------------------	----

polyEval(List1, List2)⇒uttryck

$\text{polyEval}(\{1, 2, 3, 4\}, \{2, -7\})$	$\{26, -262\}$
--	----------------

Tolkar det första argumentet som koefficienten för ett polynom med fallande ordning och ger polynomet utvärderat för det andra argumentets värde.

polyRoots()

polyRoots(Poly, Var)⇒lista

$\text{polyRoots}(y^3+1, y)$	$\{-1\}$
------------------------------	----------

polyRoots(ListaPåKoeff)⇒lista

$\text{cPolyRoots}(y^3+1, y)$	$\{-1, 0.5-0.866025 \cdot i, 0.5+0.866025 \cdot i\}$
-------------------------------	--

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: $\{ \}$.

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

$\text{polyRoots}(\{1, 2, 1\})$	$\{-1, -1\}$
---------------------------------	--------------

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.

PowerReg

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

Utför potensregressionsanalys $y = (a \cdot (x)^b)^p$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot (x)^b$

Resultatvariabel	Beskrivning
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 <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

Prgm

Katalog >

Prgm

Beräkna GCD och visa mellanliggande resultat.

Block

Define *proggcd(a,b)*=Prgm

EndPrgm

Local *d*
While *b*≠0
d:=mod(*a*,*b*)
a:=*b*
b:=*d*
Disp *a*," ",*b*
EndWhile
Disp "GCD=",*a*
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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Done

proggcd(4560,450)

450 60
60 30
30 0
GCD=30

Done

product()

Katalog > **product(List[, Start[, end]])** ⇒ *uttryck*

Ger produkten av elementen i *List*. *Start* och *end* är valfria. De specificerar ett område med element.

$$\text{product}(\{1, 2, 3, 4\}) \quad 24$$

$$\text{product}(\{4, 5, 8, 9\}, 2, 3) \quad 40$$

product(MatrixI[, Start[, end]]) ⇒ *matris*

Ger en radvektor som innehåller produkterna av elementen i kolumnerna i *MatrixI*. *Start* och *end* är valfria. De specificerar ett område med rader.

$$\text{product}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) \quad [28 \ 80 \ 162]$$

$$\text{product}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, 1, 2\right) \quad [4 \ 10 \ 18]$$

Tomma element ignoreras. För mer information om tomma element, se på sidan 216.

propFrac()

Katalog > **propFrac(ValueI[, 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.

$$\text{propFrac}\left(\frac{4}{3}\right) \quad 1 + \frac{1}{3}$$

$$\text{propFrac}\left(\frac{-4}{3}\right) \quad -1 - \frac{1}{3}$$

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.

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   eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som:
 $5E-14 \cdot \max(\dim(\text{Matrix})) \cdot \text{rowNorm}(\text{Matrix})$

QR-faktoriseringen beräknas numeriskt med Householder-transformationer. Den symboliska lösningen beräknas med Gram-Schmidt. Kolumnerna i *qMatName* är de ortonormerade basvektorer 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 kvadratiske regressionsanalysen $y = a \cdot x^2 + b \cdot x + c$ på listorna X och Y med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216.

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^2 + b \cdot x + c$
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 <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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

Utför en fjärdegrads regressionsanalys $y = a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoefficienter
stat.R ²	Determinationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

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 $\left(\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right)$	$\begin{bmatrix} 0. & 2.94771 & 0.643501 \end{bmatrix}$

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 $\left(\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right)$	$\begin{bmatrix} 3 & 4.07638 & \frac{5}{2} \end{bmatrix}$

►Rad**Katalog >** *Value*►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*

Bestämmer slumpvalsfröet.

rand() ger ett slumpvärde mellan 0 och 1.

rand(#Trials) ger en lista med *#Trials* slumpvärden mellan 0 och 1.

RandSeed 1147

Done

rand(2)

{ 0.158206, 0.717917 }

slumpBin()

Katalog > 

randBin(*n*, *p*) ⇒ *uttryck*

randBin(*n*, *p*, #Trials) ⇒ *lista*

randBin(80,0.5)

46.

randBin(80,0.5,3)

{ 43., 39., 41. }

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.

slumpBin()

Katalog > 

randInt

(*lowBound*, *upBound*)

⇒ *uttryck*

randInt

(*lowBound*, *upBound*

, #Trials) ⇒ *lista*

randInt(3,10)

3.

randInt(3,10,4)

{ 9., 3., 4., 7. }

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.

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 .

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 $[\mu-3\cdot\sigma, \mu+3\cdot\sigma]$.

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 slumpalsgeneratorn. 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(ValueI) ⇒ värde

$\text{real}(2+3 \cdot i)$	2
----------------------------	---

Ger argumentets reella del.

real(ListI) ⇒ lista

$\text{real}(\{1+3 \cdot i, 3, i\})$	$\{1, 3, 0\}$
--------------------------------------	---------------

Ger de reella delarna av alla element.

real(MatrixI) ⇒ matris

$\text{real}\left(\begin{bmatrix} 1+3 \cdot i & 3 \\ 2 & i \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$
--	--

Ger de reella delarna av alla element.

► Rect

Katalog > 

Vector ► Rect

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

$\left(\begin{bmatrix} 3 & \angle \frac{\pi}{4} & \angle \frac{\pi}{6} \end{bmatrix}\right) \blacktriangleright \text{Rect}$	$[1.06066 \quad 1.06066 \quad 2.59808]$
--	---

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.

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 ∠ θ) polär inmatning.

Med vinkelenhet Radianer:

$\left(4 \cdot e^{\frac{\pi}{3}}\right) \blacktriangleright \text{Rect}$	11.3986
$\left(4 \angle \frac{\pi}{3}\right) \blacktriangleright \text{Rect}$	2.+3.4641·i

Med vinkelenhet Nygrader:

$$\left((1 \angle 100) \right) \blacktriangleright \text{Rect} \quad i$$

Med vinkelmått Grader:

$$\left((4 \angle 60) \right) \blacktriangleright \text{Rect} \quad 2.+3.4641 \cdot i$$

Obs: För att skriva in tecknet $\angle$, välj det från symbollistan i Katalog.

ref()
 $\text{ref}(\text{Matrix}I[, Tol]) \Rightarrow \text{matrix}$
Ger radtrappstegsformen av *MatrixI*.

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   eller ställer in **Auto or Approximate** på Approximate, utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som:
 $5E-14 \cdot \max(\dim(\text{Matrix}I)) \cdot \text{rowNorm}(\text{Matrix}I)$

Undvik odefinierade element i *MatrixI*. De kan leda till oväntade resultat.

Om till exempel *a* är odefinierat i följande uttryck visas ett varningsmeddelande och resultatet ges som:

$$\text{ref} \left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{ref} \left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

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 ("|")-operatoren begränsning för att ersätta ett värde såsom visas i följande exempel.

$$\text{ref} \left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \middle| a=0 \right) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Obs: Se även **rref ()**, på sidan 134.

RefreshProbeVars

RefreshProbeVars

Låter dig visa sensordata från alla anslutna sensorer via TI-grundprogrammet.

StatusVar-värde	Status
<i>statusVar</i> =0	Normal (fortsätt med programmet) Vernier DataQuest™-applikationen är i datainsamlingsläge.
<i>statusVar</i> =1	Obs: Vernier DataQuest™-applikationen måste vara i läge "meter" för att detta kommando ska fungera. 
<i>statusVar</i> =2	Vernier DataQuest™-applikationen har inte startats.
<i>statusVar</i> =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 temperatur>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()Katalog >

remain(Value1, Value2) ⇒ värde
remain(List1, List2) ⇒ lista
remain(Matrix1, Matrix2) ⇒ matrix

Ger resten på det första argumentet med avseende på det andra argumentet definierat av identiteterna:

$\text{remain}(x,0) = x$
 $\text{remain}(x,y) = x - y \cdot \text{iPart}(x/y)$

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 97.

$\text{remain}(7,0)$	7
$\text{remain}(7,3)$	1
$\text{remain}(-7,3)$	-1
$\text{remain}(7,-3)$	1
$\text{remain}(-7,-3)$	-1
$\text{remain}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,1\}$

$\text{remain}\left(\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$
--	---

RequestKatalog >

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 *vardet* värdet 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**:

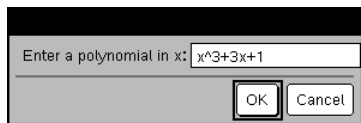
Radie: 6/2
Area= 28,2743

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: $\{-0,322185\}$

För att stoppa ett program som innehåller ett **Request**-kommando inne i en oändlig slinga:

- **Handenhet:** Håll ned  och tryck på  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 129.

RequestStr

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  och tryck på  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 127.

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

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

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

right(*Comparison*) ⇒ *uttryck*

Ger den högra sidan av en ekvation eller olikhet.

right({1,3,-2,4},3) {3,-2,4}

right("Hello",2) "lo"

rk23 ()

Katalog > 

rk23(*Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}, *depVar0*, *VarStep* [, *dif_tol*]) ⇒ *matrix*

rk23(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *dif_tol*]) ⇒ *matrix*

rk23(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *dif_tol*]) ⇒ *matrix*

Differentialekvation:

$y' = 0,001 \cdot y \cdot (100 - y)$ 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

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

med $\text{depVar}(\text{Var}0) = \text{depVar}0$ i intervallet $[\text{Var}0, \text{VarMax}]$. 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.

$\{\text{Var}0, \text{VarMax}\}$ 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 $\text{sign}(\text{VarStep}) = \text{sign}(\text{VarMax} - \text{Var}0)$ och lösningar vid $\text{Var}0 + i * \text{VarStep}$ för alla $i=0,1,2,\dots$ sådana att $\text{Var}0 + i * \text{VarStep}$ är i intervallet $[\text{Var}0, \text{VarMax}]$ (kanske inte ger ett lösningsvärde vid *VarMax*).

Om *VarStep* utvärderas till noll ges lösningar vid "Runge-Kutta"-värdena för *Var*.

diftol är feltoleransen (föreställs på 0,001).

Samma ekvation med *diftol* inställd på $1.E-6$

rk23($\{0.001 \cdot y \cdot \{100 - y\}, t, y, \{0, 100\}, 10, 1, 1.E-6\}$)				
0.	1.	2.	3.	4.
10.	10.9367	11.9495	13.0423	14.2189

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 $y2(0) = 5$

rk23($\left\{ \begin{cases} -y1 + 0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1 \right\}$)				
0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245

$$\text{root}(Value) \Rightarrow root$$
$$\text{root}(Value1, Value2) \Rightarrow \text{root}$$
$$\sqrt[3]{8} \qquad 2$$

`root(Value)` ger kvadratroten ur *Value*.

$\sqrt[3]{3}$	1.44225
---------------	---------

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.

$$\text{rotate}(\text{Integer } l[, \# \text{of Rotations}]) \Rightarrow \text{heltal}$$

I binärt basläge:

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.

rotate(0b11111111111111111111111111111111)	0b100000000000000000000000000000001
rotate(256.1)	0b1000000000

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

Om `#ofRotations` är positiv sker rotationen åt vänster. Om `#ofRotations` är negativ sker rotationen åt höger. Förinställningen är `-1` (rotera en position åt höger).

I hexadecimalt basläge:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

Vid exempelvis rotation åt höger:

Varje bit roteras åt höger.

```
0b0000000000000111010110000110101
```

Biten längst till höger roteras till positionen längst till vänster.

ger:

```
0b10000000000000011101011000011010
```

Resultatet visas enligt det inställda basläget.

Viktigt: För att skriva in ett binärt eller hexadecimalt tal, använd alltid prefixet 0b eller 0h (noll, inte bokstaven O).

$$\text{rotate}(\text{List1}, \#ofRotations) \Rightarrow \text{lista}$$

I decimalt basläge:

Ger en kopia av *List1* roterad åt höger eller vänster av *#ofRotations*-elementen. Ändrar inte *List1*.

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

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(StringI[,#ofRotations]) $\Rightarrow$ *sträng*

Ger en kopia av *StringI* roterad åt höger eller vänster av *#ofRotations*-tecknen. Ändrar inte *StringI*.

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(ValueI[, digits]) $\Rightarrow$ *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(ListI[, digits]) $\Rightarrow$ *lista*

Ger en lista på elementen avrundade till det specificerade antalet siffror.

round(MatrixI[, digits]) $\Rightarrow$ *matris*

Ger en matris över elementen avrundade till det specificerade antalet siffror.

round(1.234567,3)	1.235
-------------------	-------

round($\{\pi, \sqrt{2}, \ln(2)\}, 4$)	$\{3.1416, 1.4142, 0.6931\}$
---	------------------------------

round($\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}, 1$)	$\begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$
--	--

rowAdd()

Katalog > 

rowAdd(MatrixI, rIndex1, rIndex2) $\Rightarrow$ *matris*

Ger en kopia av *MatrixI* med rad *rIndex2* ersatt av summan av raderna *rIndex1* och *rIndex2*.

rowAdd($\begin{bmatrix} 3 & 4 \\ -3 & -2 \end{bmatrix}, 1, 2$)	$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$
--	--

rowDim()Katalog > **rowDim(Matrix)** $\Rightarrow$ uttryckGer antalet rader i *Matrix*.**Obs:** Se även **colDim()**, på sidan 23.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
<code>rowDim(m1)</code>	3

rowNorm()Katalog > **rowNorm(Matrix)** $\Rightarrow$ uttryckGer maximum av summorna av absolutbeloppen på elementen i raderna i *Matrix*.**Obs:** Alla matriselement måste förenklas till tal. Se även **colNorm()**, på sidan 23.

<code>rowNorm</code> $\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right)$	25
---	----

rowSwap()Katalog > **rowSwap(Matrix1, rIndex1, rIndex2)** $\Rightarrow$ *matrix*Ger *Matrix1* med raderna *rIndex1* och *rIndex2* växlade.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
<code>rowSwap(mat,1,3)</code>	$\begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$

rref()Katalog > **rref(Matrix1[, Tol])** $\Rightarrow$ *matrix*Ger den reducerade radtrappstegsformen av *Matrix1*.

$\text{rref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$
--	---

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  eller ställer in **Auto or Approximate** på Approximate, utförs beräkningarna med

flyttalsaritmetik.

- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som:
 $5E-14 \cdot \max(\dim(\text{Matrix } I)) \cdot \text{rowNorm}(\text{Matrix } I)$

Obs: Se även **ref()**, på sidan 125.

S

sec()

 **tangent**

sec(Value1) ⇒ värde

I vinkelläget Grader:

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 π för att tillfälligt överstyra vinkelläget.

$\sec(45)$	1.41421
$\sec(\{1, 2, 3, 4\})$	$\{1.00015, 1.00081, 1.00244\}$

$\sec^{-1}()$

 **tangent**

**$\sec^{-1}(\text{Value1}) \Rightarrow$
värde**

I vinkelläget Grader:

$\sec^{-1}(\text{List1}) \Rightarrow$ 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.

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

I vinkelläget Nygrader:

$\sec^{-1}(\sqrt{2})$	50.
-----------------------	-----

I vinkelläget Radianer:

$\sec^{-1}(\{1, 2, 5\})$	$\{0, 1.0472, 1.36944\}$
--------------------------	--------------------------

sec⁻¹()

 **tangent**

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

sech()

Katalog > 

sech(Value1) ⇒ värde

sech(3) 0.099328

sech(List1) ⇒ lista

sech({1,2,3,4})
{0.648054,0.198522,0.036619}

Ger den hyperboliska sekansfunktionen för *Value1* eller en lista på de hyperboliska sekansfunktionerna för elementen i *List1*.

sech⁻¹()

Katalog > 

sech⁻¹(Value1) ⇒ värde

I vinkelläget Radianer och i Rektangulärt komplext läge:

sech⁻¹(List1) ⇒ lista

sech⁻¹(1) 0

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

sech⁻¹({1,-2,2,1})
{0,2.0944-i,8.E-15+1.07448-i}

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

Send

Hubb-meny

SendUttrEllerSträng1 [, UttrEllerSträng2] ...

Exempel: Slå på den blå komponenten i den inbyggda RGB-lysdioden i 0,5 sekunder.

Programmeringskommando: Skickar ett eller flera TI-Innovator™ Hub kommandon till en ansluten hubb.

Send "SET COLOR.BLUE ON TIME .5"

Done

exprOrString måste vara ett giltigt TI-Innovator™ Hub Kommando. *exprOrString* innehåller vanligen ett "SET ..." -kommando för att styra en enhet eller ett "READ ..." -kommando för att begära data.

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 <i>lightval</i>	Done
<i>lightval</i>	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
<i>iostr.SendAns</i>	"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.

$\text{seq}\left(n^2, n, 1, 6\right)$	$\{1, 4, 9, 16, 25, 36\}$
$\text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	$\frac{1968329}{1270080}$

Obs: För att få ett närmevärde,

Handenhet: Tryck på  .

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 $u(n) = u(n-1)^2/2$, med $u(1)=2$ och $VarStep=1$.

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.

seqGen($ListOrSystemOfExpr$, Var , $ListOfDepVars$, $\{Var0, VarMax\}$ [, $MatrixOfInitTerms$ [, $VarStep$ [, $CeilingValue$]]]) $\Rightarrow$ *matrix*

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.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1, 5\}, \{2\}\right) \\ \left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

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:

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u_2(n-1)}{2} + u_1(n-1)\right\}, n, \{u_1, u_2\}, \{1, 5\}, \left[\frac{-}{2}\right]\right) \\ \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Obs: Tomrummet () i den initiala termmatrisen ovan används för att indikera att den initiala termen för $u_1(n)$ beräknas med den explicita talföljdsformeln $u_1(n)=1/n$.

seqn()

seqn($Expr(u, n$ [, $ListOfInitTerms$ [, $nMax$ [, $CeilingValue$]]]) $\Rightarrow$ *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($Expr(n$ [, $nMax$ [, $CeilingValue$]]]) $\Rightarrow$ *lista*

Genererar de första 6 termerna i talföljden $u(n) = u(n-1)/2$, med $u(1)=2$.

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right) \\ \left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right) \\ \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

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()

`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 <i>prog1()</i> =Prgm	Done
Disp π	
setMode(1,16)	
Disp π	
EndPrgm	
<i>prog1()</i>	
	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-hetal	Hetal 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(*Integer* *I* [, *#ofShifts*]) $\Rightarrow$ *heltal*

Skiftar bitarna i ett binärt heltal. Du kan skriva in *Integer* *I* i valfri talbas. Det omvandlas automatiskt till 64-bitars binär form. Om storleken på *Integer* *I* 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:

0b000000000000000111101011000011010

Resultatet visas enligt det inställda basläget. Inledande nollor visas inte.

shift(*List* *l* [, *#ofShifts*]) $\Rightarrow$ *lista*

Ger en kopia av *List* *l* skiftad åt höger eller vänster av *#ofShifts*-elementen. Ändrar inte *List* *l*.

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 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).

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(StringI [,#ofShifts])**⇒sträng

Ger en kopia av *StringI* skiftad åt höger eller vänster av *#ofShifts*-tecknen. Ändrar inte *StringI*.

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.

<code>shift("abcd")</code>	" abc "
<code>shift("abcd",-2)</code>	" ab "
<code>shift("abcd",1)</code>	"bcd "

sign()Katalog > **sign(ValueI)**⇒värde**sign(ListI)**⇒lista**sign(MatrixI)**⇒matris

Ger, för ett reellt eller komplext *ValueI*, *ValueI* / **abs**(*ValueI*) när *ValueI* ≠ 0.

Ger 1 om *ValueI* är positivt.

Ger -1 om *ValueI* 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.

<code>sign(-3.2)</code>	-1
<code>sign({2,3,4,-5})</code>	{ 1,1,1,-1 }

Om det komplexa formatläget är Real:

<code>sign([-3 0 3])</code>	[-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 84.

Lös för x och y:

$$x + 2y = 1$$

$$3x + 4y = -1$$

<code>simult($\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}$)</code>	$\begin{bmatrix} -3 \\ 2 \end{bmatrix}$
---	---

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:
 $5E-14 \cdot \max(\dim(coeffMatrix)) \cdot \text{rowNorm}(coeffMatrix)$

simult(coeffMatrix, constMatrix[, Tol]) ⇒ *matrix*

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:

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{array}{l} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \text{matx1} \end{array} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\text{simult}\left(\text{matx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \quad \begin{bmatrix} 0 \\ \frac{1}{2} \\ 2 \end{bmatrix}$$

Lös:

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$x + 2y = 2$$

$$3x + 4y = -3$$

$$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right) \quad \begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$$

För det första systemet är lösningen $x=-3$ och $y=2$. För det andra systemet är lösningen $x=-7$ och $y=9/2$.

sin(Value I) ⇒ *värde*

I vinkelläget Grader:

sin(List I) ⇒ *lista*

sin(Value I) Ger sinus för argumentet.

sin()**tangent**

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
$\sin(45^\circ)$	0.707107

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(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.

sin⁻¹()**tangent**

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 

$\sin^{-1}(\text{squareMatrixI}) \Rightarrow \text{kvadratMatris}$

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

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

I vinkelåget 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 > 

$\sinh(\text{NumverI}) \Rightarrow \text{värde}$

$$\sinh(1.2) = 1.50946$$

$\sinh(\text{ListI}) \Rightarrow \text{lista}$

$$\sinh(\{0, 1.2, 3\}) = \{0, 1.50946, 10.0179\}$$

$\sinh(\text{ValueI})$ ger argumentets hyperboliska sinus.

$\sinh(\text{ListI})$ ger en lista på hyperboliska sinus för varje element i *ListI*.

$\sinh(\text{squareMatrixI}) \Rightarrow \text{kvadratMatris}$

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

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

I vinkelåget 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 > 

$\sinh^{-1}(\text{ValueI}) \Rightarrow \text{värde}$

$$\sinh^{-1}(0) = 0$$

$\sinh^{-1}(\text{ListI}) \Rightarrow \text{lista}$

$$\sinh^{-1}(\{0, 2.1, 3\}) = \{0, 1.48748, 1.81845\}$$

$\sinh^{-1}(\text{ValueI})$ ger argumentets inversa hyperboliska sinus.

$\sinh^{-1}(\text{ListI})$ ger en lista på invers hyperbolisk sinus för varje element i *ListI*.

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

sinh⁻¹(squareMatrixI) ⇒ kvadratMatris

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

squareMatrixI 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

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 149.)

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.

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 216).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade <i>X List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.YReg	Lista på datapunkter i den modifierade <i>Y List</i> som används i regressionen baserat på begränsningar i <i>Freq</i> , <i>Category List</i> och <i>Include Categories</i>
stat.FreqReg	Lista på frekvenser som motsvarar <i>stat.XReg</i> och <i>stat.YReg</i>

SortA

SortA *List1* [, *List2*] [, *List3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

SortA *Vector1* [, *Vector2*] [, *Vector3*] ...

SortA <i>list1</i>	<i>Done</i>
--------------------	-------------

Sorterar elementen i det första argumentet i stigande ordning.

<i>list1</i>	$\{1,2,3,4\}$
--------------	---------------

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.

$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
---------------------------------	---------------

SortA <i>list2,list1</i>	<i>Done</i>
--------------------------	-------------

<i>list2</i>	$\{1,2,3,4\}$
--------------	---------------

<i>list1</i>	$\{4,3,2,1\}$
--------------	---------------

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 216.

SortD

Katalog >

SortD *List1* [, *List2*] [, *List3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

SortD *Vector1* [, *Vector2*] [, *Vector3*] ...

$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
---------------------------------	---------------

Identiskt med **SortA** med undantag för att **SortD** sorterar elementen i fallande ordning.

SortD <i>list1</i> , <i>list2</i>	Done
-----------------------------------	------

<i>list1</i>	$\{4,3,2,1\}$
--------------	---------------

<i>list2</i>	$\{3,4,1,2\}$
--------------	---------------

Tomma element inom det första argumentet flyttas till botten. För mer information om tomma element, se på sidan 216.

►Sphere

Katalog >

Vector ►Sphere

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

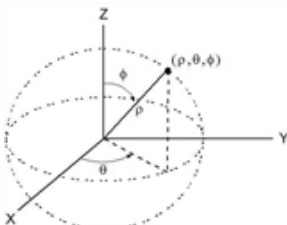
$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ ►Sphere
$\begin{bmatrix} 3.74166 & \angle 1.10715 & \angle 0.640522 \end{bmatrix}$

Visar rad- eller kolumnvektorn i sfärisk form [$\rho \angle \theta \angle \phi$].

$\begin{pmatrix} 2 & \angle \frac{\pi}{4} & 3 \end{pmatrix}$ ►Sphere
$\begin{bmatrix} 3.60555 & \angle 0.785398 & \angle 0.588003 \end{bmatrix}$

Vector måste ha dimensionen 3 och kan vara antingen en rad- eller en kolumnvektor.

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

$\sqrt{4}$	2
------------	---

sqrt(*List1*)⇒lista

$\sqrt{\{9,2,4\}}$	$\{3,1.41421,2\}$
--------------------	-------------------

Ger kvadratroten ur argumentet.

Ger, för en lista, kvadratrötterna ur alla element i *List1*.

Obs: Se även **Square root template**, på sidan 1.

stat.results

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
"r ² "	0.996109
"r"	0.998053
"Resid"	" {... } "
stat.values	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	" {-0.4,0.4,0.2,0.,-0.2} "

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dflInteract	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.F	stat.n	stat.Σx	stat.ȳ
stat.b9	stat.FBlock	stat. $\hat{p}$	stat.Σx ²	stat.ȳ1
stat.b10	stat.Fcol	stat. $\hat{p}1$	stat.Σxy	stat.ȳ2

stat.bList	stat.FInteract	stat. $\hat{p}^2$	stat. Σy	stat. $\bar{X}$ Diff
stat. χ^2	stat.FreqReg	stat. $\hat{p}$ Diff	stat. Σy^2	stat. $\bar{X}$ List
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.Complist	stat.LowerVal	stat.PValCol	stat.SEPred	stat. $\bar{y}$
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. $\hat{y}$
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat. $\hat{y}$ List
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

Se exemplet `stat.results`.

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.

stDevPop()

Katalog > 

`stDevPop(List[,freqList])` ⇒ *uttryck*

I vinkelåget Radianer och i Auto-läge:

Ger populationens standardavvikelse för elementen i *List*.

<code>stDevPop({ 1,2,5,-6,3,-2 })</code>	3.59398
--	---------

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

<code>stDevPop({ 1,3,2,5,-6,4 }, { 3,2,5 })</code>	4.11107
--	---------

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 216.

stDevPop(*Matrix1*[,*freqMatrix*]) $\Rightarrow$ *matrix*

Ger en radvektor med populationens standardavvikelse 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 216.

$$\begin{array}{l} \text{stDevPop} \left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \right) \\ \left[3.26599 \quad 2.94392 \quad 1.63299 \right] \\ \text{stDevPop} \left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix} \right) \\ \left[2.52608 \quad 5.21506 \right] \end{array}$$

stDevSamp(*List*[,*freqList*]) $\Rightarrow$ *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 216.

stDevSamp(*Matrix1*[,*freqMatrix*]) $\Rightarrow$ *matrix*

Ger en radvektor med urvalets standardavvikelse 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 216.

$$\begin{array}{l} \text{stDevSamp}(\{1, 2, 5, -6, 3, -2\}) \quad 3.937 \\ \text{stDevSamp}(\{1.3, 2.5, -6.4\}, \{3, 2, 5\}) \\ 4.33345 \end{array}$$

$$\begin{array}{l} \text{stDevSamp} \left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \right) \\ \left[4. \quad 3.60555 \quad 2. \right] \\ \text{stDevSamp} \left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix} \right) \\ \left[2.7005 \quad 5.44695 \right] \end{array}$$

Stop	Katalog > 	
Stop	<i>i</i> :=0	0
Programmeringskommando: Avslutar programmet.	Define <i>prog1</i> ()=Prgm	Done
	For <i>i</i> ,1,10,1	
	If <i>i</i> =5	
	Stop	
	EndFor	
	EndPrgm	
Stop är ej tillåtet i funktioner.	<i>prog1</i> ()	Done
Obs för att mata in exemplet: Se avsnittet Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.	<i>i</i>	5

Store	Se → (store), på sidan 199.
--------------	-----------------------------

string()	Katalog > 	
string (<i>Expr</i>)⇒ <i>sträng</i>	string(1.2345)	"1.2345"
Förenklar <i>Expr</i> och ger resultatet som en teckensträng.	string(1+2)	"3"

subMat()	Katalog > 	
subMat (<i>Matrix1</i> [, <i>startRow</i>] [, <i>startCol</i>] [, <i>endRow</i>] [, <i>endCol</i>]) ⇒ <i>matrix</i>	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
Ger specificerad undermatris av <i>Matrix1</i> .	subMat(<i>m1</i> ,2,1,3,2)	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
Förvalsinställningar: <i>startRow</i> =1, <i>startCol</i> =1, <i>endRow</i> =sista rad, <i>endCol</i> =sista kolumn.	subMat(<i>m1</i> ,2,2)	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Sum (Sigma)	Se Σ(), på sidan 191.
--------------------	-----------------------

sum()Katalog > **sum(List[, Start[, End]])**⇒uttryckGer 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 216.**sum(Matrix1[, Start[, End]])**⇒*matrix*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 216.

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

$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}\right)$	$\begin{bmatrix} 5 & 7 & 9 \end{bmatrix}$
$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)$	$\begin{bmatrix} 12 & 15 & 18 \end{bmatrix}$
$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},2,3\right)$	$\begin{bmatrix} 11 & 13 & 15 \end{bmatrix}$

sumIf()Katalog > **sumIf(List,Criteria[, SumList])**⇒värdeGer 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 matrix. *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 216.

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

sumSeq()

Se $\Sigma()$, på sidan 191.

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 $\Rightarrow$ *matrix*

Ger den komplexkonjugerade transponeringen av *Matrix1*.

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

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva @t.

tan()

 tangent

tan(*Value1*) $\Rightarrow$ värde

I vinkelläget Grader:

tan(*List1*) $\Rightarrow$ lista

tan(*Value1*) ger tangensen för argumentet.

tan() **tangent**

tan(List I) ger en lista på tangensen för alla element i *List I*.

Obs: Argumentet tolkas som en vinkel i antingen grader, nygrader eller radianer beroende på det aktuella vinkelläget. Du kan använda °, G eller π för att tillfälligt överstyra vinkelläget.

$\tan\left(\left(\frac{\pi}{4}\right)^{\circ}\right)$	1.
$\tan(45)$	1.
$\tan(\{0,60,90\})$	$\{0.,1.73205,\text{undef}\}$

I vinkelläget Nygrader:

$\tan\left(\left(\frac{\pi}{4}\right)^{\circ}\right)$	1.
$\tan(50)$	1.
$\tan(\{0,50,100\})$	$\{0.,1.,\text{undef}\}$

I vinkelläget Radianer:

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45^{\circ})$	1.
$\tan\left(\left\{\pi,\frac{\pi}{3},\pi,\frac{\pi}{4}\right\}\right)$	$\{0.,1.73205,0.,1.\}$

tan(squareMatrix I) $\Rightarrow$ kvadratMatris

Ger matrisen med tangensen för *squareMatrix I*. Detta är inte detsamma som att beräkna tangensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix I måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

$\tan\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$
--	--

tan⁻¹() **tangent**

tan⁻¹(Value I) $\Rightarrow$ värde

I vinkelläget Grader:

tan⁻¹(List I) $\Rightarrow$ lista

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

tan⁻¹(Value I) ger den vinkel vars tangens är *Value I*.

I vinkelläget Nygrader:

tan⁻¹(List I) ger en lista på den inversa tangensen för varje element i *List I*.

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

$\tan^{-1}()$

 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 (...)**.

$\tan^{-1}(\text{squareMatrixI}) \Rightarrow \text{kvadratMatris}$

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

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

I vinkelläget Radianer:

$$\tan^{-1}(\{0,0,2,0,5\}) \quad \{0,0,1.97396,0.463648\}$$

I vinkelläget Radianer:

$$\tan^{-1}\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

$\tanh()$

Katalog > 

$\tanh(\text{ValueI}) \Rightarrow \text{värde}$

$$\tanh(1.2) \quad 0.833655$$

$\tanh(\text{ListI}) \Rightarrow \text{lista}$

$$\tanh(\{0,1\}) \quad \{0,0.761594\}$$

$\tanh(\text{ValueI})$ ger den hyperboliska tangensen för argumentet.

$\tanh(\text{ListI})$ ger en lista på den hyperboliska tangensen för varje element i *ListI*.

$\tanh(\text{squareMatrixI}) \Rightarrow \text{kvadratMatris}$

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

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

I vinkelläget Radianer:

$$\tanh\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

$\tanh^{-1}()$

Katalog > 

$\tanh^{-1}(\text{ValueI}) \Rightarrow \text{värde}$

I Rektangulärt komplext format:

$\tanh^{-1}(\text{ListI}) \Rightarrow \text{lista}$

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)⇒kvadratMatrix

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})	{undef,0.518046-1.5708 <i>i</i> ,0.346574-1.5708 <i>i</i> }

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 $\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} -0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.94730 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901 \end{bmatrix}$
---	---

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 $P(X \leq upBound)$, sätt *lowBound* = -9E999.

Text

Katalog > 

TextfrågeSträng[, 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**.

Det valfria argumentet *flagga* kan vara ett valfritt uttryck.

- Om *DispFlagga* utelämnas eller beräknas till **1** sparas textmeddelandet i Räknares 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 127 eller **RequestStr**, på sidan 129.

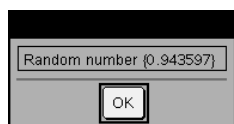
Obs: Du kan använda detta kommando inom ett användardefinierat program, men inte inom en funktion.

```
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:



tInterval *List[,Freq[,CLevel]]*

(Indata lista)

tInterval $\bar{x}, s_x, n[, CLevel]$

(Summary stats indata)

Beräknar ett *t*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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. σ_x	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 149.)

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 216).

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. $\sigma x1$, stat. $\sigma x2$	Urvalets standardavvikelser för <i>List 1</i> och <i>List 2</i>

Resultatvariabel	Beskrivning
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.sp	Den sammanslagna standardavvikelsen. Beräknas när <i>Pooled</i> = 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*.

$\text{trace} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	15
<i>a</i> :=12	12
$\text{trace} \begin{pmatrix} a & 0 \\ 1 & a \end{pmatrix}$	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 226).

Define <i>prog1</i> ()=Prgm	
Try	
<i>z</i> := <i>z</i> +1	
Disp "z incremented."	
Else	
Disp "Sorry, z undefined."	
EndTry	
EndPrgm	
	Done
<i>z</i> :=1: <i>prog1</i> ()	
	z incremented.
	Done
DelVar <i>z</i> : <i>prog1</i> ()	
	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 Räkna 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.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

Obs: Se även **ClrErr**, på sidan 22 och **PassErr**, på sidan 112.

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

tTest $\mu_0, \text{List}, \text{Freq}, [\text{Hypoth}]$

(Indata lista)

tTest $\mu_0, \bar{x}, s_x, n, [\text{Hypoth}]$

(Summary stats indata)

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 149.)

Testa $H_0: \mu = \mu_0$, mot en av följande:

För $H_a: \mu < \mu_0$, ställ *Hypoth*<0

För $H_a: \mu \neq \mu_0$ (förinställning), ställ *Hypoth*=0

För $H_a: \mu > \mu_0$, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader
stat. $\bar{x}$	Urvalsmedelvärde för datasekvensen i <i>List</i>
stat.sx	Urvalets standardavvikelse för datasekvensen
stat.n	Urvalets storlek

tTest_2Samp

tTest_2Samp *List1, List2[, Freq1[, Freq2[, Hypoth[, Pooled]]]]*

(Indata lista)

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 149.)

Testa $H_0: \mu_1 = \mu_2$, mot en av följande:

För $H_a: \mu_1 < \mu_2$, ställ *Hypoth*<0

För $H_a: \mu_1 \neq \mu_2$ (föinställning), ställ
Hypoth=0

För $H_a: \mu_1 > \mu_2$, 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 216).

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.x̄1, stat.x̄2	Medelvärden hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>
stat.sx1, stat.sx2	Standardavvikelse hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>
stat.n1, stat.n2	Storlek på urvalen
stat.sp	Den sammanslagna standardavvikelsen. Beräknad när <i>Pooled</i> =1.

tvmFV()

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 164. Se även **amortTbl()**, på sidan 7.

tvmI()

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 164. Se även **amortTbl()**, på sidan 7.

tvmN()Katalog > 

tvmN($I, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$) $\Rightarrow$ värde

tvmN(5,0,-500,77641,12,12) 120.

Finansiell funktion som beräknar antalet betalningsperioder.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 164. Se även **amortTbl()**, på sidan 7.

tvmPmt()Katalog > 

tvmPmt($N, I, PV, FV, [PpY], [CpY], [PmtAt]$) $\Rightarrow$ värde

tvmPmt(60,4,30000,0,12,12) -552.496

Finansiell funktion som beräknar beloppet på varje betalning.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 164. Se även **amortTbl()**, på sidan 7.

tvmPV()Katalog > 

tvmPV($N, I, Pmt, FV, [PpY], [CpY], [PmtAt]$) $\Rightarrow$ värde

tvmPV(48,4,-500,30000,12,12) -3426.7

Finansiell funktion som beräknar nuvärdet.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 164. Se även **amortTbl()**, på sidan 7.

TVM-argument*	Beskrivning	Datotyp
N	Antal betalningsperioder	reellt tal

TVM-argument*	Beskrivning	Datatyp
<i>I</i>	Årlig räntesats	reellt tal
<i>PV</i>	Nuvärde	reellt tal
<i>Pmt</i>	Betalningsbelopp	reellt tal
<i>FV</i>	Framtida värde	reellt tal
<i>PpY</i>	Antal betalningar per år, förinställning = 1	heltal > 0
<i>CpY</i>	Ränteperioder per år, förinställning = 1	heltal > 0
<i>PmtAt</i>	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 149.)

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 216.

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 ₁ X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q ₃ X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.MinY	Minsta y-värde
stat.Q ₁ Y	Undre kvartil för y
stat.MedY	Median för y
stat.Q ₃ Y	Övre kvartil för y

Resultatvariabel	Beskrivning
stat.MaxY	Största y-värde
stat. (x-) ²	Kvadratsumma för avvikelser från medelvärdet på x
stat. (y-) ²	Kvadratsumma för avvikelser från medelvärdet på y

U

unitV()	Katalog > 
unitV(<i>Vector1</i>) ⇒vektor	unitV($\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$)
Ger en radenhets- eller kolumnenhetsvektor beroende på formen hos <i>Vector1</i> .	$\begin{bmatrix} 0.408248 & 0.816497 & 0.408248 \end{bmatrix}$
<i>Vector1</i> måste vara antingen en enradsmatris eller en enkolumnsmatris.	unitV($\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$)
	$\begin{bmatrix} 0.267261 \\ 0.534522 \\ 0.801784 \end{bmatrix}$

unLock	Katalog > 
unLockVar1[, Var2] [, Var3] ...	a:=65 65
unLockVar.	Lock a Done
Låser upp den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.	getLockInfo(a) 1
Se Lock , på sidan 87 och getLockInfo() , på sidan 65.	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	varPop({ 5,10,15,20,25,30 }) 72.9167
Ger populationsvariansen för <i>List</i> .	
Varje <i>freqList</i> -element räknar antalet förekomster av motsvarande element i <i>List</i> .	
Obs: <i>List</i> 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 216.

varSamp()

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 216.

varSamp(MatrixI[, freqMatrix]) ⇒ *matrix*

Ger en radvektor som innehåller sampelvariansen för varje kolumn i *MatrixI*.

Varje *freqMatrix*-element räknar antalet konsekutiva förekomster av motsvarande element i *MatrixI*.

Obs: *MatrixI* 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 216.

$\text{varSamp}(\{1, 2, 5, 6, 3, -2\})$	$\frac{31}{2}$
$\text{varSamp}(\{1, 3, 5\}, \{4, 6, 2\})$	$\frac{68}{33}$

$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right)$	$[4.75 \quad 1.03 \quad 4]$
$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$	$[3.91731 \quad 2.08411]$

W

Wait

Wait *tidISekunder*

För att vänta 4 sekunder:

Wait 4

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  och tryck på  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 ()

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 235.

	warnCodes(det([1.23456E-999]),warn)
	1.23456E-999
warn	{10029}

when(*Condition*, *trueResult* [, *falseResult*] [, *unknownResult*]) $\Rightarrow$ 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.

Utelämnar både *falseResult* och *unknownResult* för att skapa ett uttryck som är definierat endast i området där *Condition* är sant.

Använd ett **undef** *falseResult* för att definiera ett uttryck som plottas endast i ett intervall.

when() är användbar för att definiera rekursiva funktioner.

$\text{when}(x < 0, x + 3) \mid x = 5$	undef
--	-------

$\text{when}(n > 0, n \cdot \text{factorial}(n - 1), 1) \rightarrow \text{factorial}(n)$	Done
$\text{factorial}(3)$	6
3!	6

While

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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $\text{sum_of_recip}(n)$ = Func	
Local $i, \text{tempsum}$	
$1 \rightarrow i$	
$0 \rightarrow \text{tempsum}$	
While $i \leq n$	
$\text{tempsum} + \frac{1}{i} \rightarrow \text{tempsum}$	
$i + 1 \rightarrow i$	
EndWhile	
Return tempsum	
EndFunc	Done
$\text{sum_of_recip}(3)$	11
	6

zIntervalKatalog > **zInterval** $\sigma, List[, Freq[, CLevel]]$

(Indatalista)

zInterval $\sigma, \bar{x}, n [, CLevel]$

(Summary stats indata)

Beräknar ett z -konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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.sx	Standardavvikelse för urvalet
stat.n	Längden på datasekvenser med urvalsmedelvärde
stat. σ	Känd populationsstandardavvikelse för datasekvensen <i>List</i>

zInterval_1PropKatalog > **zInterval_1Prop** $x, n [, CLevel]$

Beräknar ett 1-proportion z -konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

x är ett icke negativt heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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 149.)

$x1$ och $x2$ är icke negativa heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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 $\sigma_1, \sigma_2, List1, List2, [Freq1, Freq2, [CLevel]]]$

(Indatalista)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2, [CLevel]$

(Summary stats indata)

Beräknar ett 2-sampel z -konfidensintervall.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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. σx_1 , stat. σx_2	Urvalets standardavvikelse för <i>List 1</i> och <i>List 2</i>
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.r1, stat.r2	Kända populationsstandardavvikelse för datasekvenserna <i>List 1</i> och <i>List 2</i>

zTest

zTest $\mu_0, \sigma, List, [Freq[, Hypoth]]$

(Indatalista)

zTest $\mu_0, \sigma, \bar{x}, n, [Hypoth]$

(Summary stats indata)

Utför ett z -test med frekvensen *freqlist*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

Testa $H_0: \mu = \mu_0$, mot en av följande:

För $H_a: \mu < \mu_0$, ställ *Hypoth* < 0

För $H_a: \mu \neq \mu_0$ (föinställning), ställ *Hypoth* = 0

För $H_a: \mu > \mu_0$, ställ *Hypoth* > 0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.z	$(\bar{x} - \mu_0) / (\sigma / \sqrt{n})$
stat.P Value	Lägsta sannolikhet vid vilken nollhypotesen kan förkastas
stat. $\bar{x}$	Urvalsmedelvärde för datasekvensen i <i>List</i>
stat.sx	Urvalets standardavvikelse för datasekvensen. Ges endast för <i>Data</i> -inmatningen.
stat.n	Urvalets storlek

zTest_1Prop

zTest_1Prop $p_0, x, n[, Hypoth]$

Beräknar ett 1-proportion z-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

x är ett icke negativt heltal.

Testa $H_0: p = p_0$ mot en av följande:

För $H_a: p > p_0$, ställ *Hypoth*>0

För $H_a: p \neq p_0$ (förinställning), ställ *Hypoth*=0

För $H_a: p < p_0$, ställ *Hypoth*<0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

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 $x1, n1, x2, n2[, Hypoth]$

Beräknar ett 2-proportion z-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

$x1$ 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 216).

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**zTest_2Samp** $\sigma_1, \sigma_2, List1, List2[, Freq1[, Freq2[, Hypoth]]]$

(Indatalista)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2[, Hypoth]$

(Summary stats indata)

Beräknar ett 2-sampel z-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 149.)

Testa $H_0: \mu1 = \mu2$, mot en av följande:

För $H_a: \mu_1 < \mu_2$, ställ *Hypoth*<0

För $H_a: \mu_1 \neq \mu_2$ (förinställning), ställ
Hypoth=0

För $H_a: \mu_1 > \mu_2$, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 216).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.x̄1, stat.x̄2	Medelvärden hos urvalet i datasekvenserna i <i>List1</i> och <i>List2</i>
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i <i>List1</i> och <i>List2</i>
stat.n1, stat.n2	Storlek på urvalen

Symboler

+ (addera)		 tangent
$Value1 + Value2 \Rightarrow värde$	56	56
Ger summan av de två argumenten.	56+4	60
	60+4	64
	64+4	68
	68+4	72
$List1 + List2 \Rightarrow lista$	$\left\{22,\pi,\frac{\pi}{2}\right\} \rightarrow I1$	$\{22,3.14159,1.5708\}$
$Matrix1 + Matrix2 \Rightarrow matris$	$\left\{10,5,\frac{\pi}{2}\right\} \rightarrow I2$	$\{10,5,1.5708\}$
Ger en lista (eller matris) som innehåller summorna av motsvarande element i <i>List1</i> och <i>List2</i> (eller <i>Matrix1</i> och <i>Matrix2</i>).	$I1+I2$	$\{32,8.14159,3.14159\}$
Argumenten måste ha samma dimensioner.		
$Value + List1 \Rightarrow lista$	$15+\{10,15,20\}$	$\{25,30,35\}$
$List1 + Value \Rightarrow lista$	$\{10,15,20\}+15$	$\{25,30,35\}$
Ger en lista på summorna av <i>Value</i> och varje element i <i>List1</i> .		
$Value + Matrix1 \Rightarrow matris$	$20+\begin{bmatrix}1 & 2 \\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2 \\ 3 & 24\end{bmatrix}$
$Matrix1 + Value \Rightarrow matris$		
Ger en matris med <i>Value</i> adderat till varje element i diagonalen av <i>Matrix1</i> . <i>Matrix1</i> måste vara kvadratisk.		
Obs: Använd .+ (punkt plus) för att lägga till ett uttryck till varje element.		

-(subtrahera)		 tangent
$Value1 - Value2 \Rightarrow värde$	6-2	4
Ger <i>Value1</i> minus <i>Value2</i> .	$\pi-\frac{\pi}{6}$	2.61799
$List1 - List2 \Rightarrow lista$	$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,-1.85841,0\}$
$Matrix1 - Matrix2 \Rightarrow matris$	$\begin{bmatrix}3 & 4\end{bmatrix}-\begin{bmatrix}1 & 2\end{bmatrix}$	$\begin{bmatrix}2 & 2\end{bmatrix}$

–(subtrahera)

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*

$$15 - \{10, 15, 20\} \quad \{5, 0, -5\}$$

List1 – *Value* ⇒ *lista*

$$\{10, 15, 20\} - 15 \quad \{-5, 0, 5\}$$

Subtraherar varje element i *List1* från *Value* eller subtraherar *Value* från varje element i *List1* och ger en lista på resultaten.

Value – *Matrix1* ⇒ *matris*

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

Matrix1 – *Value* ⇒ *matris*

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)

Value1 · *Value2* ⇒ *värde*

$$2 \cdot 3.45 \quad 6.9$$

Ger produkten av de två argumenten.

List1 · *List2* ⇒ *lista*

$$\{1, 2, 3\} \cdot \{4, 5, 6\} \quad \{4, 10, 18\}$$

Ger en lista som innehåller produkterna av motsvarande element i *List1* och *List2*.

Listorna måste ha samma dimensioner.

Matrix1 · *Matrix2* ⇒ *matris*

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 7 & 8 \\ 7 & 8 \end{bmatrix} \quad \begin{bmatrix} 42 & 48 \\ 105 & 120 \end{bmatrix}$$

Ger en matris över produkten av *Matrix1* och *Matrix2*.

Antalet kolumner i *Matrix1* måste vara lika med antalet rader i *Matrix2*.

·(multiplicera)

[x] tangent

Value · List1 ⇒ lista

$$\pi \cdot \{4,5,6\} \quad \{12.5664, 15.708, 18.8496\}$$

List1 · Value ⇒ lista

Ger en lista på produkterna av *Value* och varje element i *List1*.

Value · Matrix1 ⇒ matris

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 \quad \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

Matrix1 · Value ⇒ matris

$$6 \cdot \text{identity}(3) \quad \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Ger en matris över produkterna av *Value* och varje element i *Matrix1*.

Obs: Använd · (punkt multiplicera) för att multiplicera ett uttryck med varje element.

/ (dividera)

[÷] tangent

Value1 / Value2 ⇒ värde

$$\frac{2}{3.45} \quad 0.57971$$

Ger kvoten av *Value1* dividerat med *Value2*.

Obs: Se även **Fraction template**, på sidan 1.

List1 / List2 ⇒ lista

$$\frac{\{1.,2,3\}}{\{4,5,6\}} \quad \left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$$

Ger en lista på kvoterna av *List1* dividerat med *List2*.

Listorna måste ha samma dimensioner.

Value / List1 ⇒ lista

$$\frac{6}{\{3,6,\sqrt{6}\}} \quad \{2,1,2.44949\}$$

List1 / Value ⇒ lista

$$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2} \quad \left\{\frac{1}{18}, \frac{1}{14}, \frac{1}{63}\right\}$$

Ger en lista på kvoterna av *Value* dividerat med *List1* eller *List1* dividerat med *Value*.

Value / Matrix1 ⇒ matris

$$\frac{\begin{bmatrix} 7 & 9 & 2 \end{bmatrix}}{7 \cdot 9 \cdot 2} \quad \begin{bmatrix} \frac{1}{18} & \frac{1}{14} & \frac{1}{63} \end{bmatrix}$$

Matrix1 / Value ⇒ matris

Ger en matris över kvoterna av *Matrix1/Value*.

Obs: Använd / (punkt dividera) för att dividera ett uttryck med varje element.

^ (potens)



$Value1 \wedge Value2 \Rightarrow \text{värde}$

$$4^2$$

16

$List1 \wedge List2 \Rightarrow \text{lista}$

$$\{2,4,6\}^{\{1,2,3\}}$$

$$\{2,16,216\}$$

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.

$Value \wedge List1 \Rightarrow \text{lista}$

$$\pi^{\{1,2,-3\}}$$

$$\{3.14159, 9.8696, 0.032252\}$$

Ger *Value* upphöjt till potensen för elementen i *List1*.

$List1 \wedge Value \Rightarrow \text{lista}$

$$\{1,2,3,4\}^{-2}$$

$$\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$$

Ger elementen i *List1* upphöjda till potensen för *Value*.

$squareMatrix1 \wedge integer \Rightarrow \text{matris}$

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

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

Ger *squareMatrix1* upphöjd till potensen för *integer* (heltal).

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

$$\begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

squareMatrix1 måste vara en kvadratisk matris.

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

$$\begin{bmatrix} \frac{11}{2} & -\frac{5}{2} \\ -\frac{15}{4} & \frac{7}{4} \end{bmatrix}$$

Om *integer* = -1 beräknas den inversa matrisen.

Om *integer* < -1 beräknas den inversa matrisen till en lämplig positiv potens.

x² (kvadrat)**x² tangent***Value*¹² ⇒ värde

4^2	16
-------	----

Ger kvadraten på argumentet.

$\{2,4,6\}^2$	$\{4,16,36\}$
---------------	---------------

*List*¹² ⇒ lista

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2$	$\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$
---	--

Ger en lista med kvadraterna på elementen i *List*¹.*squareMatrix*¹² ⇒ matris

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^{\wedge 2}$	$\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$
--	--

Ger en matris över kvadraten på *squareMatrix*¹. Detta är inte detsamma som att beräkna kvadraten på varje element. Använd ^{^2} för att beräkna kvadraten på varje element.

+. (punkt addera)**.+ tangenter***Matrix*¹ .+ *Matrix*² ⇒ matris

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$
---	--

Value .+ *Matrix*¹ ⇒ matris

$5 + \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$
--	--

*Matrix*¹ .+ *Matrix*² ger en matris som är summan av varje par av motsvarande element i *Matrix*¹ och *Matrix*².

Value .+ *Matrix*¹ ger en matris som är summan av *Value* och varje element i *Matrix*¹.

.- (punkt subtrahera)**.- tangenter***Matrix*¹ .- *Matrix*² ⇒ matris

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	$\begin{bmatrix} -9 & -18 \\ -27 & -36 \end{bmatrix}$
---	---

Value .- *Matrix*¹ ⇒ matris

$5 - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	$\begin{bmatrix} -5 & -15 \\ -25 & -35 \end{bmatrix}$
--	---

*Matrix*¹ .- *Matrix*² ger en matris som är skillnaden mellan varje par av motsvarande element i *Matrix*¹ och *Matrix*².

Value .- *Matrix*¹ ger en matris som är skillnaden mellan *Value* och varje element i *Matrix*¹.

· (punkt multiplicera)

 tangenter

Matrix1 · *Matrix2* ⇒ *matrix*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$$

Value · *Matrix1* ⇒ *matrix*

$$5 \cdot \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 50 & 100 \\ 150 & 200 \end{bmatrix}$$

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*.

/ (punkt dividera)

 tangenter

Matrix1 / *Matrix2* ⇒ *matrix*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} / \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{10} \end{bmatrix}$$

Value / *Matrix1* ⇒ *matrix*

$$5 / \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{8} \end{bmatrix}$$

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*.

^ (punkt potens)

 tangenter

Matrix1 ^ *Matrix2* ⇒ *matrix*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} ^ \begin{bmatrix} 0 & 2 \\ 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 27 & \frac{1}{4} \end{bmatrix}$$

Value ^ *Matrix1* ⇒ *matrix*

$$5 ^ \begin{bmatrix} 0 & 2 \\ 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 25 \\ 125 & \frac{1}{5} \end{bmatrix}$$

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*.

-(negation)

 tangent

-*Value1* ⇒ värde

$$-2.43 \quad -2.43$$

-*List1* ⇒ lista

$$\{-1, 0.4, 1.2 \text{E}19\} \quad \{1., -0.4, -1.2 \text{E}19\}$$

-*Matrix1* ⇒ matris

-(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.

-Ob100101

Ob11111111111111111111111111111111▶

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

% (procent)

  tangenter

Value1 % ⇒ *värde*

List1 % ⇒ *lista*

Matrix1 % ⇒ *matris*

argument

Ger 100

Ger, för en lista eller matris, en lista eller matris med varje element dividerat med 100.

Obs: För att få ett närmvärde,

Handenhet: Tryck på  .

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 $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ Then

Return -x

ElseIf $x \geq 0$ and $x \neq 10$ Then

Return x

ElseIf $x = 10$ Then

Return 3

EndIf

EndFunc

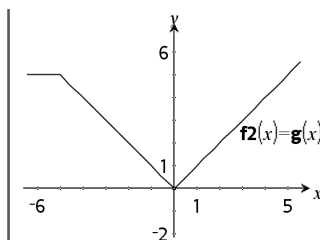
Done

= (lika med)

 **tangent**

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

Resultat från plottning av $g(x)$



 $f2(x)=g(x)$

≠ (inte lika med)

  **tangenter**

$Expr1 \neq Expr2 \Rightarrow$ Booleskt uttryck

Se exemplet "=" (lika med).

$List1 \neq List2 \Rightarrow$ Boolesk lista

$Matrix1 \neq Matrix2 \Rightarrow$ 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 /=

< (mindre än)

  **tangenter**

$Expr1 < Expr2 \Rightarrow$ Booleskt uttryck

Se exemplet "=" (lika med).

$List1 < List2 \Rightarrow$ Boolesk lista

$Matrix1 < Matrix2 \Rightarrow$ Boolesk matris

Ger resultatet sant om $Expr1$ bestäms vara mindre än $Expr2$.

< (mindre än)

  tangent

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.

$\leq$ (mindre än eller lika med)

  tangent

$Expr1 \leq Expr2 \Rightarrow$ Boolesk uttryck

Se exemplet "=" (lika med).

$List1 \leq List2 \Rightarrow$ Boolesk lista

$Matrix1 \leq Matrix2 \Rightarrow$ 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 $\leq$

> (större än)

  tangent

$Expr1 > Expr2 \Rightarrow$ Boolesk uttryck

Se exemplet "=" (lika med).

$List1 > List2 \Rightarrow$ 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.

> (större än)

ctrl **=** **tangenter**

Ger, för listor och matriser, jämförelser
element för element.

≥ (större än eller lika med)

ctrl **=** **tangenter**

$Expr1 \geq Expr2 \Rightarrow$ Booleskt uttryck

Se exemplet "=" (lika med).

$List1 \geq List2 \Rightarrow$ Boolesk lista

$Matrix1 \geq Matrix2 \Rightarrow$ 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 $\geq$

$\Rightarrow$ (logisk implikation)

ctrl **=** **knappar**

$BoolesktUtr1 \Rightarrow BoolesktUtr2$ ger
 $Booleskt$ uttryck

$5 > 3$ or $3 > 5$ true

$5 > 3 \Rightarrow 3 > 5$ false

$BooleskList1 \Rightarrow BooleskList2$ ger
 $Boolesk$ lista

3 or 4 7

$3 \Rightarrow 4$ -4

$BooleskMatris1 \Rightarrow BooleskMatris2$ ger
 $Boolesk$ matris

$\{1, 2, 3\}$ or $\{3, 2, 1\}$ $\{3, 2, 3\}$

$\{1, 2, 3\} \Rightarrow \{3, 2, 1\}$ $\{-1, -1, -3\}$

$Heltal1 \Rightarrow Heltal2$ ger $Heltal$

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 $\Rightarrow$

$\Leftrightarrow$ (logisk dubbel implikation, XNOR)

  **knappar**

BoolesktUttr1 $\Leftrightarrow$ *BoolesktUttr2* ger
Booleskt uttryck

$5 > 3 \text{ xor } 3 > 5$ true

$5 > 3 \Leftrightarrow 3 > 5$ false

BooleskLista1 $\Leftrightarrow$ *BooleskLista2* ger
Boolesk lista

$3 \text{ xor } 4$ 7

$3 \Leftrightarrow 4$ -8

BooleskMatris1 $\Leftrightarrow$ *BooleskMatris2* ger
Boolesk matris

$\{1,2,3\} \text{ xor } \{3,2,1\}$ $\{2,0,2\}$

$\{1,2,3\} \Leftrightarrow \{3,2,1\}$ $\{-3,-1,-3\}$

Heltal1 $\Leftrightarrow$ *Heltal2* ger *Heltal*

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 $\Leftrightarrow$

! (fakultet)

 **tangent**

Value1! $\Rightarrow$ värde

$5!$ 120

List1! $\Rightarrow$ lista

$\{\{5,4,3\}\}!$ $\{120,24,6\}$

Matrix1! $\Rightarrow$ matris

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$ $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Ger argumentets fakultet.

Ger, för en lista eller matris, en lista eller matris med elementens faktieller.

& (lägg till)

  **tangenter**

String1 & *String2* $\Rightarrow$ sträng

"Hello " & "Nick"

"Hello Nick"

Ger en textsträng som är *String2* bifogad till *String1*.

d(Uttr1, Var[, Ordning]) |
Var = Värde ⇒ värde

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

d(Uttr1, Var[, Ordning]) ⇒ värde

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

d(Lista1, Var[, Ordning]) ⇒ lista

$$x:=3: \frac{d}{dx}(\{x^2, x^3, x^4\}) \quad \{6, 27, 108\}$$

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.

$$\frac{d}{dx} \left(x \cdot (x^2 + x)^{\frac{1}{3}} \right) \Big|_{x=0} \quad \text{undef}$$

$$\text{centralDiff} \left(x \cdot (x^2 + x)^{\frac{1}{3}}, x \right) \Big|_{x=0} \quad 0.000033$$

Studera exemplet till höger.

Förstaderivatan av $x \cdot (x^2 + x)^{1/3}$ 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()**.

$\int()$ (integral)

Katalog > 

$\int(Uttr1, Var, Undre, Övre) \Rightarrow värde$

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()`.

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

Obs: Se även `nint()`, på sidan 104 och **Mallen för bestämd integral**, på sidan 6.

$$\int_0^1 x^2 dx = 0.333333$$

$\sqrt{()}$ (kvadratrot)

  **tangenter**

$\sqrt{Value1} \Rightarrow värde$

$$\sqrt{4} = 2$$

$\sqrt{List1} \Rightarrow lista$

$$\sqrt{\{9,2,4\}} = \{3,1.41421,2\}$$

Ger kvadratroten ur argumentet.

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.

$\prod()$ (prodSeq)

Katalog > 

$\prod(Expr1, Var, Low, High) \Rightarrow uttryck$

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

Utvärderar *Expr1* för varje värde på *Var* från *Low* till *High* och ger produkten av resultaten.

Obs: Se även **Product template** ($\prod$), på sidan 5.

$$\prod_{n=1}^5 \left(\frac{1}{n} \right) = \frac{1}{120}$$
$$\prod_{n=1}^5 \left(\left\{ \frac{1}{n}, n, 2 \right\} \right) = \left\{ \frac{1}{120}, 120, 32 \right\}$$

$\Pi()$ (prodSeq)

Katalog > 

$\Pi(\text{Expr1}, \text{Var}, \text{Low}, \text{Low}-1) \Rightarrow 1$

$$\prod_{k=4}^3 (k) = 1$$

$\Pi(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow 1/\Pi(\text{Expr1}, \text{Var}, \text{High}+1, \text{Low}-1)$ om $\text{High} < \text{Low}-1$

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.

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) = 6$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) \cdot \prod_{k=2}^4 \left(\frac{1}{k}\right) = \frac{1}{4}$$

$\Sigma()$ (sumSeq)

Katalog > 

$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow \text{uttryck}$

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.

$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{Low}-1) \Rightarrow 0$

$$\sum_{n=1}^5 \left(\frac{1}{n}\right) = \frac{137}{60}$$

$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow -\Sigma(\text{Expr1}, \text{Var}, \text{High}+1, \text{Low}-1)$ om $\text{High} < \text{Low}-1$

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.

$$\sum_{k=4}^1 (k) = -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) = 4$$

$\Sigma\text{Int}()$ **Katalog >** 

$\Sigma\text{Int}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [roundValue]) \Rightarrow \text{värde}$

$\Sigma\text{Int}(1, 3, 12, 4.75, 20000., 12, 12)$	-213.48
--	---------

$\Sigma\text{Int}(NPmt1, NPmt2, amortTable) \Rightarrow \text{värde}$

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 164.

- Om du utelämnar Pmt används förinställningen $Pmt = \text{tvmPmt}(N, I, PV, FV, PpY, CpY, PmtAt)$.
- 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.

$\Sigma\text{Int}(NPmt1, NPmt2, amortTable)$ 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 $\Sigma\text{Prn}()$ nedan och **Bal()**, på sidan 15.

$tbl := \text{amortTbl}(12, 12, 4.75, 20000., 12, 12)$				
0	0.	0.	20000.	
1	-77.49	-1632.43	18367.6	
2	-71.17	-1638.75	16728.8	
3	-64.82	-1645.1	15083.7	
4	-58.44	-1651.48	13432.2	
5	-52.05	-1657.87	11774.4	
6	-45.62	-1664.3	10110.1	
7	-39.17	-1670.75	8439.32	
8	-32.7	-1677.22	6762.1	
9	-26.2	-1683.72	5078.38	
10	-19.68	-1690.24	3388.14	
11	-13.13	-1696.79	1691.35	
12	-6.55	-1703.37	-12.02	
$\Sigma\text{Int}(1, 3, tbl)$				-213.48

 $\Sigma\text{Prn}()$ **Katalog >** 

$\Sigma\text{Prn}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [roundValue]) \Rightarrow \text{värde}$

$\Sigma\text{Prn}(1, 3, 12, 4.75, 20000., 12, 12)$	-4916.28
--	----------

$\Sigma\text{Prn}(NPmt1, NPmt2, amortTable) \Rightarrow \text{värde}$

Amorteringsfunktion som beräknar kapitalsumman 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 164.

- 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)

  **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"→*sI* "r"

#*sI* 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@**E**4 för att mata in 2.3**E**4.

23000.	23000.
23000000000.+4.1 E 15	4.1 E 15
3·10 ⁴	30000

g (nygrad)

1 tangent

*Expr***l****g**⇒*uttryck*

*List***l****g**⇒*lista*

*Matrix***l****g**⇒*matrix*

Denna funktion ger dig ett sätt att specificera en vinkel i nygrader när du är i läge Grader eller Radianer.

Multiplikerar i vinkelläget Radianer *Expr***l** med $\pi/200$.

Multiplikerar i vinkelläget Grader *Expr***l** med $g/100$.

Ger i läget Nygrader *Expr***l** 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)

1 tangent

*Value***l****r** ⇒*värde*

*List***l****r** ⇒*lista*

*Matrix***l****r** ⇒*matrix*

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
$\cos\left(\left\{0^r,\left(\frac{\pi}{12}\right)^r,(\pi)^r\right\}\right)$	$\{1,0.965926,-1.\}$

° (radian)

1 tangent

Multipliserar i vinkelläget Grader argumentet med $180/\pi$.

Ger i vinkelläget Radianer argumentet oförändrat.

Multipliserar i vinkelläget Nygrader argumentet med $200/\pi$.

Tips: Använd ° 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 @π.

° (grader)

1 tangent

Value°⇒värde

List°⇒lista

Matrix°⇒matrix

Denna funktion ger dig ett sätt att specificera en vinkel i grader när du är i läge Nygrader eller Radianer.

Multipliserar i vinkelläget Radianer argumentet med $\pi/180$.

Ger i vinkelläget Grader argumentet oförändrat.

Multipliserar 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:

$\cos(45^\circ)$	0.707107
------------------	----------

I vinkelläget Radianer:

°, ', " (grad/minut/sekund)

ctrl tangent

dd°*mm'**ss.ss''*⇒uttryck

*dd*Ett positivt eller negativt tal

*mm*Ett icke negativt tal

*ss.ss*Ett icke negativt tal

I vinkelläge Grader:

$25^\circ 13' 17.5''$	25.2215
$25^\circ 30'$	$\frac{51}{2}$

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)

$[Radius, \angle_Angle] \Rightarrow \text{vektor}$

(polär indata)

$[Radius, \angle_Angle, Z_Coordinate] \Rightarrow \text{vektor}$

(cylindrisk indata)

$[Radius, \angle\theta_Angle, \angle\theta_Angle] \Rightarrow \text{vektor}$

(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	∠60°	∠45°
[1.76777 3.06186 3.53553]		

cylindrisk

5	∠60°	∠45°
[3.53553 ∠1.0472 3.53553]		

sfärisk

5	∠60°	∠45°
[5. ∠1.0472 ∠0.785398]		

$(Magnitude \angle Angle) \Rightarrow \text{complexValue}$

(polär indata)

Matar in ett komplext värde i $(r\angle\theta)$ polär form. *Angle* tolkas enligt det inställda vinkelläget.

I vinkelläget Radianer och i Rektangulärt komplext format:

10^()

Katalog > 

10^ (Value) ⇒ värde

10^{1.5}

31.6228

10^ (List1) ⇒ lista

Ger 10 upphöjd till argumentets potens.

Ger, för en lista, 10 upphöjd till potensen för elementen i *List1*.

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.

$$10^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}} = \begin{bmatrix} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{bmatrix}$$

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

^-1 (inverterat värde)

Katalog > 

Value1 ^-1 ⇒ värde

(3.1)⁻¹

0.322581

List1 ^-1 ⇒ lista

Ger argumentets inverterade värde.

Ger, för en lista, de inverterade värdena på elementen i *List1*.

squareMatrix1 ^-1 ⇒ kvadratMatris

Ger inversen av *squareMatrix1*.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} = \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

squareMatrix1 måste vara en icke singulär kvadratisk matris.

| (operatörn begränsning)

  tangenter

Uttr | BoolesktUttr1 [andBoolesktUttr2]...

x+1|x=3

4

x+55|x=sin(55)

54.0002

Uttr | BoolesktUttr1 [orBoolesktUttr2]...

("|")-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.

Operatör 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.

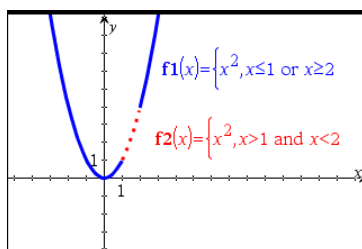
Uttr | Variabel = värde ersätter värde vid varje förekomst av *Variabel* i *Uttr*.

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.

Uteslutningar använder jämförelseoperatör "inte lika med" ($\neq$ eller $\neq$) för att utesluta ett specifikt värde från övervägning.

$x^3-2\cdot x+7\rightarrow f(x)$	Done
$f(x) x=\sqrt{3}$	8.73205

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x)$	0.
$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x) x>0 \text{ and } x<5$	3.



→ (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.`

$\frac{\pi}{4} \rightarrow myvar$	0.785398
$2 \cdot \cos(x) \rightarrow y1(x)$	Done
$\{1,2,3,4\} \rightarrow lst5$	$\{1,2,3,4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → str1	"Hello"

:= (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"

© [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 Räkna 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+i2
    EndFor
    Return result
EndFunc
Done
g(3) 14
```

0b, 0h

0b *binaryNumber*

I decimalt basläge:

0b10+0hF+10	27
-------------	----

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).

I binärt basläge:

0b10+0hF+10	0b11011
-------------	---------

Resultaten visas enligt det inställda basläget.

I hexadecimalt basläge:

0b10+0hF+10	0h1B
-------------	------

TI-Nspire™ CX II-Kommandon för att rita

Detta är ett kompletterande dokument för TI-Nspire™-referensguide och TI-Nspire™ CAS-referensguide. Alla TI-Nspire™ CX II-kommandon kommer att inkorporeras och publiceras i version 5.1 av TI-Nspire™-referensguide och TI-Nspire™ CAS-referensguide.

Grafikprogrammering

Nya kommandon för grafikprogrammering har lagts till för TI-Nspire™ CX II-handenheter och TI-Nspire™-datorprogramvara.

TI-Nspire™ CX II-handenheterna kommer att byta till detta grafikläge samtidigt som de kör grafikkommandon och byter tillbaka till kontexten i vilken programmet kördes efter slutförande av programmet.

"Kör..." kommer att visas i skärmens övre del medan programmet körs. "Avslutad" kommer att visas när programmet slutförs. Valfri knapptryckning kommer att få systemet att gå ut ur grafikläget.

- Övergången till grafikläge triggas automatiskt när en av kommandona för Rita (grafik) påträffas under körning av TI-Basicprogrammet.
- Denna övergång kommer endast att ske när ett program körs från Räknare-appen i ett dokument eller från Beräkna i Scratchpad.
- Övergången ut ur grafikläge sker vid avslutande av programmet.
- Grafikläget är endast tillgängligt på TI-Nspire™ CX II-handenheter och handenhetsvyn på TI-Nspire™ CX II CAS-programvara. Detta innebär att det inte är tillgängligt i datordokumentsvyn eller PublishView (.tnsp) på datorn eller på iOS.
 - Om ett grafikkommando påträffas medan ett TI-Basic-program körs från inkorrekt kontext så kommer ett felmeddelande att visas och TI-Basic-programmet avslutas.

Grafikskärm

Grafikskärmen kommer att innehålla en rubrik på skärmens övre del som inte kan åstadkommas av grafikkommandon.

Grafikskärmens ritområde kommer att rensas (färg = 255,255,255) när grafikskärmen initialiseras.

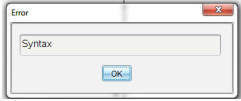
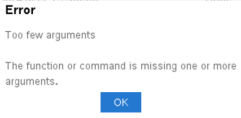
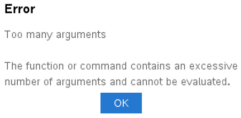
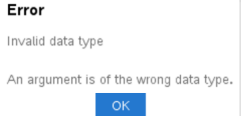
Grafikskärm	Förval
Höjd	212
Bredd	318
Färg	vit: 255,255,255

Standardvy och inställningar

- Statusikonerna i skärmens övre del (batteristatus, tryck-för-test-status, nätverksindikator osv.) kommer inte att synas när ett grafikprogram körs.
- Standardfärg för att rita: Svart (0,0,0)
- Standardstil på penna - normal, mjuk
 - Tjocklek: 1 (tunn), 2 (normal), 3 (tjockast)
 - Stil 1 (mjuk), 2 (prickad), 3 (streckad)
- Alla kommandon för att rita kommer att använda nuvarande färg och penninställningar; antingen standardvärden eller de som ställts in via TI-Basic-kommandon.
- Typsnitt är bestämt och kan inte ändras.
- Varje utmatning till grafikskärmen kommer att ritas inom ett klippfönster som är storleken av grafikskärmens ritområde. Varje ritad utmatning som sträcker sig utanför detta klippta ritområde för grafikskärmen kommer inte att ritas. Inget felmeddelande kommer att visas.
- Alla x- och y-koordinater specificerade för kommandon för att rita är definierade på så sätt att 0,0 är det vänstra övre hörnet på ritområdet för grafikskärmen.
 - **Undantag:**
 - **DrawText** använder koordinaterna i nedre vänstra hörnet av begränsningsrutan för texten.
 - **SetWindow** använder koordinaterna i skärmens nedre vänstra hörn
- Alla parametrar för kommandona kan ges som uttryck som värderas till ett värde som sedan avrundas till närmsta heltal.

Felmeddelanden på grafikskärmen

Om valideringen misslyckas så kommer ett felmeddelande att visas.

Felmeddelande	Beskrivning	Visa
Fel Syntax	Om syntaxkontrollen hittar syntaxfel visar den ett felmeddelande och försöker placera markören nära det första felet så att du kan korrigera det.	
Fel Too few arguments (För få argument)	Funktionen eller kommandot saknar ett eller flera argument	
Fel Too many arguments (För många argument)	Funktionen eller kommandot innehåller för många argument och kan inte utvärderas.	
Fel Invalid data type (Ogiltig datatyp)	Ett argument är av fel datatyp.	

Ogiltiga kommandon i grafikläge

Vissa kommandon är inte tillåtna då programmet växlat till grafikläge. Om dessa kommandon påträffas i grafikläge så kommer ett fel visas och programmet kommer att avslutas.

Otillåtet kommando	Felmeddelande
Begär	Förfrågan kan inte utföras i grafikläge
BegärStr	RequestStr kan inte utföras i grafikläge
Text	Text kan inte utföras i grafikläge

Kommandona som skriver ut text till Räknare-appen - **disp** och **dispAt** - kommer att stödja kommandon i grafikkontexten. Texten från dessa kommandon kommer att skickas till Räknare-skärmen (inte på Grafik) och kommer att synas efter att programmet avslutas och systemet byter tillbaka till Räknare-appen

Rensa

Katalog > 
CXII**Rensa x , y , *bredd*, *höjd***

Rensar hela skärmen om inga parametrar är specificerade.

Om x , y , *bredd* och *höjd* är specificerade så kommer rektangeln definierad av parametrarna att rensas.

Rensa

Rensar hela skärmen

Rensa 10,10,100,50

Rensar ett rektangelområde med övre vänster hörn (10, 10) och med bredd 100, höjd 50

DrawArc

Katalog > 
CXII**DrawArc** *x, y, bredd, höjd, startAngle, arcAngle*

Rita en båge inom den definierade avgränsande rektangeln med de tillhandahållna start- och bågvinklarna.

x, y: övre vänstra koordinaten i avgränsande rektangel

bredd, höjd: dimensioner i avgränsande rektangel

"Bågvinkeln" definierar bågens kurva.

Dessa parametrar kan ges som uttryck som värderas till ett värde som sedan avrundas till närmsta heltal.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



Se även: [FillArc](#)

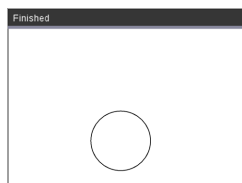
DrawCircle

Katalog > 
CXII**DrawCircle** *x, y, radie*

x, y: koordinat i mittpunkt

radie: cirkelns radie

DrawCircle 150,150,40



Se även: [FillCircleI](#)

DrawLine

Katalog > 
CXII

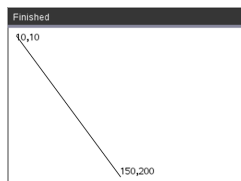
DrawLine *x1, y1, x2, y2*

Rita en linje från *x1, y1, x2, y2*.

Uttryck som värderas till ett värde som sedan avrundas till närmsta heltal.

Skärmgränser: Om de specificerade koordinaterna orsakar att någon del av linjen ritas utanför grafikskärmen så kommer den delen av linjen att klippas av och inget felmeddelande kommer att visas.

DrawLine 10,10,150,200



DrawPoly

Katalog > 
CXII

Kommandona har två varianter:

DrawPoly *xlist, ylist*

eller

DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*

Obs: DrawPoly *xlist, ylist*

Form kommer att binda samman *x1, y1* to *x2, y2, x2, y2* till *x3, y3* och så vidare.

Obs: DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*

xn, yn kommer **INTE** automatiskt att bindas samman till *x1, y1*.

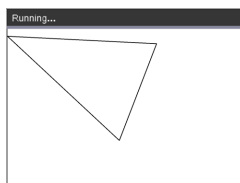
Uttryck som utvärderas till en lista med reella flyttal
xlist, ylist

Uttryck som utvärderas till ett reellt flyttal
x1, y1...xn, yn = koordinater för polygonens hörn

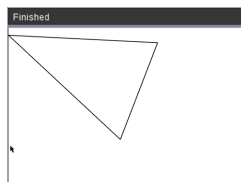
xlist:={0,200,150,0}

ylist:={10,20,150,10}

DrawPoly *xlist, ylist*



DrawPoly 0,10,200,20,150,150,0,10



Obs: DrawPoly: Inmatningens storleksdimensioner (bredd/höjd) i förhållande till ritade linjer. Dessa linjer är ritade i en begränsningsruta runt de specificerade koordinaterna och dimensionerna på så sätt att den faktiska storleken på den ritade polygonen kommer att vara större än bredden och höjden.

Se även: [FillPoly](#)

DrawRect

DrawRect *x, y, bredd, höjd*

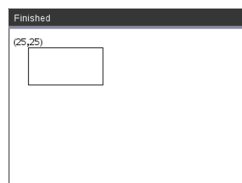
x, y: övre vänstra koordinaten i rektangeln

bredd, höjd: rektangelns bredd och höjd (rektangel ritad nedåt och höger från startkoordinaten).

Obs: Dessa linjer är ritade i en begränsningsruta runt de specificerade koordinaterna och dimensionerna på så sätt att den faktiska storleken på den ritade rektangeln kommer att vara större än vad bredden och höjden indikerar.

Se även: [FillRect](#)

DrawRect 25,25,100,50



DrawText

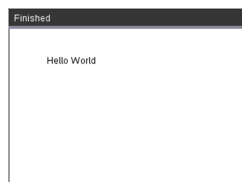
DrawText *x, y, exprOrString1*
[,exprOrString2]...

x, y: koordinat för textutmatning

Ritar texten i *exprOrString* vid specificerad *x, y*-koordinatplats.

Reglerna för *exprOrString* är samma som för **Disp** - **DrawText** kan ta flera argument.

DrawText 50,50,"Hej världen"



FillArc

Katalog > 
CXII

FillArc *x, y, bredd, höjd, startVinkel, bågeVinkel*

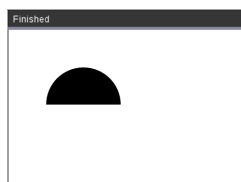
x, y: övre vänstra koordinaten i avgränsande rektangel

Rita och fyll en båge inom den definierade avgränsande rektangeln med de tillhandahållna start- och bågvinklarna.

Standardfyllningsfärgen är svart. Fyllningsfärgen kan ställas in med [SetColor](#)-kommandot

"Bågvinkeln" definierar bågens kurva

FillArc 50,50,100,100,0,180



FillCircle

Katalog > 
CXII

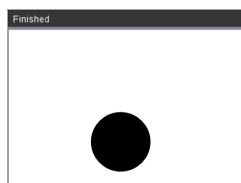
FillCircle *x, y, radie*

x, y: koordinat i mittpunkt

Rita och fyll en cirkel vid specificerad mittpunkt med den specificerade radien.

Standardfyllningsfärgen är svart. Fyllningsfärgen kan ställas in med [SetColor](#)-kommandot.

FillCircle150,150,40



Här!

FillPoly

Katalog > 
CXII

FillPoly *xlist, ylist*

eller

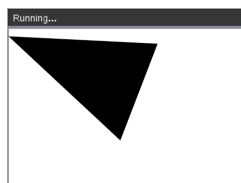
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

Obs: Linjen och färgen specificeras av [StällInFärg](#) och [StällInPenna](#)

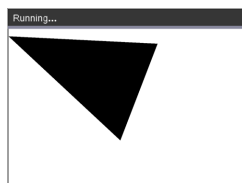
xlist:={0,200,150,0}

ylist:={10,20,150,10}

FillPoly xlist, ylist



FillPoly 0,10,200,20,150,150,0,10



FillRect

FillRect *x, y, bredd, höjd*

x, y: övre vänstra koordinaten i rektangeln

bredd, höjd: rektangelns bredd och höjd

Rita och fyll en rektangel med övre vänstra hörnet vid koordinaten specificerad av (*x,y*)

Standardfyllningsfärgen är svart.

Fyllningsfärgen kan ställas in med [SetColor](#)-kommandot

Obs: Linjen och färgen specificeras av [StällInFärg](#) och [StällInPenna](#)

FillRect 25,25,100,50



getPlatform()Katalog > 
CXII**getPlatform()**

getPlatform()

"dt"

Ger:

"dt" på applikationer för datorprogramvara

"hh" på TI-Nspire™ CX-handenheter

"ios" på TI-Nspire™ CX iPad®-app

PaintBuffer

Katalog > 
CXII

PaintBuffer

Färggrafikbuffert till skärm

Detta kommando används i samband med UseBuffer för att öka hastigheten på skärmens display när programmet genererar flera grafiska objekt.

UseBuffer

För n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

radie:=randInt(10,50)

Wait 0,5

XDrawCircle x,y,radie

EndFor

PaintBuffer

Detta program kommer att visa alla 10 cirklar på en gång.

Om "UseBuffer"-kommandot tas bort så kommer varje cirkel att visas såsom den ritas.

Se även: [UseBuffet](#)

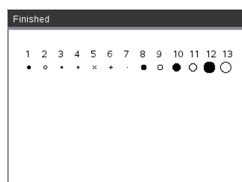
PlotXY*x, y, form**x, y*: koordinater för att plotta form*form*: ett tal mellan 1 och 13 som specificerar form

- 1 - Fylld cirkel
- 2 - Tom cirkel
- 3 - Fylld kvadrat
- 4 - Tom kvadrat
- 5 - Kors
- 6 - Plus
- 7 - Tunn
- 8 - medumpunkt, solid
- 9 - medumpunkt, tom
- 10 - större punkt, solid
- 11 - större punkt, tom
- 12 - största punkt, solid
- 13 - största punkt, tom

PlotXY 100,100,1

For *n*,1,13DrawText 1+22**n*,40,*n*PlotXY 5+22**n*,50,*n*

EndFor



SetColorKatalog > 
CXII**SetColor**

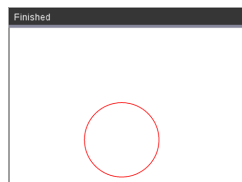
Röd-värde, grön-värde, blå-värde

Värdena för röd, grön och blå måste vara mellan 0 och 255

Ställ in färgen för efterföljande kommandon för Rita

SetColor 255,0,0

DrawCircle 150,150,100

**SetPen**Katalog > 
CXII**SetPen**

tjocklek, stil

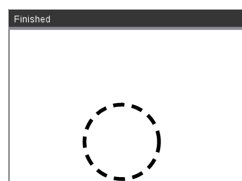
tjocklek: 1 <= tjocklek <= 3 | 1 är tunnast, 3 är tjockast

stil: 1 = Mjuk, 2 = Prickad, 3 = Streckad

Ställer in pennstilen för efterföljande kommandon för Rita

SetPen 3,3

DrawCircle 150,150,50

**SertWindow**Katalog > 
CXII**SetWindow**

xMin, xMax, yMin, yMax

Etablerar ett logiskt fönster som mappas till grafikens ritområde. Alla parametrar krävs.

Om en del av det ritade objektet är utanför fönstret så kommer utmatningen att klippas (visas ej) och inget felmeddelande kommer att synas.

SetWindow 0,160,0,120

kommer att ställa in så att utmatningsfönstret har 0,0 i nedre vänstra hörnet och en bredd på 160 och en höjd på 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

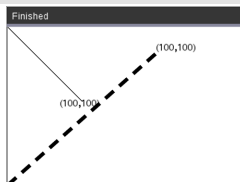
DrawLine 0,0,100,100

Om xmin är större eller lika med xmax eller om ymin är större eller lika med ymax så kommer ett felmeddelande att visas.

Alla objekt som ritats innan kommandot SetWindow kommer inte att återritas i den nya konfigurationen.

För att återställa fönsterparametrarna till standard, använd:

SetWindow 0,0,0,0



UseBuffer

Katalog > 
CXII

UseBuffer

Rita till grafikbuffert istället för skärm (för att öka prestanda)

Detta kommando används i samband med PaintBuffer för att öka hastigheten på skärmens display när programmet genererar flera grafiska objekt.

Med UseBuffer så kommer all grafik att visas endast efter att nästa PaintBuffer-kommando körs.

Kommandot UseBuffer behöver bara användas en gång i programmet, dvs varje användning av PaintBuffer behöver ingen motsvarande UseBuffer

UseBuffer

För n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

radie:=randInt(10,50)

Wait 0,5

XDrawCircle x,y,radie

EndFor

PaintBuffer

Detta program kommer att visa alla 10 cirklar på en gång.

Om "UseBuffer"-kommandot tas bort så kommer varje cirkel att visas såsom den ritas.

Se även: [PaintBuffer](#)

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.	$\square$	–
	<code>gcd(100, □)</code>	–
	<code>3+□</code>	–
	$\{5, \square, 10\} - \{3, 6, 9\}$	$\{2, \square, 1\}$

Lista argument som innehåller tomma element

Följande funktioner och kommandon ignorerar (hoppa över) tomma element som påträffas i listargument:	<code>sum({2, □, 3, 5, 6, 6})</code>	16.6
	<code>median({1, 2, □, □, 3})</code>	2
	<code>cumulativeSum({1, 2, □, 4, 5})</code>	$\{1, 3, \square, 7, 12\}$
<code>count</code> , <code>countif</code> , <code>cumulativeSum</code> , <code>freqTable</code> , <code>lista</code> , <code>frekvens</code> , <code>max</code> , <code>medelvärde</code> , <code>median</code> , <code>produkt</code> , <code>stDevPop</code> , <code>stDevSamp</code> , <code>summa</code> , <code>sumlf</code> , <code>varPop</code> och <code>varSamp</code> samt regressionsberäkningar, <code>OneVar</code> , <code>TwoVar</code> och <code>FiveNumSummary</code> statistik, konfidensintervaller och stat-tester	<code>cumulativeSum</code> $\begin{pmatrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{pmatrix}$	$\begin{bmatrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{bmatrix}$

SortA och SortD flyttar alla tomma element inom det första argumentet till botten.	$\{5, 4, 3, \square, 1\} \rightarrow list1$	$\{5, 4, 3, \square, 1\}$
	$\{5, 4, 3, 2, 1\} \rightarrow list2$	$\{5, 4, 3, 2, 1\}$
	<code>SortA list1, list2</code>	<i>Done</i>
	<code>list1</code>	$\{1, 3, 4, 5, \square\}$
	<code>list2</code>	$\{1, 3, 4, 5, 2\}$

Lista argument som innehåller tomma element

I regressionsberäkningar introducerar ett tomrum i en X- eller Y-lista ett tomrum i motsvarande element för residualen.

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD list1,list2	Done
list1	$\{5,3,2,1,_{-}\}$
list2	$\{5,3,2,1,4\}$
$I1:=\{1,2,3,4,5\}; I2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx I1,I2	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	$\{1,_,1,1,1,1\}$

En utelämnad kategori i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
$cat:=\{"M", "M", "F", "F"\}; incl:=\{"F"\}$	$\{"F"\}$
LinRegMx I1,I2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1,1\}$

En frekvens på 0 i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx I1,I2, $\{1,0,1,1\}$	Done
stat.Resid	$\{0.069231,_, -0.276923, 0.207692\}$
stat.XReg	$\{1,_,4,5\}$
stat.YReg	$\{2,_,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1\}$

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 `sqr t (6)` på inmatningsraden. När du trycker på `enter` ändras uttrycket `sqr t (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 (...)
$\sqrt{()}$	sqr t (...)
$\Sigma()$ (mallen Summa)	sumSeq (...)
$\Pi()$ (mallen Produkt)	prodSeq (...)
$\sin^{-1}()$, $\cos^{-1}()$, ...	arcsin (...), arccos (...), ...
$\Delta\text{List}()$	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:
$^{\circ}$ (grader)	@d
g (nygrader)	@g
$\angle$ (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™ Inläringsteknologi 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	Logiskt 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 operatoren (#) 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 $10 \rightarrow r$ och " r " $\rightarrow s1$ erhålls $\#s1=10$.

Postoperatorer

Postoperatorer är operatörer som kommer direkt efter ett argument, t.ex. $5!$, 25% eller $60^\circ 15' 45''$. 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 2^4 (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 $\boxed{-}$ följt av talet. Postoperationer och exponentberäkningar utförs före negationer. Till exempel är resultatet av $-x^2$ 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 ("|")-operatoren begränsning ger en uppsättning av begränsningar som påverkar utvärderingen av argumentet som föregår operatoren.

TI-Nspire CX II - TI-Basic programmeringsfunktioner

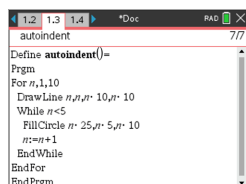
Autoindentering i Programeditor

Programeditorn för TI-Inspire™ autoindenterar nu satser inuti ett blockkommando.

Blockkommandon är If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

Editorn förbereder automatiskt utrymmen för programkommandon inom ett blockkommando. Blockets stängningskommando kommer att justeras tillsammans med öppningskommandot.

Exemplet nedan visar autoindentering i nästlade blockkommandon.



Kodfragment som har kopierats och klistrats in kommer att behålla ursprungsindenteringen.

Öppning av program som skapats i tidigare version av programvaran kommer att behålla ursprungsindenteringen.

Förbättrade felmeddelanden för TI-Basic

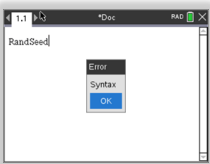
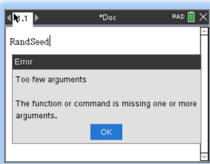
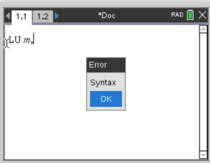
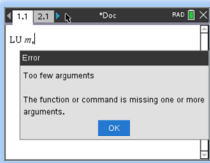
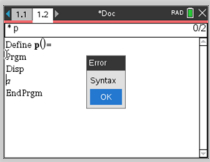
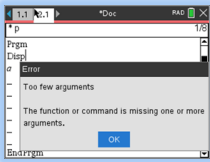
Fel

Feltillstånd	Nytt meddelande
Fel i villkorssats (If/While)	En villkorssats löstes inte till SANT eller FALSKT OBS: Med ändringen att flytta markören på raden med felet så behöver vi inte längre specificera om felet är i en "If"-sats eller i en "While"-sats.
Saknar EndIf	Förväntade EndIf men hittade en annan end-sats
Saknar EndFor	Förväntade EndFor men hittade en annan end-sats
Saknar EndWhile	Förväntade EndWhile men hittade en annan endsats
Saknar EndLoop	Förväntade EndLoop men hittade en annan endsats

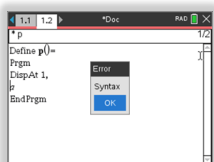
Feltillstånd	Nytt meddelande
Saknar EndTry	Förväntade EndTry men hittade en annan end-sats
"Then" utelämnad efter If <condition>	Saknar If...Then
"Then" utelämnad efter Elseif <condition>	Then saknas i block: Elseif .
När " Then ", " Else " och " Elseif " påträffades utanför kontrollblock	Else ogiltigt utanför block: If..Then..EndIf eller Try..EndTry
"Elseif" syns utanför " If..Then..EndIf "-block	Elseif ogiltigt utanför block: If...Then...EndIf
"Then" syns utanför " If....EndIf "-block	Then ogiltigt utanför block: If..EndIf

Syntaxfel

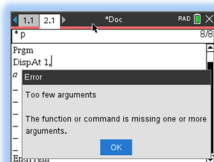
Om kommandon som förväntar sig ett eller flera argument anropas med en ofullständig lista över argument så kommer ett "**För få argument-fel**" att utfärdas istället för ett "**syntax**"-fel

Aktuellt uppträdande	Nytt CX II-uppträdande
	
	
	

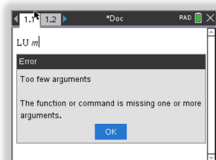
Aktuellt uppträdande



Nytt CX II-uppträdande



Obs: När en ofullständig lista över argument inte följs av ett kommatecken så är felmeddelandet: "för få argument". Detta är samma som för tidigare versioner.



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	Molvolyt	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 161.

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 sökindervallet.
180	Avsluta Tangenten  eller  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 $a+1 \rightarrow a$, där a är en odefinierad variabel, detta fel.
200	Invalid constraint (Ogiltig begränsning) Som exempel skulle $\text{solve}(3x^2-4=0,x) \mid x<0 \text{ eller } x>5$ 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 $\text{rand}(0)$ 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 (Resursutmattnings)
673	Resource exhaustion (Resursutmattnings)
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 grafitning)
860	Recursion too deep (Rekursjonen 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. (Operatörn 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 <code>xxx\yyy</code> , där: - Delen <code>xxx</code> kan ha 1 till 16 tecken. - Delen <code>yyy</code> 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, <code>:=</code> eller <code>sto</code> →. - 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 vinkelägena 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 169.

Varningskod	Meddelande
10000	Operationen kan ge falska lösningar. Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
10001	Derivering av en ekvation kan ge en falsk ekvation.
10002	Tvivelaktig lösning Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
10003	Tvivelaktig noggrannhet Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
10004	Operationen kan förlora lösningar. Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
10005	cSolve kan ange fler nollställen.
10006	Solve kan ange fler nollställen. Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
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) Försök att använda grafiska metoder för att verifiera resultaten, om tillämpligt.
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 ersätts med 1
10014	undef^0 ersätts med 1
10015	1^∞ eller 1^undef ersätts med 1

Varningskod	Meddelande
10016	$1^{\wedge}undef$ ersattes med 1
10017	För stort värde ersattes med ∞ eller $-\infty$
10018	Operationen kräver och ger ett 64-bitars värde.
10019	Resursutmattnig, 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.
10022	En lösning kan vara att ange lämpliga övre och undre gränser.
10023	Skalär har multiplicerats med enhetsmatris.
10024	Resultat uppnått med approximerad aritmetik.
10025	Ekvivalens kan inte verifieras i EXAKT läge.
10026	Begränsning kan ignoreras. Ange begränsning i formen "Variable MathTestSymbol Constant" eller en sammanslagning av dessa former, t.ex. " $x < 3$ and $x > 12$ "

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.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Innehållsförteckning

$\wedge$, potens	181
$ $	
$ $, operatörn begränsning	197
$+$	
$+$, addera	178
$/$	
$/$, dividera[*]	180
$=$	
$\neq$, inte lika med[*]	185
$=$, lika med	184
$>$	
$>$, större än	186
Π	
Π , produkt[*]	190
Σ	
$\Sigma()$, summa[*]	191
$\Sigma\text{Int}()$	192
$\Sigma\text{Prn}()$	192
$\sqrt{}$	
$\sqrt{}$, kvadratroten[*]	190
$\int$	
$\int$, integral[*]	190
$\leq$	
$\leq$, mindre än eller lika med	186
$\geq$	
$\geq$, större än eller lika med	187
$\wedge$	
$\wedge^{-1}$, inverterat värde	197
$\cdot$	
$\cdot$, punkt subtraktion	182
$\cdot$, punkt multiplikation	183
$\cdot$, punkt division	183
$\cdot$, punkt potens	183
$\cdot$, punkt addition	182
$:$	
$:$, tildela	199
$\wedge$	
$\wedge^{-1}$, inverterat värde	197
$\cdot$, minutnotation	195
$-$	
$-$, subtrahera[*]	178
$!$	
$!$, fakultet	188
$"$	
$"$, sekundnotation	195
$\#$	
$\#$, indirection	193
$\#$, indirection, operator	221
$\%$	
$\%$, procent	184
$\&$	
$\&$, lägg till	188
$*$	
$*$, multiplicera	179

►		1	
►, konvertera till nygradvinkel[Grad]	69	10 [^] (), tiopotens	197
►approxFraction()	13	2	
►Base10, visa som decimalt heltal[Bas 10]	17	2-sampel F Test	58
►Base16, visa som hexadecimal[Bas 16]	18	A	
►Base2, visa som binär[Bas 2]	16	abs(), absolutbelopp	7
►Cylind, visa som cylindrisk vektor [Cylind]	35	absolutbelopp	3-4
►DD, visa som decimal vinkel[DD]	36	mall	178
►Decimal, visa resultat som decimal [Decimal]	36	addera, +	7, 15
►DMS, visa som grad/minut/sekund [DMS]	43	amorteringstabell, amortTbl()	7, 15
►Polar, visa som polär vektor[Polär]	113	amortTbl(), amorteringstabell	8
►Rad, omvandla till radianer	121	and, Booleska operatörer	6
►Rect, visa som ortogonal vektor	124	andrerivata	9
►Sphere, visa som sfärisk vektor[sfär]	148	mall för	69
→		angle(), vinkel	10
→, lagra	199	annars, Else	9
⇒		ANOVA 2-vägs, 2-vägs variansanalys	12
⇒, logisk implikation[*]	187, 218	ANOVA, 1-vägs variansanalys	35
⇔		Ans, sista svar	12
⇔, logisk dubbel implikation[*]	188	antal dagar mellan datum, dbd()	35
©		approx(), ungefärlig	12
©, kommentar	200	approxRational()	13
°		arccos()	13
°, grader/minuter/sekunder[*]	195	arccosh()	13
°, gradnotation[*]	195	arccosinus, cos ⁻¹ ()	26
0		arccot()	13
Ob, binär indikator	200	arccoth()	13
Oh, hexadecimal indikator	200	arccsc()	13
		arccsch()	14
		arcsec()	14
		arcsech()	14
		arcsin()	14
		arcsinh()	14
		arcsinus, sin ⁻¹ ()	144
		arctan()	14
		arctangens, tan ⁻¹ ()	155
		arctanh()	14
		argument i TVM-funktioner	164
		augment(), sammanfoga/slå ihop	14

avgRC(), genomsnittlig ändringsfrekvens	15	χ^2 Pdf()	22
avrunda, round()	133	ClearAZ	22
avsluta om, EndIf	69	ClrErr, rensa fel	22
avsluta om, EndIf	69	colAugment	23
avsluta, Exit	49	colDim(), matriskolumndimension ..	23
B		colNorm(), matrixskolumnnorm ...	23
Begär	127	conj(), komplexkonjugat	24
beräkning, ordning vid	220	constructMat(), konstruera matris ..	24
bestämd integral mall	6	corrMat(), korrelationsmatris	25
bibliotek skapa genvägar till objekt	78	$\cos^{-1}$, arccosinus	26
binomCdf()	18, 74	cos(), cosinus	25
binomPdf()	19	$\cosh^{-1}()$, hyperbolisk arccosinus ...	28
binär indikator, Ob	200	cosh(), hyperbolisk cosinus	27
visa, ►Base2	16	cosinus, cos()	25
blandade bråk, använda propFrac()		$\cot^{-1}()$, arccotangens	29
med	117	cot(), cotangens	28
Booleska operatorer		cotangens, cot()	28
$\Rightarrow$	187, 218	$\coth^{-1}()$, hyperbolisk arccotangens ..	29
$\Leftrightarrow$	188	$\coth()$, hyperbolisk cotangens	29
and	8	count(), räkna poster i en lista	30
eller	110	countif(), villkorligt räkna poster i en lista	30
icke	106	cPolyRoots()	31
negerad	100	crossP(), vektorprodukt	31
nor	104	$\csc^{-1}()$, invers cosekant	32
xor	171	csc(), cosekant	32
bråk		$\operatorname{csch}^{-1}()$, invers hyperbolisk cosekant	33
mall	1	csch(), hyperbolisk cosekant	33
propFrac	117	CubicReg, kubisk regression	33
C		cumulativeSum(), kumulativ summa	34
Cdf()	53	Cycle, cykel	35
ceiling(), tak	19	cykel, Cycle	35
centralDiff()	19	D	
char(), teckensträng	20	d(), förstaderivata	189
χ^2 2way	20	days between dates (dagar mellan datum), dbd()	35
χ^2 Cdf()	21	dbd(), days between dates (dagar mellan datum)	35
χ^2 GOF	21	decimal heltalsvisning, ►Base10	17
		vinkelvisning, ►DD	36
		Define	37

Define LibPriv	38	ekvationssystem (N ekvationer)	
Define LibPub	38	mall	3
define, Define	37	eller (boolesk), eller	110
Define, define	37	eller, Boolesk operator	110
defining		else if, Elseif	46
private function or program ...	38	Elseif, else if	46
public function or program	38	end	
dela heltal, intDiv()	73	for, EndFor	55
delmatris, subMat()	154	funktion, EndFunc	58
deltaList()	39	end function, EndFunc	58
DelVar, ta bort variabel	39	EndTry, slut försök	160
delVoid(), ta bort tomma element .	40	EndWhile, slut medan	170
derivata		enhetsvektor, unitV()	167
förstaderivata, d()	189	envariabelstatistik, OneVar	108
numerisk derivata, nDeriv() ...	103	EOS (Ekvationsoperativsystem)	220
numerisk derivata, nDerivative()	102	euler(), Euler function	47
det(), matrisdeterminant	40	Exit, avsluta	49
diag(), diagonal matris	41	exp(), e till en potens	50
dim(), dimension	41	exponent, E	194
dimension, dim()	41	exponenter	
Disp, visa data	41, 136	mall	1
DispAt	42	exponentiell regression, ExpReg	51
dividera, /	180	expr(), sträng till uttryck	51
dotP(), skalärprodukt	44	ExpReg, exponentiell regression	51

E

e exponent	
mall	2
e till en potens, e^()	44, 50
E, exponent	194
e^(), e till en potens	44
eff), konvertera nominell till effektiv	
ränta	45
effektiv ränta, eff()	45
egentligt bråk, propFrac	117
egenvektor, eigVc()	45
egenvärde, eigVl()	46
eigVc(), egenvektor	45
eigVl(), egenvärde	46
Ekvationsoperativsystem (EOS)	220
ekvationssystem (2 ekvationer)	
mall	3

F

factor(), faktor	52
faktor, factor()	52
fakultet, !	188
fel och felsökning	
flytta fel, PassErr	112
rensa fel, ClrErr	22
Felkoder och meddelanden	235
Fill, fylla matris	53
finansiella funktioner, tvmlFV()	163
finansiella funktioner, tvml()	163
finansiella funktioner, tvmlN()	164
finansiella funktioner, tvmlPmt() ...	164
finansiella funktioner, tvmlPV()	164
FiveNumSummary	53
floor(), golv	54
flytta fel, PassErr	112
For	55

for, For	55	Goto, gå till	69
For, for	55	grader/minuter/sekunder	195
format(), formatsträng	55	gradnotation, °	195
formatsträng, format()	55	grupper, låsa och låsa upp	87, 167
fpart(), funktionsdel	56	grupper, testa låsstatus	65
freqTable()	56	gå till, Goto	69
frequency()	57		
Frobenius norm, norm()	105	H	
Func, funktion	58	heltal, int()	72
Func, programfunktion	58	heltalsdel, iPart()	75
functions		hexadecimal	
user-defined	37	indikator, Oh	200
funktioner		visa, ▶Base16	18
del, fpart()	56	hyperbolisk	
programfunktion, Func	58	arccosinus, cosh ⁻¹ ()	28
funktioner och variabler		arcsinus, sinh ⁻¹ ()	145
kopiera	24	arctangens, tanh ⁻¹ ()	156
füll	208-209	cosinus, cosh()	27
förstaderivata		sinus, sinh()	145
mall för	5	tangens, tanh()	156
försök, Try	160	hämta/ge	
		nämnare, getDenom()	61
G		tal, getNum()	67
g, nygrader	194	variabelinformation, getVarInfo(	
gcd(), största gemensamma delare ..	59	)	65, 68
genomsnittlig ändringsfrekvens,		höger, right()	130
avgRC()	15	höger, right()	73
geomCdf()	59		
geomPdf()	60	I	
Get	60, 210	icke, Booleska operatorer	106
getDenom(), hämta/ge nämnare ..	61	identitetsmatris, identity()	69
getKey()	61	identity(), identitetsmatris	69
getLangInfo(), hämta/ge		If, om	69
språkinformation	65	ifFn()	71
getLockInfo(), testar låsstatus hos		imag(), imaginärdel	71
en variabel eller		imaginärdel, imag()	71
variabelgrupp	65	indirection, #	193
getMode(), hämta lägesinställningar	66	indirection, operator (#)	221
getNum(), hämta/ge tal	67	inom sträng, inString()	72
GetStr	67	inString(), inom sträng	72
getType(), get type of variable	67	inställningar, hämta aktuella	66
getVarInfo(), hämta/ge		int(), heltal	72
variabelinformation	68	intDiv(), dela heltal	73
golv, floor()	54		

inte lika med, $\neq$	185	Lbl, märka	77
integral, $\int$	190	lcm, minsta gemensamma multipel	77
interpolate(), interpolera	73	left(), vänster	77
invers kumulativ normalfördelning, invNorm()	75	LibPriv	38
invers, $\wedge^{-1}$	197	LibPub	38
inverterat värde, $\wedge^{-1}$	197	libShortcut(), skapa genvägar till biblioteksobjekt	78
invF()	74	lika med, =	184
invNorm(), invers kumulativ normalfördelning	75	linjär regression, LinRegAx	80
invt()	75	linjär regression, LinRegBx	78, 81
Inv χ^2 ()	73	LinRegBx, linjär regression	78
iPart(), heltalsdel	75	LinRegMx, linjär regression	80
irr(), internränta internal rate of return, irr()	76	LinRegtIntervals, linjär regression ..	81
isPrime(), primtest	76	LinRegtTest	82
isVoid(), testa om sant	76	linSolve()	84
K		Δ list(), listskillnad	84
kombinationer, nCr()	101	list►mat(), lista till matris	85
Kommandot Wait	168	lista till matris, list►mat()	85
kommentar, ©	200	lista, räkna poster	30
komplext konjugat, conj()	24	lista, villkorligt räkna poster	30
konstruera matris, constructMat() ..	24	listor	
konvertera 4Grad	69	kumulativ summa, cumulativeSum()	34
kopiera variabel eller funktion, CopyVar	24	lista till matris, list►mat()	85
korrelationsmatris, corrMat()	25	matris till lista, mat►list()	92
kortkommandon	218	maximum, max()	92
kortkommandon, tangentbord	218	med tomma element	216
kubisk regression, CubicReg	33	minimum, min()	95
kumulativ summa, cumulativeSum() ..	34	mittsträng, mid()	95
kvadratisk regression, QuadReg	118	ny, newList()	102
kvadratroten mall	1	produkt, product()	117
kvadratroten, $\sqrt{}$	148, 190	sammanfoga/slå ihop, augment()	14
kvartär regression, QuartReg	120	skalärprodukt, dotP()	44
L		skillnad, @list()	84
lagra symbol, &	199	skillnader i en lista, @list()	84
		sortera fallande, SortD	148
		sortera stigande, SortA	147
		summering, sum()	153
		vektorprodukt, crossP()	31
		ln(), naturlig logaritm	85
		LnReg, logaritmisk regression	86
		Local, lokal variabel	87

Log		stegvis funktion (2 steg)	2
mall	2	stegvis funktion (N steg)	3
logaritmer	85	summa (G)	5
logaritmisk regression, LnReg	86	mat►list(), matris till lista	92
logisk dubbel implikation, $\Leftrightarrow$	188	matris (1 × 2)	
logisk implikation, $\Rightarrow$	187, 218	mall	4
Logistic, logistisk regression	88	matris (2 × 1)	
LogisticD, logistisk regression	89	mall	4
logistisk regression, Logistic	88	matris (2 × 2)	
logistisk regression, LogisticD	89	mall	4
lokal variabel, Local	87	matris (m × n)	
lokal, Local	87	mall	4
Loop, slinga	91	matris till lista, mat►list()	92
LU, matris undre-övre uppdelning ..	91	matriser	
läsa upp variabler eller		delmatris, subMat()	154
variabelgrupper	167	determinant, det()	40
läsa variabler eller variabelgrupper ..	87	diagonal, diag()	41
Läsa, läsa variabel eller variabelgrupp	87	dimension, dim()	41
lägen		egenvektor, eigVc()	45
inställning, setMode()	139	egenvärde, eigVl()	46
lägesinställningar, getMode()	66	fylla, Fill	53
lägg till, &	188	identitet, identity()	69
längd på sträng	41	kolumndimension, colDim() ...	23
		kolumnnorm, colNorm()	23
		kumulativ summa,	
		cumulativeSum()	34
		lista till matris, list►mat()	85
		matris till lista, mat►list()	92
		maximum, max()	92
		minimum, min()	95
		ny, newMat()	103
		produkt, product()	117
		punkt addition, .+	182
		punkt division, .P	183
		punkt multiplikation, .*	183
		punkt potens, .^	183
		punkt subtraktion, .N	182
		QR-faktorisering, QR	118
		radaddition, rowAdd()	133
		raddimension, rowDim()	134
		radmultiplikation och addition,	
		mRowAdd()	97
		radnorm rowNorm()	134
		radoperation, mRow()	97
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		

reducerad radtrappstegsform, rref()	134	N	
sammanfoga/slå ihop, augment()	14	n:te rot	
slump, randMat()	123	mall	1
summering, sum()	153	nand, Boolesk operator	100
transponera, T	154	naturlig logaritm, ln()	85
trappstegsform, ref()	125	nCr(), kombinationer	101
undermatris, subMat()	152	nDerivative(), numerisk derivata ...	102
undre-övre uppdelning, LU	91	negation, skriva in negativa tal	221
växla matrisrad, rowSwap() ...	134	nettovärde, npv()	107
max(), maximum	92	newList(), ny lista	102
maximum, max()	92	newMat(), ny matris	103
mean(), medelvärde	93	nfMax(), numeriskt funktionsmaximum	103
med,	197	nfMin(), numeriskt funktionsminimum	103
medan, While	170	nInt(), numerisk integral	104
medelvärde, mean()	93	nom), konvertera effektiv till nominell ränta	104
median(), median	93	nominell ränta, nom()	104
median, median()	93	nor, Booleska operatorer	104
medium-medium-linjeregression, MedMed	94	norm(), Frobenius norm	105
MedMed, medium-medium- linjeregression	94	normCdf()	105
mid(), mittsträng	95	normPdf()	106
min(), minimum	95	nPr(), permutationer	107
mindre än eller lika med, {	186	npv(), nettovärde	107
minimum, min()	95	nSolve(), numerisk lösning	108
minsta gemensamma multipel, lcm .	77	numeric derivata, nDeriv()	103
minutnotation,	195	numerisk derivata, nDeriv()	103
mirr(), modifierad internränta	96	derivata, nDerivative()	102
mittsträng, mid()	95	integral, nInt()	104
mod(), modul	97	lösning, nSolve()	108
modifierad internränta, mirr()	96	ny	
modul, mod()	97	lista, newList()	102
mRow(), matrisradoperation	97	matris, newMat()	103
mRowAdd(), matrisradmultiplikation och addition	97	nygradsnotation, g	194
Multipelt linjärt regression-t-test ...	99	när, when()	170
multiplicera, *	179	O	
MultReg	97	objekt	
MultRegIntervals()	98	skapa genvägar till bibliotek	78
MultRegTests()	99	om, lf	69
märka, Lbl	77		

omvandla			
►Rad	121		
OneVar, envariabelstatistik	108		
operatorer			
ordning vid beräkning	220		
operatorn begränsning " "	197		
operatorn begränsning, ordning vid			
utvärdering	220		
ord(), numerisk teckenkod	111		
ortogonalvektorvisning, ►Rect	124		
fördelningsfunktioner			
binomCdf()	18, 74		
binomPdf()	19		
χ^2 2way()	20		
invNorm()	75		
invt()	75		
Inv χ^2 ()	73		
normCdf()	105		
normPdf()	106		
χ^2 Cdf()	21		
poissCdf()	113		
poissPdf()	113		
tCdf()	157		
tPdf()	160		
χ^2 GOF()	21		
χ^2 Pdf()	22		
P			
P►Rx(), rektangulär x-koordinat	111		
P►Ry(), rektangulär y-koordinat	112		
PassErr, flytta fel	112		
Pdf()	56		
permutationer, nPr()	107		
piecewise()	113		
poissCdf()	113		
poissPdf()	113		
polyEval(), utvärdera polynom	114		
polynom			
slump, randPoly()	123		
utvärdera, polyEval()	114		
PolyRoots()	114		
polär			
visa vektor, ►Polar	113		
polära			
koordinater, R►Pθ()	121		
potens, ^	181		
potensregression,	114-115, 127, 129,		
PowerReg	157		
PowerReg, potensregression	115		
Prgm, definiera program	116		
primtalstest, isPrime()	76		
procent, %	184		
prodSeq()	116		
product(), produkt	117		
produkt (P)			
mall	5		
produkt, product()	117		
produkt, Π()	190		
program och programmering			
försök, Try	160		
rensa fel, ClrErr	22		
slut försök, EndTry	160		
slut program, EndPrgm	116		
visa I/O- skärm, Disp	136		
visa I/O-fönster, Disp	41		
programmera			
definiera program, Prgm	116		
flytta fel, PassErr	112		
visa data, Disp	41		
programmering			
visa data, Disp	136		
programs			
defining private library	38		
defining public library	38		
propFrac, egentligt bråk	117		
punkt			
addition, .+	182		
division, .P	183		
multiplikation, .*	183		
potens, .^	183		
subtraktion, .N	182		
Q			
QR-faktorisering, QR	118		
QR, QR-faktorisering	118		
QuadReg, kvadratisk regression	118		

QuartReg, kvartär regression	120	rest, remain()	127
R		resultat, statistik	149
R, radian	194	resultvärden, statistik	150
R*Pr(), polära koordinater	121	retur, Return	130
R*Pθ(), polära koordinater	121	Return, retur	130
radian, R	194	right(), höger	130
rand(), slumptal	122	right, right()	47, 169
randBin, slumptal	122	rita	205-207
randInt(), slumpsheltal	122	rk23(), Runge Kutta-funktion	130
randMat(), slumpmatris	123	rotate(), rotera	132
randNorm(), slumpnorm	123	rotera, rotate()	132
randPoly(), slumppolynom	123	round(), avrunda	133
randSamp()	123	rowAdd(), matrisradaddition	133
RandSeed, slumpvalsfrö	124	rowDim(), matrisradimension	134
real(), reell	124	rowNorm(), matrisradnorm	134
reducerad radtrappstegsform, rref()	134	rowSwap(), växla matrisrad	134
reell, real()	124	rref(), reducerad trappstegsform ..	134
ref(), trappstegsform	125	räkna poster i en lista villkorligt , countif()	30
RefreshProbeVars	126	räkna poster i en lista, count()	30
regression		S	
exponentiell, ExpReg	51	sammanfoga/slå ihop, augment() ..	14
kubisk, CubicReg	33	samtida ekvationer, simlut()	142
regressioner		sannolik täthet, normPdf()	106
kvadratisk, QuadReg	118	sannolikhet för normalfördelning, normCdf()	105
kvartär, QuartReg	120	sannolikhet för student-t-fördelning, tCdf()	157
linjär regression, LinRegAx	80	sec ⁻¹ (), invers sekansfunktion	135
linjär regression, LinRegBx	78, 81	sec(), sekansfunktion	135
logaritmisk, LnReg	86	sech ⁻¹ (), invers hyperbolisk sekansfunktion	136
Logistic	88	sech(), hyperbolisk sekansfunktion ..	136
logistisk, Logistic	89	sekundnotation, "	195
medium-medium-linje, MedMed	94	seq(), talföljd	137
MultReg	97	seqGen()	137
potensregression,	114-115, 127,	seqn()	138
PowerReg ..	129, 157	sequence, seq()	137-138
sinus, SinReg	146	setMode(), ställ in läge	139
rektangulär x-koordinat, P*Rx() ..	111	sfärisk vektorvisning, ►Sphere	148
rektangulär y-koordinat, P*Ry() ..	112	shift(), skifta	141
remain(), rest	127	sign(), tecken	142
rensa			
fel, ClrErr	22		
Rensa	204		
RequestStr	129		

simult(), samtidiga ekvationer	142	slump norm, randNorm()	123
sin ⁻¹ (), arcsinus	144	slumptalsfrö, RandSeed	124
sin(), sinus	143	standardavvikelse, stdDev()	150-151, 167
sinh ⁻¹ (), hyperbolisk arcsinus	145	tvåvariabelresultat, TwoVar ...	165
sinh(), hyperbolisk sinus	145	varians, variance()	168
SinReg, sinusregression	146	stdDevPop(), standardavvikelse för population	150
sinus, sin()	143	stdDevSamp(), standardavvikelse för urval	151
sinusregression, SinReg	146	stegvis funktion (2 steg) mall	2
skalär		stegvis funktion (N steg) mall	3
produkt, dotP()	44	Stoppkommando	152
skifta, shift()	141	string(), uttryck till sträng	152
slinga, Loop	91	strings	
slump		right, right()	47, 169
matris, randMat()	123	sträng	
norm, randNorm()	123	dimension, dim()	41
polynom, randPoly()	123	längd	41
slumpfrö, RandSeed	124	strängar	
slump sampel	123	använda för att skapa variabelnamn	221
slut		format, format()	55
försök, EndTry	160	formatera	55
medan, EndWhile	170	höger, right()	73, 130
program, EndPrgm	116	indirection, #	193
slinga, EndLoop	91	inom, inString	72
slut medan, EndWhile	170	lägg till, &	188
slut slinga, EndLoop	91	mittsträng, mid()	95
SortA, sortera stigande	147	numerisk teckenkod, ord()	111
SortD, sortera fallande	148	rotera, rotate()	132
sortera		skifta, shift()	141
fallande, SortD	148	sträng till uttryck, expr()	51
stigande, SortA	147	teckensträng, char()	20
språk		uttryck till sträng, string()	152
hämta språkinformation	65	vänster, left()	77
sqrt(), kvadratrot	148	ställ in	
standardavvikelse, stdDev() ..	150-151, 167	läge, setMode()	139
stat.results	149	större än eller lika med, 	187
stat.values	150	större än, >	186
statistik		största gemensamma delare, gcd() .	59
envariabelstatistik, OneVar	108	subMat(), delmatris	154
fakultet, !	188		
kombinationer, nCr()	101		
medelvärde, mean()	93		
median, median()	93		
permutationer, nPr()	107		

subMat(), undermatrix	152	tidsjusterat pengavärde, nuvärde ..	164
substitution med " " -operatorn ...	197	tidsjusterat pengavärde, ränta	163
subtrahera, -	178	TInterval, t konfidensintervall	158
sum(), summering	153	tInterval_2Samp, 2sampel t-	
sumIf()	153	konfidensintervall	159
summa (G)		tiopotens, 10^()	197
mall	5	tomma element	216
summa av kapitalbetalningar	192	tomma element, ta bort	40
summa av räntebetalningar	192	tPdf(), studentt täthetsfunktion ...	160
summa, Σ ()	191	trace()	160
summering, sum()	153	transponera, T	154
sumSeq()	154	trappstegsform, ref()	125
svar (sista), Ans	12	Try, felhanteringskommando	160
		Try, försök	160
T		tTest, t-test	161
t-test, tTest	161	tTest_2Samp, 2-sampel t-test	162
T, transponera	154	TVM-argument	164
ta bort		tvmFV()	163
tomma element från lista	40	tvmI()	163
variabel, DelVar	39	tvmN()	164
tak, ceiling()	19, 31	tvmPmt()	164
talföljd, seq()	137	tvmPV()	164
tan ⁻¹ () , arctangens	155	tvåvariabelresultat, TwoVar	165
tan(), tangens	154	TwoVar, tvåvariabelresultat	165
tangens, tan()	154	täthetsfunktion för student-t, tPdf()	160
tanh ⁻¹ () , hyperbolisk arctangens ...	156		
tanh(), hyperbolisk tangens	156	U	
tCdf(), studentt		undermatrix, subMat()	152
fördelningssannolikhet	157	ungefärlig, approx()	12
tecken		unitV(), enhetsvektor	167
numerisk kod, ord()	111	unLock, låsa upp variabel eller	
sträng, char()	20	variabelgrupp	167
tecken, sign()	142	user-defined functions	37
teckensträng, char()	20	user-defined functions and	
test for void, isVoid()	76	programs	38
Test_2S, 2-sample F test	58	uteslutning med " " -operatorn	197
Text, kommando	157	uttryck	
tidsjusterat pengavärde, antal		sträng till uttryck, expr()	51
betalningsperioder	164	utvärdera polynom, polyEval()	114
tidsjusterat pengavärde,			
betalningsbelopp	164	V	
tidsjusterat pengavärde, framtida		variabel	
värde	163	skapa namn från en	221

teckensträng		xor, Booleskt exklusivt eller	171
variabler		Z	
lokal, Local	87	zInterval, z konfidensintervall	172
rensa alla enstaka bokstäver ...	22	zInterval_1Prop, 1-proportion z- konfidensintervall	172
ta bort, DelVar	39	zInterval_2Prop, 2-proportion z- konfidensintervall	173
variabler och funktioner		zInterval_2Samp, 2-sampel z- konfidensintervall	173
kopiera	24	zTest	174
variabler, låsa och låsa upp	65, 87, 167	zTest_1Prop, 1-proportion z-test ...	175
varians, variance()	168	zTest_2Prop, 2-proportion z-test ...	176
varningskoder och meddelanden ..	235	zTest_2Samp, 2-sampel z-test	176
varPop()	167		
varSamp(), sampelvarians	168		
vektorer			
enhet, unitV()	167		
skalärprodukt, dotP()	44		
vektorprodukt, crossP()	31		
visa cylindrisk vektor, ►Cylind ..	35		
vektorprodukt, crossP()	31		
vinkel, angle()	9		
visa cylindrisk vektor, ►Cylind	35		
visa data, Disp	41, 136		
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	124		
polär vektor, ►Polar	113		
sfärisk vektor, ►Sphere	148		
visning i grad/minut/sekund, ►DMS ..	43		
void, testa om	76		
vänster, left()	77		
W			
warnCodes(), Warning codes	169		
when(), när	170		
While, medan	170		
X			
x ² , kvadrat	182		
XNOR	188		