

TI-Nspire™ CX CAS Referenshandbok

Viktigt information

Texas Instruments lämnar inga garantier, vare sig uttryckliga eller underförstådda, inklusive men inte begränsade till underförstådda garantier i fråga om säljbarhet eller lämplighet för ett speciellt syfte, rörande program eller bokmaterial och gör endast sådant material tillgängligt på en "i befintligt skick"-grund. Under inga omständigheter kommer Texas Instruments att vara skyldigt för speciella, omgivande, oavsiktliga eller följdaktiga skador i samband med eller uppkomna genom köpet eller användandet av dessa material och det enda och exklusiva åtagande som tillfaller Texas Instruments, oavsett formen av åtgärd, skall inte överstiga summan som anges i programmets licens. Texas Instruments är inte heller skadeståndsskyldigt för några krav som uppstått genom användning av denna produkt av annan användare.

© 2024 Texas Instruments Incorporated

De faktiska produkterna kan variera något från de visade bilderna.

Innehållsförteckning

Mallar för uttryck	1
Alfabetisk lista	7
A	7
B	15
C	19
D	33
E	42
F	49
G	57
I	67
L	75
M	89
N	97
O	105
P	107
Q	113
R	116
S	131
T	149
U	160
V	161
W	162
X	164
Z	165
Symboler	172
TI-Nspire™ CX II-Kommandon för att rita	191
Grafikprogrammering	191
Grafikskärm	191
Standardvy och inställningar	192
Felmeddelanden på grafikskärmen	193
Ogiltiga kommandon i grafikläge	193
C	194
D	195
F	198
G	200
P	201
S	203
U	205

Tomma element	206
Kortkommandon för att mata in matematiska uttryck	208
EOS™-hierarki (Equation Operating System)	210
TI-Nspire CX II - TI-Basic programmeringsfunktioner	212
Autoindentering i Programeditor	212
Förbättrade felmeddelanden för TI-Basic	212
Konstanter och värden	215
Felkoder och meddelanden	216
Varningskoder och meddelanden	225
Allmän information	227
Index	228

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 173.

Exempel:

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

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 173.

Exempel:

$$2^3 \qquad 8$$

Mall för Kvadratrot

ctrl **x²** tangenter



Obs: Se även $\sqrt{}$ (kvadratrot), på sidan 180.

Exempel:

Mall för N:te rot

ctrl **^** tangenter



Obs: Se även **root()**, på sidan 128.

Exempel:

e exponent mall

e^x tangent

$e^{[]}$

Exempel:

Basen för den naturliga logaritmen e upphöjd till

Obs: Se även $e^{\wedge}()$, på sidan 42.

Mall för Log

ctrl 10^x tangenter

$\log_{[]}([])$

Exempel:

$$\log_{\frac{1}{4}}(2) = 0.5$$

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

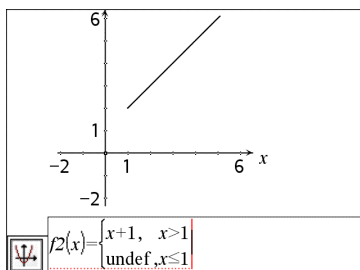
Obs: Se även $\log()$, på sidan 85.

Stegvis mall (2 steg)

Katalog > \int

$\left\{ \begin{array}{l} [] \\ [] \end{array} \right.$

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 109.

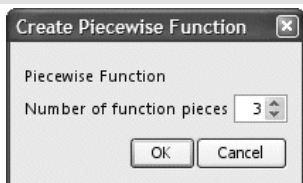
Stegvis mall (N steg)

Katalog > \int

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 109.

Mall för System med 2 ekvationer



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

Obs: Se även `system()`, på sidan 148.

Exempel:

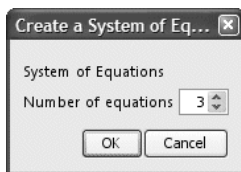
$$\text{solve} \left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y \right) \quad x = \frac{5}{2} \text{ and } y = -\frac{5}{2}$$

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

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

Mall för System med N ekvationer

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



Obs: Se även `system()`, på sidan 148.

Exempel:

Se exemplet på mall för Ekvationssystem (2 ekvationer).

Mall för Absolutbelopp



Obs: Se även `abs()`, på sidan 7.

Exempel:

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

Mall för dd°mm'ss.ss''

Katalog > 

0°00''

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.

Matrismall (2 x 2)

Katalog > 

$\begin{bmatrix} 00 \\ 00 \end{bmatrix}$

Exempel:

Skapar en 2 x 2-matris.

Matrismall (1 x 2)

Katalog > 

$[00]$

Exempel:

$\text{crossP}([1 \ 2],[3 \ 4])$ $[0 \ 0 \ -2]$

Matrismall (2 x 1)

Katalog > 

$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Exempel:

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

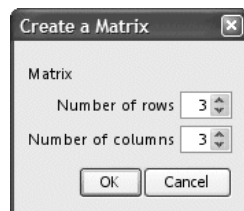
Matrismall (m x n)

Katalog > 

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

Exempel:

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



Matrismall (m x n)

Katalog > 

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

Mall för Summa (Σ)

Katalog > 

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

Exempel:

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

Obs: Se även $\Sigma()$ (**sumSeq**), på sidan 181.

Mall för Produkt (Π)

Katalog > 

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

Exempel:

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

Obs: Se även $\Pi()$ (**prodSeq**), på sidan 181.

Mall för förstaderivata

Katalog > 

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

Exempel:

Obs: Se även **d()** (**derivata**), på sidan 180.

Andraderivata, mall

Katalog > 

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

Exempel:

Obs: Se även **d()** (**derivata**), på sidan 180.

$$\int_a^b f(x) dx$$

Exempel:

Alfabetisk lista

Poster som inte är alfabetiska (till exempel, +, ! och >) listas i slutet av detta avsnitt och börjar, på sidan 172. Om inget annat anges har alla exempel i detta avsnitt utförts i det förinställda återställningsläget och alla variabler betraktas som odefinierade.

A

abs()

Katalog > 

$\text{abs}(\text{List } I) \Rightarrow \text{lista}$

$\text{abs}(\text{Matrix } I) \Rightarrow \text{matrix}$

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 > 

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

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

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

$N, I, PV, Pmt, FV, PpY, CpY$ och $PmtAt$ beskrivs i tabellen över TVM-argument, se på sidan 158.

$\text{amortTbl}(12, 60, 10, 5000, \dots, 12, 12)$

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

- 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.

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 182, och **bal()**, se på sidan 15.

and (och)

BooleanExpr1 and BooleanExpr2 ⇒ Booleskt uttryck

$$\begin{array}{ccc} x \geq 3 \text{ and } x \geq 4 & & x \geq 4 \\ \hline \{x \geq 3, x \leq 0\} \text{ and } \{x \geq 4, x \leq 2\} & & \{x \geq 4, x \leq 2\} \end{array}$$

BooleanList1 and BooleanList2 ⇒ Boolesk lista

BooleanMatrix1 and BooleanMatrix2 ⇒ Boolesk matris

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

Integer1 and Integer2 ⇒ heltal

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

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

I hexadecimalt basläge:

$$\begin{array}{ccc} 0h7AC36 \text{ and } 0h3D5F & & 0h2C16 \end{array}$$

Viktigt: Noll, inte bokstaven O.

I binärt basläge:

$$\begin{array}{ccc} 0b100101 \text{ and } 0b100 & & 0b100 \end{array}$$

I decimalt basläge:

$$\begin{array}{ccc} 37 \text{ and } 0b100 & & 4 \end{array}$$

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

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.

angle()

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:

vinkel(List1)⇒*lista*

vinkel(Matrix1)⇒*matris*

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

ANOVA

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 143.)

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

Resultatvariabel	Beskrivning
stat.df	Frihetsgrader hos grupperna
stat.SS	Kvadratsumma hos grupperna
stat.MS	Kvadratmedelvärde hos grupperna
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
stat.sp	Sammanslagen (pooled) standardavvikelse
stat.xbarlist	Medelvärdet på listornas indata
stat.CLowerList	95 % konfidensintervall för medelvärdet hos varje indatalista
stat.CUpperList	95 % konfidensintervall för medelvärdet hos varje indatalista

ANOVA2way

Katalog > 

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

Beräknar en 2-vägs variansanalys för att jämföra medelvärdena hos 2 till 10 populationer. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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
stat.MS	Kvadratmedelvärde hos kolumnfaktorn
Statistik.FBlock	F statistik för faktor

Resultatvariabel	Beskrivning
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 felet
stat.SSError	Kvadratsumma hos felet
stat.MSError	Kvadratmedelvärde hos felet
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
stat.PValInteract	Sannolikhetsvärde på interaktionen
stat.dfInteract	Frihetsgrader hos interaktionen

Resultatvariabel	Beskrivning
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)

  tangenter

Ans \Rightarrow värde

56 56



Ger resultatet på det senast beräknade uttrycket.

56+4 60
60+4 64

approx()

Katalog > 

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

Detta motsvarar att skriva in argumentet och trycka på  .

$\text{approx}\left(\frac{1}{3}\right)$ 0.333333

$\text{approx}\left(\left\{\frac{1}{3}, \frac{1}{9}\right\}\right)$ {0.333333, 0.111111}

$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$ {0., -1.}

$\text{approx}([\sqrt{2}, \sqrt{3}])$ [1.41421 1.73205]

$\text{approx}\left(\left[\frac{1}{3}, \frac{1}{9}\right]\right)$ [0.333333 0.111111]

$\text{approx}(\text{List } l) \Rightarrow \text{lista}$

$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$ {0., -1.}

$\text{approx}(\text{Matrix } l) \Rightarrow \text{matrix}$

$\text{approx}([\sqrt{2}, \sqrt{3}])$ [1.41421 1.73205]

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

approxFraction()Katalog > *List* ▶ **approxFraction**([*Tol*]) ⇒ *lista**Matrix* ▶ **approxFraction**([*Tol*]) ⇒ *matrix*

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

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

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

$$0.8333333333333333 \blacktriangleright \text{approxFraction}(5.E-14) \quad \frac{5}{6}$$

$$\{\pi, 1.5\} \blacktriangleright \text{approxFraction}(5.E-14) \quad \left\{ \frac{5419351}{1725033}, \frac{3}{2} \right\}$$

approxRational()Katalog > *List*, [*Tol*] ⇒ *lista**Matrix*, [*Tol*] ⇒ *matrix*

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

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

$$\text{approxRational}(\{0.2, 0.33, 4.125\}, 5.E-14) \quad \left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$$

arccos()Se $\cos^{-1}()$, på sidan 26.**arccosh()**Se $\cosh^{-1}()$, på sidan 27.**arccot()**Se $\cot^{-1}()$, på sidan 28.**arccoth()**Se $\coth^{-1}()$, på sidan 28.**arccsc()**Se $\csc^{-1}()$, på sidan 30.

arccsch()Se $\text{csch}^{-1}()$, på sidan 31.**arcsec()**Se $\text{sec}^{-1}()$, på sidan 131.**arcsech()**Se $\text{sech}^{-1}()$, på sidan 131.**arcsin()**Se $\text{sin}^{-1}()$, på sidan 139.**arcsinh()**Se $\text{sinh}^{-1}()$, på sidan 140.**arctan()**Se $\text{tan}^{-1}()$, på sidan 149.**arctanh()**Se $\text{tanh}^{-1}()$, på sidan 150.**augment()**Katalog > **augment(List1, List2) ⇒ lista** $\text{augment}(\{1, -3, 2\}, \{5, 4\}) \quad \{1, -3, 2, 5, 4\}$ Ger en ny lista med *List2* inlagd i slutet på *List1*.**augment(Matrix1, Matrix2) ⇒ matris**

1 2	→ m1	1 2
3 4		3 4
5	→ m2	5
6		6

Ger en ny matris med *Matrix2* fogad till *Matrix1*. När kommatecknet (,) används måste matriserna ha samma raddimensioner och *Matrix2* fogas till *Matrix1* som nya kolumner. Ändrar inte *Matrix1* eller *Matrix2*. $\text{augment}(m1, m2) \quad \begin{matrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{matrix}$

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

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

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

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

Ger differenskvoten i positiv riktning.

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

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

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

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

B

bal()

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

bal(*NPmt*,*amortTable*) \Rightarrow värde

Amorteringsfunktion som beräknar planerad balans efter en specificerad inbetalning.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 158.

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 158.

- Om du utelämnar *Pmt* används

bal (5,6,5.75,5000,,12,12)	833.11
tbt:=amortTbt (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
bal (4,tbt)	1674.27

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 $\Sigma Int()$ och $\Sigma Prn()$, på sidan 182.

►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

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

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

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

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

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

►Base10

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()

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

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 listaBeräknar en sannolikhet för den diskreta
binomialfördelningen med n antal försök och
sannolikheten p för att lyckas vid varje
försök.**C****ceiling()**Katalog > Ger det närmaste heltal som är \geq
argumentet.

<code>ceiling(.456)</code>	1.
----------------------------	----

Argumentet kan vara ett reellt eller ett
komplex tal.**Obs:** Se även **floor()**.**ceiling**(List1) \Rightarrow lista

<code>ceiling({-3.1,1.2.5})</code>	{-3.,1.3.}
------------------------------------	------------

ceiling(Matrix1) \Rightarrow matrix

<code>ceiling($\begin{pmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{pmatrix}$)</code>	$\begin{pmatrix} 0 & -3 \cdot i \\ 2. & 4 \end{pmatrix}$
--	--

Ger en lista eller matrix över taket för
varje element.**centralDiff()**Katalog > **centralDiff**(Utr1,Var [=Värde]
[,Steg]) \Rightarrow uttryck**centralDiff**(Utr1,Var
[,Steg]) | Var=Värde \Rightarrow uttryck**centralDiff**(Utr1,Var [=Värde]
[,Lista]) \Rightarrow lista**centralDiff**(Lista1,Var [=Värde]
[,Steg]) \Rightarrow lista**centralDiff**(Matrix1,Var [=Värde]
[,Steg]) \Rightarrow matrixGer 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 *Listal* eller *Matrisl* utförs operationen på värdena i listan eller matriselementen.

Obs: Se även `avgRC()`.

char()

`char(Integer)` ⇒ tecken

<code>char(38)</code>	"&"
<code>char(65)</code>	"A"

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

 χ^2 2way

χ^2 2way *ObsMatrix*

chi22way *ObsMatrix*

Beräknar ett χ^2 -test för association på 2-vägstabellen över antal i den observerade matrisen *ObsMatrix*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

För information om effekten av tomma element i en matris, se "Tomma element" (på sidan 206).

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

$\chi^2\text{Cdf}(\text{lowBound}, \text{upBound}, \text{df}) \Rightarrow \text{tal}$ 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 $P(X > \text{upBound})$, sätt $\text{lowBound} = 0$.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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

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

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 143.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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

$\chi^2\text{Pdf}(X\text{Val}, \text{df}) \Rightarrow \text{tal}$ om $X\text{Val}$ är ett tal, *lista*

om *XVal* är en lista

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

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

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

ClearAZ

ClearAZ

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

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 161.

ClrErr

ClrErr

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 108 och **Try**, på sidan 154.

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

För ett exempel på **ClrErr**, se exempel 2 under kommandot **Try**, på sidan 154.

colAugment()Katalog > **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*.

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

colDim()Katalog > **colDim**(*Matrix*) \Rightarrow *uttryck*

Ger antalet kolumner i *Matrix*.

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

Obs: Se även **rowDim()**.**colNorm()**Katalog > **colNorm**(*Matrix*) \Rightarrow *uttryck*

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

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

Obs: Odefinierade matriselement är ej tillåtna. Se även **rowNorm()**.**conj()**Katalog > **conj**(*List1*) \Rightarrow *lista***conj**(*Matrix1*) \Rightarrow *matrix*

Ger argumentets komplexkonjugat.

Obs: Alla odefinierade variabler behandlas som reella variabler.**constructMat()**Katalog > 

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

Ger en matris baserad på argumenten.

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

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*.

CopyVar

CopyVar *Var1*, *Var2*

CopyVar *Var1*., *Var2*.

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

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

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

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

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

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

<i>aa.a</i> :=45	45																
<i>aa.b</i> :=6.78	6.78																
CopyVar <i>aa</i> ., <i>bb</i> ..	Done																
getVarInfo()	<table border="1"> <tr> <td><i>aa.a</i></td> <td>"NUM"</td> <td>"⊞"</td> <td>0</td> </tr> <tr> <td><i>aa.b</i></td> <td>"NUM"</td> <td>"⊞"</td> <td>0</td> </tr> <tr> <td><i>bb.a</i></td> <td>"NUM"</td> <td>"⊞"</td> <td>0</td> </tr> <tr> <td><i>bb.b</i></td> <td>"NUM"</td> <td>"⊞"</td> <td>0</td> </tr> </table>	<i>aa.a</i>	"NUM"	"⊞"	0	<i>aa.b</i>	"NUM"	"⊞"	0	<i>bb.a</i>	"NUM"	"⊞"	0	<i>bb.b</i>	"NUM"	"⊞"	0
<i>aa.a</i>	"NUM"	"⊞"	0														
<i>aa.b</i>	"NUM"	"⊞"	0														
<i>bb.a</i>	"NUM"	"⊞"	0														
<i>bb.b</i>	"NUM"	"⊞"	0														

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

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

cos() **tangent**

cos(List1)⇒lista

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

cos(squareMatrix1)⇒kvadratMatris

I vinkelläget Radianer:

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

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

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

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

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

Forma matriserna:

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

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

$\cos(B) =$

cos() **tangent**

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

Alla beräkningar utförs med flyttalsaritmetik.

cos⁻¹() **tangent****cos⁻¹(List1)**⇒*lista*

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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.

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

$$\cos^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$$

$$\begin{bmatrix} 1.73485+0.064606 \cdot i & -1.49086+2.10514 \\ -0.725533+1.51594 \cdot i & 0.623491+0.778369 \\ -2.08316+2.63205 \cdot i & 1.79018-1.27182 \cdot i \end{bmatrix}$$

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

cosh()**Katalog** > **cosh(List1)**⇒*lista*

I vinkelläget Grader:

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

cosh(squareMatrix1)⇒*kvadratMatris*

I vinkelläget Radianer:

cosh()

Katalog >

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

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

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

cosh⁻¹()

Katalog >

cosh⁻¹(List1) ⇒ lista

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:

$$\cosh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} 2.52503+1.73485 \cdot i & -0.009241-1.4908i \\ 0.486969-0.725533 \cdot i & 1.66262+0.623491i \\ -0.322354-2.08316 \cdot i & 1.26707+1.79018i \end{bmatrix}$$

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

cot()

tangent

cot(List1) ⇒ lista

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

I vinkelläget Grader:

I vinkelläget Nygrader:

I vinkelläget Radianer:

$\cot^{-1}()$

trig **tangent**

$\cot^{-1}(List1) \Rightarrow lista$

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

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

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

$\coth()$

Katalog > 

$\coth(List1) \Rightarrow lista$

$\coth^{-1}()$

Katalog > 

$\coth^{-1}(List1) \Rightarrow lista$

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

`count()`

Katalog > 

`count(Value1orList1 [,Value2orList2 [,...]])` \Rightarrow 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 206.

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.
- 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 206.

Obs: Se även **sumlf()**, på sidan 147 och **frequency()**, på sidan 54.

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

Räknar 1 och 3.

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

Räknar 3, 5 och 7.

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

Räknar 1, 3, 7 och 9.

cPolyRoots()

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*.

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

Obs: Se även **polyRoots()**, på sidan 110.

crossP(List1, List2) ⇒ lista

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

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

crossP(Vector1, Vector2) ⇒ vektor

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

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

crossP([1 2 3],[4 5 6])	[-3 6 -3]
crossP([1 2],[3 4])	[0 0 -2]

csc()

 **tangent**

csc(List1) ⇒ lista

I vinkelläget Grader:

I vinkelläget Nygrader:

I vinkelläget Radianer:

csc⁻¹()

 **tangent**

csc⁻¹(List1) ⇒ lista

I vinkelläget Grader:

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

csc⁻¹(1) 90.

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

I vinkelläget Nygrader:

csc⁻¹(1) 100.

I vinkelläget Radianer:

csch()Katalog > **csch**(*List1*) ⇒ lista**csch⁻¹()**Katalog > **csch⁻¹**(*List1*) ⇒ lista

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 143.

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 kategorikoder 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 206).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$

Resultatvariabel	Beskrivning
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()

Katalog > 

cumulativeSum(List1)⇒lista

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

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

cumulativeSum(Matrix1)⇒matrix

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

<table border="1"> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>5</td><td>6</td></tr> </table>	1	2	3	4	5	6	→ <i>m1</i>	<table border="1"> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>5</td><td>6</td></tr> </table>	1	2	3	4	5	6
1	2													
3	4													
5	6													
1	2													
3	4													
5	6													
cumulativeSum(<i>m1</i>)		<table border="1"> <tr><td>1</td><td>2</td></tr> <tr><td>4</td><td>6</td></tr> <tr><td>9</td><td>12</td></tr> </table>	1	2	4	6	9	12						
1	2													
4	6													
9	12													

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 206.

Cycle

Katalog > 

Cycle

Funktion som listar heltal från 1 till 100 utom 50.

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

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.

D

dbd()

$dbd(date1, date2) \Rightarrow värde$

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

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

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

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

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

MM.DDYY (format som ofta används i USA)

DDMM.YY (format som ofta används i Europa)

►DD

Expr1 ►DD⇒värde

List1 ►DD⇒lista

Matrix1 ►DD⇒matris

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

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

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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

►Decimal

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

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

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

Define (Definiera)

Define *Var* = *Uttryck*

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

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

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

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

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

Define *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.

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.

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 $g(x,y)=\text{Prgm}$	
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

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

Define LibPriv

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 34 och **Define LibPub**, på sidan 36.

Define LibPub

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 34 och **Define LibPriv**, på sidan 36.

deltaList()Se Δ List(), på sidan 82.**DelVar**Katalog > **DelVar** *Var1* [, *Var2*] [, *Var3*] ...**DelVar** *Var*.

Tar bort den specificerade variabeln eller variabelgruppen från minnet.

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 161.

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 border="1"> <tbody> <tr> <td><i>aa.a</i></td> <td>"NUM"</td> <td>"[]"</td> </tr> <tr> <td><i>aa.b</i></td> <td>"NUM"</td> <td>"[]"</td> </tr> <tr> <td><i>aa.c</i></td> <td>"NUM"</td> <td>"[]"</td> </tr> </tbody> </table>	<i>aa.a</i>	"NUM"	"[]"	<i>aa.b</i>	"NUM"	"[]"	<i>aa.c</i>	"NUM"	"[]"
<i>aa.a</i>	"NUM"	"[]"								
<i>aa.b</i>	"NUM"	"[]"								
<i>aa.c</i>	"NUM"	"[]"								
DelVar <i>aa</i> .	Done									
getVarInfo()	"NONE"									

delVoid()Katalog > **delVoid**(*List1*) \Rightarrow *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 206.

det()

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

Ger determinanten för *squareMatrix*.

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

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

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

diag()

diag(*List*) \Rightarrow matrix

diag([2 4 6])	2 0 0
	0 4 0
	0 0 6

diag(*rowMatrix*) \Rightarrow matrix

diag(*columnMatrix*) \Rightarrow matrix

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

diag(*squareMatrix*) \Rightarrow radMatrix

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

4 6 8	4 6 8
1 2 3	1 2 3
5 7 9	5 7 9
diag(Ans)	4 2 9

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

Ger antalet tecken i teckensträngen *String*.

dim("Hello "&"there") 11

Disp

Katalog >

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

```
Define chars(start,end)=Prgm
  For i,start,end
  Disp i," ",char(i)
  EndFor
EndPrgm
Done
```

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

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

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

```
chars(240,243)
-----
240 ø
241 ñ
242 ò
243 ó
-----
Done
```

DispAt

Katalog >

DispAt *int,expr1* [,*expr2* ...] ...**DispAt** låter dig specificera den rad där det specifika uttrycket eller strängen kommer att visas på skärmen.

Radnumret kan specificeras som ett uttryck.

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

DispAt

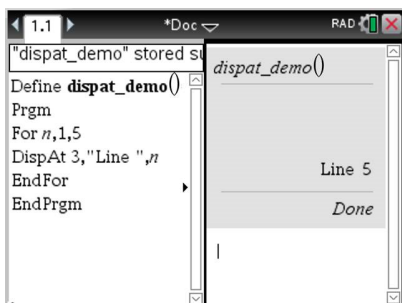
Exempel

```
1.1 | *Doc | RAD |
-----
dispat_demo 2/3
Define dispat_demo()
Prgm
For n,1,5
DispAt n,"Linje ",n
EndFor
EndPrgm
-----
dispat_demo()
Line 1
Line 2
Line 3
Line 4
Line 5
-----
Done
```

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

DispAtoch Disp kan användas i samma program.

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



Illustrativa exempel:

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

EndPrgm

Feltillstånd:

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

►DMS

List ►DMS

I vinkelläget Grader:

Matrix ►DMS

(45.371) ►DMS	$45^{\circ}22'15.6''$
$\{(45.371,60)\}$ ►DMS	$\{45^{\circ}22'15.6'',60^{\circ}\}$

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 186 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

Ger "prick"-produkten av två listor.



$\text{dotP}(\text{Vector1}, \text{Vector2}) \Rightarrow$ uttryck

Ger "prick"-produkten av två vektorer.

Båda måste vara radvektorer eller båda måste vara kolumnvektorer.

E

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

Obs: Att trycka på  för att visa e^x 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.

$e^{\text{List1}} \Rightarrow$ lista

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

$e^{\text{squareMatrix1}} \Rightarrow$ kvadratMatrix

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

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

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

$\text{eff}(\text{nominalRate}, \text{CpY}) \Rightarrow$ värde

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

5.90398

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 100.

eigVc()

eigVc(squareMatrix)⇒matrix

I Rektangulärt komplext format:

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

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

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

För att se hela resultatet, tryck på \blacktriangle och använd sedan \blacktriangleleft och \blacktriangleright för att flytta markören.

eigVl()

eigVl(squareMatrix)⇒lista

I läget Rektangulärt komplext format:

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

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

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

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

För att se hela resultatet, tryck på \blacktriangle och använd sedan \blacktriangleleft och \blacktriangleright för att flytta markören.

Else

Se If, på sidan 67.

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 $g(x)$ =FuncIf $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ ThenReturn $-x$ ElseIf $x \geq 0$ and $x \neq 10$ ThenReturn x ElseIf $x = 10$ Then

Return 3

EndIf

EndFunc

*Done***EndFor**

Se For, på sidan 52.

EndFunc

Se Func, på sidan 56.

EndIf

Se If, på sidan 67.

EndLoop

Se Loop, på sidan 88.

EndPrgm

Se Prgm, på sidan 112.

euler ()

Katalog > 

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

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

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

Använder Eulers metod för att lösa systemet

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

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.

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*).

Differentialekvation:

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

$$\text{euler}\left(0,001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right)$$

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:

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

med *yI*(0)=2 och *y2*(0)=5

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

0.	1.	2.	3.	4.	5.
2.	1.	1.	3.	27.	243.
5.	10.	30.	90.	90.	-2070.

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

eval ()

Hubb-meny

eval(*Expr*) ⇒ *string*

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.

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

<i>lum</i> :=127	127
Send "SET COLOR.BLUE eval(<i>lum</i>)"	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 58), **GetStr** (på sidan 64), och **Send** (på sidan 132).

Exit

Exit

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.

Funktionslista:

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

exp()

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

exp()

e^x -tangent

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) ⇒ kvadratMatrix

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.

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

expr()

Katalog > 

expr(String) ⇒ uttryck

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

ExpReg

Katalog > 

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 143.)

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 kategorikoder 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 206).

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

factor(*rationalNumber*) ger det rationella talet faktorerat 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.

F Cdf()

F Cdf

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

FCdf

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

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

För $P(X \leq \text{övrGräns})$, ange *undrGräns* = 0.

Fill

matrixVar måste redan existera.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	→ <i>amatrix</i>	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
Fill 1.01, <i>amatrix</i>		<i>Done</i>
<i>amatrix</i>		$\begin{bmatrix} 1.01 & 1.01 \\ 1.01 & 1.01 \end{bmatrix}$

listVar måste redan existera.

$\{1,2,3,4,5\} \rightarrow$	<i>alist</i>	$\{1,2,3,4,5\}$
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 143.)

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 kategorikoder 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 206.

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()

Katalog > 

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 l) \Rightarrow lista

floor(Matrix l) \Rightarrow matris

Ger en lista eller matris med golvvärden för varje element.

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

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

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

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

For

Katalog > 

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

Block

EndFor

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

Var får inte vara en systemvariabel.

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

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

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

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

$g()$	5050
-------	------

format()

Katalog > 

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.

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 enstaka tecken som specificerar vad som skall ersättas för baspunkten.

<code>format(1.234567,"f3")</code>	<code>"1.235"</code>
<code>format(1.234567,"s2")</code>	<code>"1.23E0"</code>
<code>format(1.234567,"e3")</code>	<code>"1.235E0"</code>
<code>format(1.234567,"g3")</code>	<code>"1.235"</code>
<code>format(1234.567,"g3")</code>	<code>"1,234.567"</code>
<code>format(1.234567,"g3,r:")</code>	<code>"1:235"</code>

fPart()

Katalog > 

`fPart(Expr1)⇒uttryck`

`fPart(List1)⇒lista`

`fPart(Matrix1)⇒matrix`

Ger argumentets bråkdel.

Ger, för en lista eller matrix, elementens bråkdelar.

Argumentet kan vara ett reellt eller ett komplext tal.

<code>fPart(-1.234)</code>	<code>-0.234</code>
<code>fPart({1,-2.3,7.003})</code>	<code>{0,-0.3,0.003}</code>

$\text{FPdf}(XVal, dfNumer, dfDenom) \Rightarrow tal$ 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()

freqTable►list

$(List1, freqIntegerList) \Rightarrow lista$

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

$List1$ 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 206.

```
freqTable►list({1,2,3,4},{1,4,3,1})
                {1,2,2,2,3,3,3,4}
```

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

frequency()

$\text{frequency}(List1, binsList) \Rightarrow lista$

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

```
datalist={1,2,e,3,pi,4,5,6,"hello",7}
          {1,2,2.71828,3,3.14159,4,5,6,"hello",7}
frequency(datalist,{2.5,4.5})           {2,4,3}
```

Förklaring av resultat:

Om *binsList* är { $b(1), b(2), \dots, b(n)$ } är de specificerade områdena $\{? \leq b(1), b(1) < ? \leq b(2), \dots, b(n-1) < ? \leq b(n), b(n) > ?\}$. 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 206.

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

Obs: Se även **countif()**, på sidan 29.

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

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 143.)

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 206).

Resultatvariabel	Beskrivning
stat.F	Beräknad \hat{U} -statistik för datasekvensen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.dfNumer	täljare, frihetsgrader = $n1-1$
stat.dfDenom	nämnare, frihetsgrader = $n2-1$
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i <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

Func

Func

Definiera en stegvis funktion:

Block

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

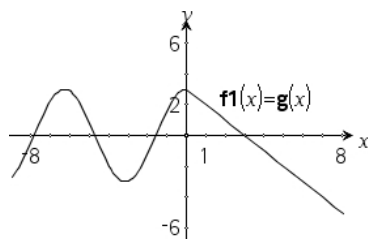
EndFunc

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.

Resultat från plottning av $g(x)$



gcd()Katalog > **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 bråkital 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 *matrix* $\text{gcd}\left(\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}\right)$ $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$

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

geomCdf()Katalog > 

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

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

geomPdf()Katalog > 

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

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.

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
```

Hubb-meny

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

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 64 och **Send**, på sidan 132.

getDenom()

Katalog > 

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

getKey()

Katalog > 

getKey([0|1]) ⇒ returnString

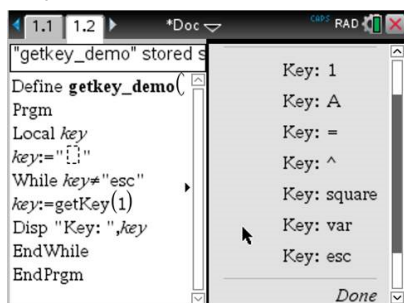
Beskrivning: **getKey()** – låter ett TI-Basic-program tolka tangentbordsinmatning – på handenhet, dator och emulator på dator.

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.

`getKey()`

Exempel:



Hantering av tangentnedtryckningar:

Handhållen enhet/emulatortangent	Desktop (dator)	Returvärde
Esc	Esc	"esc"
Pekplatta - toppklick	Ej tillämpligt	"up"
På	Ej tillämpligt	"home"

Handhållen enhet/emulatortangent	Desktop (dator)	Returvärde
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"
=	=	"_="
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"
+	+	"_+"
-	-	"_-"
(("("

Handhållen enhet/emulator tangent	Desktop (dator)	Returvärde
)))"
.	.	". "
(-)	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"
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 handenheter 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()

getLockInfo(Var)⇒värde

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

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

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

Se **Lock**, på sidan 85 och **unLock**, på sidan 161.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

getMode()

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 134.

Lägets namn	Läges-heltal	Heltal för inställningar
Display Digits (Antal siffror)	1	1 =Float, 2 =Float1, 3 =Float2, 4 =Float3, 5 =Float4, 6 =Float5, 7 =Float6, 8 =Float7, 9 =Float8, 10 =Float9, 11 =Float10, 12 =Float11, 13 =Float12, 14 =Fix0, 15 =Fix1, 16 =Fix2, 17 =Fix3, 18 =Fix4, 19 =Fix5, 20 =Fix6, 21 =Fix7, 22 =Fix8, 23 =Fix9, 24 =Fix10, 25 =Fix11, 26 =Fix12
Angle (Vinkel)	2	1 =Radian, 2 =Degree, 3 =Gradian
Exponential Format (Exponentiellt format)	3	1 =Normal, 2 =Scientific, 3 =Engineering
Real or Complex (Reellt eller Komplext)	4	1 =Real, 2 =Rectangular, 3 =Polar
Auto or Approx. (Auto eller Ungefärlig)	5	1 =Auto, 2 =Approximate
Vector Format (Vektorformat)	6	1 =Rectangular, 2 =Cylindrical, 3 =Spherical
Base (Bas)	7	1 =Decimal, 2 =Hex, 3 =Binary

getNum()

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

GetStr

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 58 och **Send**, på sidan 132.

getType()

Katalog > 

getType(var) ⇒ *sträng*

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

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

{ 1,2,3 } → <i>temp</i>	{ 1,2,3 }
getType(<i>temp</i>)	"LIST"
3 · <i>i</i> → <i>temp</i>	3 · <i>i</i>
getType(<i>temp</i>)	"EXPR"
DelVar <i>temp</i>	<i>Done</i>
getType(<i>temp</i>)	"NONE"

getVarInfo()

Katalog > 

getVarInfo() ⇒ *matris* eller *sträng*

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

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

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

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

Om biblioteket *LibNameString* inte finns uppstår ett fel.

getVarInfo()	"NONE"												
Define $x=5$	<i>Done</i>												
Lock x	<i>Done</i>												
Define LibPriv $y=\{1,2,3\}$	<i>Done</i>												
Define LibPub $z(x)=3 \cdot x^2 - x$	<i>Done</i>												
getVarInfo()	<table border="1"> <tbody> <tr> <td>x</td> <td>"NUM"</td> <td>"{ }"</td> <td>1</td> </tr> <tr> <td>y</td> <td>"LIST"</td> <td>"LibPriv "</td> <td>0</td> </tr> <tr> <td>z</td> <td>"FUNC"</td> <td>"LibPub "</td> <td>0</td> </tr> </tbody> </table>	x	"NUM"	"{ }"	1	y	"LIST"	"LibPriv "	0	z	"FUNC"	"LibPub "	0
x	"NUM"	"{ }"	1										
y	"LIST"	"LibPriv "	0										
z	"FUNC"	"LibPub "	0										
getVarInfo(<i>tmp3</i>)	"Error: Argument must be a string"												
getVarInfo("tmp3")	[<i>volcy12</i> "NONE" "LibPub " 0]												

getVarInfo()

Katalog > 

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.

$a:=1$	1												
$b:=[1\ 2]$	$[1\ 2]$												
$c:=[1\ 3\ 7]$	$[1\ 3\ 7]$												
$vs:=\text{getVarInfo}()$	<table border="1"><tr><td><i>a</i></td><td>"NUM"</td><td>"[]"</td><td>0</td></tr><tr><td><i>b</i></td><td>"MAT"</td><td>"[]"</td><td>0</td></tr><tr><td><i>c</i></td><td>"MAT"</td><td>"[]"</td><td>0</td></tr></table>	<i>a</i>	"NUM"	"[]"	0	<i>b</i>	"MAT"	"[]"	0	<i>c</i>	"MAT"	"[]"	0
<i>a</i>	"NUM"	"[]"	0										
<i>b</i>	"MAT"	"[]"	0										
<i>c</i>	"MAT"	"[]"	0										
$vs[1]$	$[1\ \text{"NUM"}\ \text{"[]"}\ 0]$												
$vs[1,1]$	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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $g()$ =Func	<i>Done</i>
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)	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1

IfKatalog > 

If BooleanExpr
Påståenden

If BoolesktUttr Then
Block

EndIf

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 BoolesktUttr Then
Block1

Else
Block2

EndIf

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

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

Block1 och *Block2* kan vara enstaka påståenden.

```
Define g(x)=Func Done
  If x<0 Then
    Return x2
  EndIf
EndFunc
```

g(-2)	4
-------	---

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

g(12)	12
g(-12)	12

```

If BoolesktUttr1 Then
    Block1
ElseIf BoolesktUttr2 Then
    Block2
:
ElseIf BoolesktUttrN Then
    BlockN
EndIf

```

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

```

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

```

Done

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

ifFn ()

```

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*.

```

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.

```

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.

```

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

```


ifFn()

Katalog > 

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.

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()

Katalog > 

Ger argumentets imaginärdel.

imag(ListI) ⇒ *lista*

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

$$\{0, -1, 1\}$$

Ger en lista på elementens imaginärdelar.

imag(MatrixI) ⇒ *matris*

Ger en matris med elementens imaginärdelar.

Indirection

Se #(), på sidan 184.

inString()

Katalog > 

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

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

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*.

int()

Katalog > 

$\text{int}(\text{List1}) \Rightarrow \text{lista}$

$\text{int}(\text{Matrix1}) \Rightarrow \text{matrix}$

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

Argumentet kan vara ett reellt eller ett komplext tal.

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

$\text{int}(-2.5)$	-3.
$\text{int}([-1.234 \ 0 \ 0.37])$	$[-2. \ 0 \ 0.]$

intDiv()

Katalog > 

$\text{intDiv}(\text{Number1}, \text{Number2}) \Rightarrow \text{heltal}$

$\text{intDiv}(\text{List1}, \text{List2}) \Rightarrow \text{lista}$

$\text{intDiv}(\text{Matrix1}, \text{Matrix2}) \Rightarrow \text{matrix}$

Ger heltalsdelen med tecken av ($\text{Number1} \div \text{Number2}$).

Ger, för listor och matriser, heltalsdelen med tecken av ($\text{argument1} \div \text{argument2}$) för varje elementpar.

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

Interpolera ()

Katalog > 

$\text{Interpolera}(x\text{Value}, x\text{List}, y\text{List}, y\text{PrimeList}) \Rightarrow \text{lista}$

Denna funktion gör följande:

Förutsatt $x\text{List}$, $y\text{List}=\mathbf{f}(x\text{List})$ och $y\text{PrimeList}=\mathbf{f}'(x\text{List})$ används för en okänd funktion \mathbf{f} en kubisk interpolant för att uppskatta funktionen \mathbf{f} vid $x\text{Value}$. Det förutsätts att $x\text{List}$ 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 $x\text{List}$ och söker ett intervall $[x\text{List}[i], x\text{List}[i+1]]$ som innehåller $x\text{Value}$. Om funktionen hittar ett sådant intervall ger den ett interpolerat värde på $\mathbf{f}(x\text{Value})$, annars ger den **undef**.

Differentialekvation:

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

$r\text{rk}=\text{rk}23(-3 \cdot y+6 \cdot t+5, t, y, \{0, 10\}, 5, 1)$				
0.	1.	2.	3.	4.
5.	3.19499	5.00394	6.99957	9.00593

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

Interpolera ()

Katalog > 

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

Använd funktionen `interpolate()` för att beräkna funktionsvärdena för x valueList:

```
xvalueList:=seq(i,i,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,7.,7.5,8.,8.5,9.,9.5,10.}
xlist:=mat▶list(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=mat▶list(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.9979}
yprimeList:=-3*y+6*t+5|y=ylist and t=xlist
{-10.,1.41503,1.98819,2.00129,1.98221,2.0061}
interpolate(xvalueList,xlist,ylist,yprimeList)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011,7.00011,8.00011,9.00011,10.00011}
```

inv χ^2 ()

Katalog > 

`inv χ^2 (Area,df)`

`invChi2(Area,df)`

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

invF()

Katalog > 

`invF(Area,dfNumer,dfDenom)`

`invF(Area,dfNumer,dfDenom)`

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

invBinom

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

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

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

OutputForm=1, visar resultatet som en matris.

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

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

invBinomN()

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

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

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

OutputForm=1, visar resultatet som en matris.

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

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

invNorm()

invNorm(*Area*, μ , σ)

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

inv(*Area,df*)

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

iPart()

iPart(*Number*) ⇒ *heltal*

iPart(*List1*) ⇒ *lista*

iPart(*Matrix1*) ⇒ *matris*

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

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 93.

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

isPrime()

Katalog > 

isPrime(Number) ⇒ *Booleskt konstantuttryck*

isPrime(5)	true
isPrime(6)	false

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.

Funktion för att hitta nästa primtal efter ett specificerat tal:

Define <i>nextprim(n)</i> = Func	<i>Done</i>
Loop	
<i>n</i> + 1 → <i>n</i>	
If isPrime(<i>n</i>)	
Return <i>n</i>	
EndLoop	
EndFunc	
<i>nextprim(7)</i>	11

isVoid()

Katalog > 

isVoid(Var) ⇒ *Booleskt konstantuttryck*

isVoid(Expr) ⇒ *Booleskt konstantuttryck*

isVoid(List) ⇒ *lista av Booleska konstantuttryck*

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

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

För mer information om tomma element, se på sidan 206.

Lbl

Katalog > **Lbl** *labelName*

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

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

labelName måste uppfylla samma krav på namngivning som ett variabelnamn.

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

Define $g()$ =Func	<i>Done</i>
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	
<hr/> $g()$	<hr/> 55

lcm()

Katalog > **lcm**(*Number1, Number2*) \Rightarrow uttryck

lcm(6,9)	18
----------	----

lcm(*List1, List2*) \Rightarrow lista

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

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()

Katalog > **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*]) \Rightarrow lista

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

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

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

left(*Comparison*) \Rightarrow uttryck

```
left(x<3) x
```

Ger den vänstra sidan av en ekvation eller olikhet.

libShortcut()

libShortcut(*LibNameString*,
ShortcutNameString

[, *LibPrivFlag*]) \Rightarrow 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 37.

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 143.)

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 kategorikoder 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 206).

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 143.)

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 kategorikoder 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 206).

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 <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>

LinRegtIntervals

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

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

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

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

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 206).

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
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.ŷ	$a + b \cdot X_{\text{val}}$

LinRegtTest

Katalog > 

LinRegtTest $X, Y, \text{Freq}, \text{Hypoth}$

Utför en linjär regressionsanalys på listorna X och Y och ett t -test på lutningens värde β samt korrelationskoefficienten ρ för ekvationen $y = \alpha + \beta x$. Det testar nollhypotesen $H_0: \beta = 0$ (equivalently, $\rho = 0$) mot en av tre alternativa hypoteser.

Alla listor måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

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

Hypoth är ett valfritt värde som specificerar en av tre alternativa hypoteser mot vilka nollhypotesen ($H_0: \beta = \rho = 0$) kommer att testas.

För $H_a: \beta \neq 0$ och $\rho \neq 0$ (förinställning), ställ $\text{Hypoth} = 0$

För $H_a: \beta < 0$ och $\rho < 0$, ställ $\text{Hypoth} < 0$

För $H_a: \beta > 0$ och $\rho > 0$, ställ $\text{Hypoth} > 0$

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionssekvation: $a + b \cdot x$
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()

linSolve(SystemAvLinjäraEkv, Var1, Var2, ...) ⇒ lista

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

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

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

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

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

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

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

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()

$\Delta\text{List}(\text{List}I) \Rightarrow \text{lista}$

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

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

Ger en lista på skillnaderna mellan konsekutiva element i *ListI*. Varje element i *ListI* subtraheras från nästa element i *ListI*. Den resulterande listan är alltid ett element kortare än den ursprungliga *ListI*.

list@mat()

$\text{list@mat}(\text{List } [, \text{elementsPerRow}]) \Rightarrow \text{matrix}$

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

Ger en matris fylld rad efter rad med elementen från *List*.

elementsPerRow, om inkluderad, specificerar antalet element per rad. Förinställningen är antalet element i *List* (en rad).

Om *List* inte fyller den resulterande listan läggs nollor till.

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

ln()

$\ln(\text{List}I) \Rightarrow \text{lista}$

$\ln(2)$	0.693147
----------	----------

Ger argumentets naturliga logaritm.

Ger, för en lista, elementens naturliga logaritmer.

$\ln(\text{squareMatrix1}) \Rightarrow \text{kvadratMatris}$

Ger matrisen med naturlig logaritm för *squareMatrix1*. Detta är inte detsamma som att beräkna den naturliga logaritmen för varje element. För information om beräkningsmetoden, se **cos()**.

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

Om det komplexa formatläget är Real:

$$\ln(\{-3, 1.2, 5\})$$

"Error: Non-real calculation"

Om det komplexa formatläget är Rectangular:

I vinkelläget Radianer och i Rektangulärt komplext format:

$$\ln \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$$

1.83145+1.73485 <i>i</i>	0.009193-1.49086
0.448761-0.725533 <i>i</i>	1.06491+0.623491 <i>i</i>
-0.266891-2.08316 <i>i</i>	1.12436+1.79018 <i>i</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 143.)

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 kategorikoder 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 206).

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

Local

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

Betecknar specificerade *vars* som lokala variabler. Dessa variabler existerar endast under utvärderingen av ett uttryck och tas bort när exekveringen av uttrycket är klar.

Obs: Lokala variabler sparar minne eftersom de endast existerar tillfälligt. De stör heller inga befintliga globala variabelvärden. Lokala variabler måste användas för **For**-slingor och för att temporärt spara värden i en flerradig funktion eftersom modifieringar av globala värden inte är tillåtna i en funktion.

```

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

```

	Done
<i>rollcount</i> ()	16
<i>rollcount</i> ()	3

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

Lock

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 161 och **getLockInfo()**, på sidan 63.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

log()

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

Om det andra argumentet utelämnas används 10 som bas.

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

Om basargumentet utelämnas används 10 som bas.




Om det komplexa formatläget är Real:

Om det komplexa formatläget är Rectangular:

I vinkelläget Radianer och i Rektangulärt komplext format:

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

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

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

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

Utför den logistiska regressionsanalysen $= (c/(1+a \cdot e^{-bx}))$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 kategorikoder 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 206).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade X List som används i regressionen baserat på begränsningar i $Freq$, $Category$ List och $Include$ Categories
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i $Freq$, $Category$ List och $Include$ Categories
stat.FreqReg	Lista på frekvenser som motsvarar $stat.XReg$ och $stat.YReg$

LogisticD 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 143.)

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 kategorikoder 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 206).

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>
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>

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

Loop

Katalog >

Loop

Block

EndLoop

Exekverar påståendena i *Block* upprepade gånger. Observera att slingan upprepas i all oändlighet såvida inte en **Goto**- eller **Exit**-instruktion exekveras inom *Block*.

Block är en serie av påståenden separerade med tecknet “.”.

Obs för att mata in exemplet: Se avsnittet 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

LU

Katalog >

LU *Matris*, *lMatris*, *uMatris*, *pMatris* [,*Tol*]

Beräknar uppdelningen Doolittle LU (undre-övre) av en reell eller komplex matris. Den undertriangulära matrisen lagras i *lMatris*, den övertriangulära matrisen lagras i *uMatris* och permutationsmatrisen (som beskriver radväxlingarna som har gjorts under beräkningen) lagras i *pMatris*.

$lMatris \cdot uMatris = pMatris \cdot matris$

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

$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$
LU <i>m1</i> , <i>lower</i> , <i>upper</i> , <i>perm</i>	Done
<i>lower</i>	$\begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$
<i>upper</i>	$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$
<i>perm</i>	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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

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(...)`.

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

max()

Katalog > 

max(List1, List2) ⇒ lista

max(Matrix1, Matrix2) ⇒ matris

Ger de två argumentens maximum. Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller maximumvärdet för varje par av motsvarande element.

max(List) ⇒ uttryck

Ger maximelementet i *list*.

max(Matrix1) ⇒ matris

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

<code>max(2.3,1.4)</code>	<code>2.3</code>
<code>max({1,2},{-4,3})</code>	<code>{1,3}</code>
<code>max({0,1,-7,1.3,0.5})</code>	<code>1.3</code>
<code>max([1 -3 7; -4 0 0.3])</code>	<code>[1 0 7]</code>

Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

Obs: Se även min().

mean()

mean(List[, freqList]) ⇒ *uttryck*

Ger medelvärdet för elementen i *List*.

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

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

Ger en radvektor med medelvärdena för alla kolumner i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

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

I vektorformatet Rectangular:

$\text{mean}\left(\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}\right)$	$[-0.133333 \quad 0.833333]$
$\text{mean}\left(\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & \frac{-1}{2} \end{bmatrix}\right)$	$\left[\frac{-2}{15} \quad \frac{5}{6}\right]$
$\text{mean}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \begin{bmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{bmatrix}\right)$	$\left[\frac{47}{15} \quad \frac{11}{3}\right]$

median()

median(List[, freqList]) ⇒ *uttryck*

Ger medianen för elementen i *List*.

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

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

Ger en radvektor som innehåller medianerna för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

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

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

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 206.

MedMed

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

Beräknar median-median-linjens $(m \cdot x + b)$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 kategorikoder 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 206).

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 teckenumret *Start*.

Ger, om *Count* utelämnas eller är större än dimensionen på *sourceString*, alla tecken från *sourceString* med start från teckenumret *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(*List1*, *List2*) ⇒ *lista*

min(*Matrix1*, *Matrix2*) ⇒ *matrix*

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

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.

$\text{min}(\text{List}) \Rightarrow \text{uttryck}$

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

Ger minimelementet för *List*.

$\text{min}(\text{Matrix } I) \Rightarrow \text{matris}$

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

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

Obs: Se även **max()**.

mirr

$(\text{financeRate}, \text{reinvestRate}, \text{CF0}, \text{CFList}, \text{CFFreq})$

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 73.

$\text{list1} := \{6000, -8000, 2000, -3000\}$	$\{6000, -8000, 2000, -3000\}$
$\text{list2} := \{2, 2, 2, 1\}$	$\{2, 2, 2, 1\}$
$\text{mirr}(4.65, 12, 5000, \text{list1}, \text{list2})$	13.41608607

mod()Katalog > **mod(List1, List2) ⇒ lista**

mod(7,0) 7

mod(Matrix1, Matrix2) ⇒ matris

mod(7,3) 1

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

mod(-7,3) 2

mod(7,-3) -2

mod(-7,-3) -1

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

mod(x,0) = x

mod(x,y) = x - y floor(x/y)

När det andra argumentet är skilt från noll är resultatet periodiskt i det argumentet. Resultatet är antingen noll eller har samma tecken som det andra argumentet.

Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller modulen för varje par av motsvarande element.

Obs: Se även **remain()**, på sidan 123

mRow()Katalog > 

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

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

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

Beräknar den multipla linjära regressionen i lista *Y* på listorna *X1, X2, ..., X10*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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.yList	$\hat{y}List = b_0+b_1 \cdot x_1+ \dots$
stat.Resid	Residualer från regressionsanalysen

MultRegIntervals

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

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

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.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
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelvärdet på svaret
stat.LowerPred, stat.UpperrPred	Prediktionsintervall för en enstaka observation

Resultatvariabel	Beskrivning
stat.MEPred	Prediktionsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.bList	Lista på regressionskoefficienter, {b0,b1,b3,...}
stat.Resid	Residualer från regressionen

MultRegTests

Katalog > 

MultRegTests $Y, X1[,X2[,X3,...[,X10]]]$

Ett multipelt linjärt regressionstest utför en multipel linjär regressionsanalys på givna data och ger den globala F -teststatistiken och t -teststatistiken för koefficienterna.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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
stat.SSReg	Regressionens kvadratsumma
stat.MSReg	Regression medelkvadrat
stat.dfError	Fel hos frihetsgrader
stat.SSError	Felens kvadratsumma

Resultatvariabel	Beskrivning
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

$x \geq 3$ and $x \geq 4$

$x \geq 4$

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

$x \geq 3$ nand $x \geq 4$

$x < 4$

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.

nandctrl  knappar*Heltal1 nand Heltal2* ⇒ *heltal*

Jämför två reella heltal bit för bit med en **nand**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs är resultatet 0 om båda bitarna är 1; annars är resultatet 1. Det returnerade värdet representerar bitresultaten och visas enligt basläget.

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

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

nCr()Katalog > *nCr(List1, List2)* ⇒ *lista*

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

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

nCr(Matrix1, Matrix2) ⇒ *matrix*

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

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

nDerivative()Katalog > *nDerivative(Uttr1, Var = Värde [, Ordning])* ⇒ *värde**nDerivative(Uttr1, Var [, Ordning]) | Var = Värde* ⇒ *värde*

Ger den numeriska derivatan beräknad med automatiska deriveringsmetoder.

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

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

nDerivative()Katalog > 

Ordning för derivatan måste vara 1 eller 2.

newList()Katalog > 

newList(numElements) ⇒ lista

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

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

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

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

nfMax(Expr, Var, undrGräns) ⇒ värde

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

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

nfMin(Expr, Var, undrGräns) ⇒ värde

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

nfMin(Expr, Var, undrGräns, övrGräns) ⇒ värde

nfMin()Katalog >  $\text{övrGräns}) \Rightarrow \text{värde}$ **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) ⇒ uttryck
$$\text{nInt}(e^{-x^2}, x, -1, 1) \quad 1.49365$$

Om integranden *Expr1* inte innehåller någon variabel utöver *Var*, och om *Lower* och *Upper* är konstanter, positiv ∞ eller negativ ∞ , ger **nInt()** en uppskattning av $\int(\text{Expr1}, \text{Var}, \text{Lower}, \text{Upper})$. 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.

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*) ⇒ 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 42.

nor

  knappar

BoolesktUttr1 **nor** *BoolesktUttr2* ger
Booleskt uttryck

$x \geq 3$ or $x \geq 4$	$x \geq 3$
--------------------------	------------

$x \geq 3$ nor $x \geq 4$	$x < 3$
---------------------------	---------

BooleskLista1 **nor** *BooleskLista2* ger
Boolesk lista

Keep CAS image?

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.

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}
---------------------	------------

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).

norm()

Katalog > 

norm(*Matrix*) \Rightarrow uttryck

norm(*Vector*) \Rightarrow uttryck

Ger Frobenius norm.

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).

normPdf()

Katalog > 

normPdf(*XVal*[, μ],[σ]) \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 (icke)

Katalog > 

not *BooleanExpr* \Rightarrow Booleskt uttryck

Ger en sann, falsk eller förenklad form av argumentet.

not *IntegerI* \Rightarrow heltal

Ger ettkomplementet till ett reellt heltal. Internt omvandlas *IntegerI* 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).

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

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

I binärt basläge:

not (icke)

Katalog > 

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.

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 > 

$nPr(List1, List2) \Rightarrow lista$

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

$nPr(\{5,4,3\}, \{2,4,2\})$	$\{20,24,6\}$
-----------------------------	---------------

$nPr(Matrix1, Matrix2) \Rightarrow matrix$

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

$nPr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$
--	---

npv()

Katalog > 

$npv(InterestRate, CFO, CFList [, CFFreq])$

Finansiell funktion som beräknar nettovärdet, dvs. summan av aktuella värden för kassainflöden och kassautflöden. Ett positivt resultat för npv indikerar en vinstgivande investering.

$list1 := \{6000, -8000, 2000, -3000\}$	$\{6000, -8000, 2000, -3000\}$
$list2 := \{2, 2, 2, 1\}$	$\{2, 2, 2, 1\}$
$npv(10, 5000, list1, list2)$	4769.91

InterestRate är räntan med vilken kassaflödena (kapitalanskaffningskostnaderna) diskonteras under en period.

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

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

CFFreq är en lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

nSolve()

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

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

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

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".

$\text{nSolve}(x^2+5\cdot x-25=9,x)$	3.84429
$\text{nSolve}(x^2=4,x=-1)$	-2.
$\text{nSolve}(x^2=4,x=1)$	2.

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

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

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 143.)

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 kategorikoder 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 206.

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

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
stat.SSX	Kvadratsumma av avvikelser från medelvärdet på x

or (eller)

Katalog > 

BoolesktUttr1 **or** *BoolesktUttr2* 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 Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Integer1 **or** *Integer2* ⇒ *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
```

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).

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

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)` ⇒ *integer*

<code>ord("hello")</code>	104
---------------------------	-----

`ord(List l)` ⇒ *lista*

<code>char{104}</code>	"h"
------------------------	-----

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

<code>ord(char{24})</code>	24
----------------------------	----

<code>ord>{"alpha", "beta"}</code>	{97,98}
---------------------------------------	---------

P

►►Rx()

`►►Rx(rExpr, θExpr)` ⇒ *uttryck*

I vinkelläget Radianer:

`►►Rx(rList, θList)` ⇒ *lista*

`►►Rx(rMatrix, θMatrix)` ⇒ *matrix*

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 $^\circ$, $^\text{G}$ eller $^\text{r}$ för att tillfälligt överstyra vinkelläget.

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

P@Ry(*rList*, *θList*) \Rightarrow *lista*

I vinkelläget Radianer:

P@Ry(*rMatrix*, *θMatrix*) \Rightarrow *matrix*

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 154.

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 154.

Kommentarer om inmatningen av exemplet: applikationen Räkna kan du skriva in flerradiga definitioner genom att trycka på  i stället för **enter** i slutet av varje linje. På datorns tangentbord, håll ned **Alt** och tryck på **Enter**.

piecewise()

Katalog > 

piecewise(*Expr1* [, *Condition1* [, *Expr2* [, *Condition2* [, ...]]])

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$ Done

Ger definitioner för en stegvis funktion i form av en lista. Du kan också skapa stegvisa definitioner med hjälp av en mall.

$p(1)$	1
$p(-1)$	undef

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

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

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 120.

complexValue ►Polar

I vinkelläget Radianer:

Visar *complexVector* i polär form.

- Vinkelläget Grader ger $(r \angle \theta)$.
- Vinkelläget Radianer ger $re^{i\theta}$.

I vinkelläget Nygrader:

$(4 \cdot i)$ ►Polar

$(4 \angle 100.)$

complexValue kan ha valfri komplex form. En inmatning av $re^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

I vinkelläget Grader:

Obs: Du måste använda parenteserna för en $(r \angle \theta)$ polär inmatning.

polyEval()

polyEval(List1, Expr1) ⇒ uttryck

polyEval(List1, List2) ⇒ uttryck

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

polyRoots()

polyRoots(Poly, Var) ⇒ lista

polyRoots(ListaPåKoeff) ⇒ lista

Den första syntaxen, **polyRoots(Poly, Var)**, ger en lista på reella rötter i polynomet *Poly* med avseende på variabeln *Var*. Om inga reella rötter existerar ger den en tom lista: { }.

Den andra syntaxen, **polyRoots(ListaPåKoeff)** ger en lista på reella rötter för koefficienterna i *ListapåKoeff*.

Obs: Se även **cPolyRoots()**, på sidan 29.

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

Utför potensregressionsanalys $y = (a \cdot (x)^b)$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 kategorikoder 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 206).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot (x)^b$
stat.a, stat.b	Regressionskoefficienter
stat.r ²	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data ($\ln(x)$, $\ln(y)$)
stat.Resid	Residualer associerade med potensmodellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade X List som används i regressionen baserat på begränsningar i $Freq$, $Category$ List och $Include$ Categories
stat.YReg	Lista på datapunkter i den modifierade Y List som används i regressionen baserat på begränsningar i $Freq$, $Category$ List och $Include$ Categories
stat.FreqReg	Lista på frekvenser som motsvarar $stat.XReg$ och $stat.YReg$

Prgm

Beräkna GCD och visa mellanliggande resultat.

Block

EndPrgm

Mall för att skapa ett användardefinierat program. Måste användas med kommandot **Define**, **Define LibPub** eller **Define LibPriv**.

Block kan vara ett enstaka påstående, en serie av påståenden separerade med tecknet ":" eller en serie av påståenden på separata rader.

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

```
Define proggcd(a,b)=Prgm
  Local d
  While b≠0
  d:=mod(a,b)
  a:=b
  b:=d
  Disp a," ",b
  EndWhile
  Disp "GCD=",a
EndPrgm
Done
```

```
proggcd(4560,450)
-----
450 60
60 30
30 0
GCD=30
-----
Done
```

prodSeq()

Se $\Pi()$, på sidan 181.

Product (PI)

Se $\Pi()$, på sidan 181.

product()

product(List[, Start[, end]]) ⇒ uttryck

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

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

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

```
product( [ 1 2 3 ] [28 80 162]
         [ 4 5 6 ]
         [ 7 8 9 ] )
product( [ 1 2 3 ] [4 10 18]
         [ 4 5 6 ] ,1,2
         [ 7 8 9 ] )
```

product()Katalog > 

Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

propFrac()Katalog > 

propFrac(*rational_number*) ger *rational_number* som summan av ett heltal och ett bråk med samma tecken och med större nämnare än täljare.

propFrac(*rational_expression,Var*) ger summan av egentliga bråk och ett polynom med avseende på *Var*. Graden hos *Var* i nämnaren överskrider graden hos *Var* i täljaren i varje egentligt bråk. Liknande potenser av *Var* samlas in. Termerna och deras faktorer sorteras med *Var* som huvudvariabel.

Om *Var* utelämnas utförs en utveckling av ett egentligt bråk med avseende på den mest betydande variabeln.

Koefficienterna för polynomdelen görs sedan egentliga, först med avseende på deras mest betydande variabel och så vidare.

Du kan använda funktionen **propFrac()** för att representera blandade bråk och demonstrera addition och subtraktion av blandade bråk.

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

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

$$\text{propFrac}\left(\frac{x^2+x+1}{x+1}+\frac{y^2+y+1}{y+1},x\right) \quad \frac{1}{x+1}+x+\frac{y^2+y+1}{y+1}$$

$$\text{propFrac}(\text{Ans}) \quad \frac{1}{x+1}+x+\frac{1}{y+1}+y$$

$$\text{propFrac}\left(\frac{11}{7}\right) \quad 1+\frac{4}{7}$$

$$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right) \quad 8+\frac{37}{44}$$

$$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right) \quad -2-\frac{29}{44}$$

Q**QR**Katalog > 

QR *Matrix, qMatName, rMatName[, Tol]*

Siffran för flytande komma (9.) i m1 medför att resultat beräknas med flyttalsaritmetik.

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 basvektorerna som täcker utrymmet som definieras av *matrix*.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow mI$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
QR <i>mI,qm,rm</i>	Done
<i>qm</i>	$\begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$
<i>rm</i>	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow mI$	$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$
QR <i>mI,qm,rm</i>	Done
<i>qm</i>	$\begin{bmatrix} \frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+o^2}} \\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$
<i>rm</i>	$\begin{bmatrix} \sqrt{m^2+o^2} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+o^2}} \\ 0 & \frac{ m \cdot p - n \cdot o }{\sqrt{m^2+o^2}} \end{bmatrix}$

QuadReg

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 143.)

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 kategorikoder 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 206.

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 143.)

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 kategorikoder 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 206).

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

R ► Pθ()

R ► Pθ ($xList$, $yList$) ⇒ lista

Med vinkelmått Grader:

R ► Pθ ($xMatrix$, $yMatrix$) ⇒ matrix

Med vinkelenhet Nygrader:

R ▶ Pθ()Katalog > 

Ger den ekvivalenta θ -koordinaten för (x,y) värden.

Med vinkelenhet Radianer:

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 (...)**.

R ▶ Pr()Katalog > 

Med vinkelenhet Radianer:

R ▶ Pr ($xList, yList$) \Rightarrow lista

R ▶ Pr ($xMatrix, yMatrix$) \Rightarrow matrix

Ger den ekvivalenta r-koordinaten för argumentparen (x,y) .

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

▶ RadKatalog > 

Konverterar argumentet till en vinkel i radianer.

Med vinkelmått Grader:

$(1.5) \blacktriangleright \text{Rad}$	$(0.02618)^r$
--	---------------

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

Med vinkelenhet Nygrader:

$(1.5) \blacktriangleright \text{Rad}$	$(0.023562)^r$
--	----------------

rand()Katalog > 

rand() \Rightarrow uttryck

rand(#Trials) \Rightarrow lista

rand() ger ett slumpvärde mellan 0 och 1.

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

Bestämmer slumpvalsfröet.

RandSeed 1147	Done
$\text{rand}(2)$	$\{0.158206, 0.717917\}$

randBin(n, p) \Rightarrow *uttryck*
randBin($n, p, \#Trials$) \Rightarrow *lista*

randBin(n, p) ger ett reellt slumpstal från en specificerad binomialfördelning.

randBin($n, p, \#Trials$) ger en lista med $\#Trials$ reella slumpstal från en specificerad binomialfördelning.

randInt
(
lowBound, upBound)
 \Rightarrow *uttryck*
randInt
(*lowBound, upBound*
, #Trials) \Rightarrow *lista*

randInt
(
lowBound, upBound)
ger ett slumpstal med heltalsvärde inom det område som specificeras av heltalsgränserna *lowBound* och *upBound*.

randInt
(
lowBound
, upBound, #Trials)
ger en lista med $\#Trials$ slumpstal 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.

RandSeed *Tal*

Om *Number* = 0 ställs fröna in på fabriksinställningarna för slumpvalsgeneratorn. Om *Number* ≠ 0, används det för att generera två frön, vilka lagras i systemvariablerna *seed1* och *seed2*.

RandSeed 1147

Done

rand()

0.158206

real()

Ger argumentets reella del.

real(List1) ⇒ *lista*

Ger de reella delarna av alla element.

real(Matrix1) ⇒ *matrix*

Ger de reella delarna av alla element.

► Rect

Vector ► **Rect**

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

Visar *Vector* i rektangulär form [x, y, z]. Vektorn måste ha dimensionen 2 eller 3 och kan vara en rad eller en kolumn.

Obs: ► **Rect** är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad, och den uppdaterar inte *ans*.

Obs: Se även ► **Polar**, på sidan 109.

complexValue ► **Rect**

Visar *complexValue* i rektangulär form $a+bi$. *complexValue* kan ha valfri komplex form. En inmatning av $re^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

Obs: Du måste använda parenteserna för en $(r \angle \theta)$ polär inmatning.

Med vinkelenhet Radianer:

Med vinkelenhet Nygrader:

$((1 \angle 100)) \text{►Rect}$ *i*

Med vinkelmått Grader:



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

ref ()

ref(Matrix1[, Tol]) \Rightarrow matris

Ger radtrappsteigsformen av *Matrix1*.

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

- Om du använder - eller ställer in **Auto or Approximate** på Approximate, utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som:
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

Undvik odefinierade element i *Matrix1*. De kan leda till oväntade resultat.

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

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

Varningen visas på grund av att det generaliserade elementet $1/a$ inte skulle vara giltigt för $a=0$.

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

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.

$$\text{ref} \left(\begin{array}{ccc} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right) | a=0 \quad \begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array}$$

Obs: Se även **rref ()**, på sidan 130.


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()

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 94.

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

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

Request

Begär *promptString*, *var* [, *DispFlag* [, *statusVar*]]

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

Programmeringskommando: Pausar programmet och visar en dialogruta med meddelandet *promptString* och en svarsruta där användaren ska skriva in svaret.

När användaren skriver in svaret och klickar på **OK** tilldelas variabeln *vardet* värde som står i svarsrutan.

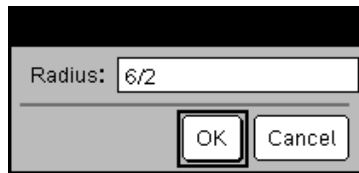
Om användaren klickar på **Cancel** (Avbryt) fortsätter programmet utan att acceptera någon inmatning. Programmet använder det tidigare värdet på *var* om *var* redan var definierad.

Definiera ett program:

```
Definiera begär_demo()=Prgm
  Begär "Radië: ",r
  Disp "Area = ",pi*r^2
EndPrgm
```

Kör programmet och skriv in ett svar:

`request_demo()`



Resultat efter tryckning på **OK**:

Det valfria argumentet *DispFlag* kan vara ett valfritt uttryck.

- Om *DispFlag* utelämnas eller beräknas till **1**, visas promptmeddelandet och användarens svar i Räknares 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.

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

- **Handenhet:** Håll ned  och tryck på  upprepade gånger.

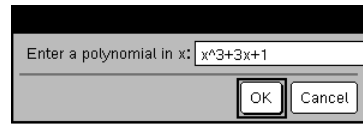
Radie: 6/2
Area= 28,2743

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: $\{-0,322185\}$

- **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 125.



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 123.

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 [*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
```

$$\text{factorial}(3)$$

6

right()

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.

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

$$\text{right}(\text{"Hello"},2) \quad \text{"lo"}$$

$$\text{right}(x<3) \quad 3$$
rk23 ()

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

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

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

Differentialekvation:

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

$$\text{rk23}(0,001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1)$$

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 $\text{Var}0$ till VarMax .

$\text{ListOfDepVars}0$ 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).

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

$$\text{rk23}\left\{0.001 \cdot y \cdot \{100 - y\}, t, y, \{0, 100\}, 10, 1, 1.E-6\right\}$$

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 $y2(0) = 5$

$$\text{rk23}\left\{\begin{cases} y1' + 0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1\right\}$$

0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245

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

diftol är feltoleransen (förinställs på 0,001).

Rot()

Obs: Se även **Nth root template**, på sidan 1.

rotate()

rotate(*IntegerI*[,*#ofRotations*]) \Rightarrow *heltal*

I binärt basläge:

Roterar bitarna i ett binärt heltal. *IntegerI* kan anges i valfri talbas. Det omvandlas automatiskt till 64 positioners binär form. Om storleken på *IntegerI* är alltför stor för denna form, för en symmetrisk modulooperation talet inom området. För mer information, se **► Base2**, på sidan 16.

rotate(0b11111111111111111111111111111111)	
0b100000000000000000000000000000000001	1
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:

Vid exempelvis rotation åt höger:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h8000000000000000000001E3
rotate(0h78E,2)	0h1E38

Varje bit roteras åt höger.

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

0b00000000000001111010110000110101

Biten längst till höger roteras till positionen längst till vänster.

ger:

0b100000000000000111101011000011010

Resultatet visas enligt det inställda basläget.

rotate(*ListI*[,*#ofRotations*]) \Rightarrow *lista*

I decimalt basläge:

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

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

rotate()

Katalog > 

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är -1 (rotera ett element åt höger).

rotate(*String1*[,*#ofRotations*]) \Rightarrow *sträng*

Ger en kopia av *String1* roterad åt höger eller vänster av *#ofRotations*-tecknen. Ändrar inte *String1*.

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är -1 (rotera ett tecken åt höger).

round()

Katalog > 

Ger argumentet avrundat till det specificerade antalet siffror efter decimalpunkten.

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

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(*List1*[,*digits*]) \Rightarrow *lista*

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

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

round(*Matrix1*[,*digits*]) \Rightarrow *matrix*

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

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(*Matrix1*, *rIndex1*, *rIndex2*) \Rightarrow *matrix*

Ger en kopia av *Matrix1* med rad *rIndex2* ersatt av summan av raderna *rIndex1* och *rIndex2*.

rowDim()

Katalog > 

rowDim(Matrix) ⇒ uttryck

Ger antalet rader i Matrix.

Obs: Se även **colDim()**, på sidan 23.

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

rowNorm()

Katalog > 

rowNorm(Matrix) ⇒ uttryck

Ger maximum av summorna av absolutbeloppen på elementen i raderna i Matrix.

Obs: Alla matriselement måste förenklas till tal. Se även **colNorm()**, på sidan 23.

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

rowSwap()

Katalog > 

rowSwap(Matrix1, rIndex1, rIndex2) ⇒ matrix

Ger Matrix1 med raderna rIndex1 och rIndex2 växlade.

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

rref()


Katalog > 



rref(Matrix1[, Tol]) ⇒ matrix

Ger den reducerade radtrappstegsformen av Matrix1.

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

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

 rref $\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
---	--

- 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 121.

S

sec()

`sec(List1)` ⇒ lista

I vinkelläget Grader:

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.

sec⁻¹()

`sec-1(List1)` ⇒ lista

I vinkelläget Grader:

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

I vinkelläget Nygrader:

I vinkelläget Radianer:

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

sech()

`sech(List1)` ⇒ lista

sech⁻¹()

`sech-1(List1)` ⇒ lista

I vinkelläget Radianer och i Rektangulärt komplext läge:

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

Send

`SendUtrEllerSträng1 [, UtrEllerSträng2] ...`

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

`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.

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 58), **GetStr** (på sidan 64) och **eval()** (på sidan 46).

Hubb-meny

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

Send "SET COLOR.BLUE ON TIME .5"	Done
----------------------------------	------

Exempel: Begär nuvarande värde från hubbens inbyggda ljusnivåsensor. Ett **Get**-kommando hämtar värdet och tilldelar det till variabeln `lightval`.

Send "READ BRIGHTNESS"	Done
Get <code>lightval</code>	Done
<code>lightval</code>	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.

<code>n:=50</code>	50
<code>m:=4</code>	4
Send "SET SOUND eval(m·n)"	Done
<code>iostr.SendAns</code>	"SET SOUND 200"

seq()

`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.


Katalog > 


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

Obs: För att få ett närmevärde,

Handenhet: Tryck på  .

Windows®: Tryck på **Ctrl+Enter**.

Macintosh®: Tryck på +Enter.

iPad®: Håll ned **enter** och välj .

$$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right) \quad 1.54977$$

seqGen()

seqGen(Expr, Var, depVar, {Var0, VarMax}, [ListOfInitTerms [, VarStep [, CeilingValue]])] ⇒ lista

Genererar en lista på termer för talföljden $\text{depVar}(Var)=Expr$ enligt följande: Ökar den oberoende variabeln Var från $Var0$ till $VarMax$ i steg om $VarStep$, utvärderar $\text{depVar}(Var)$ 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]])] ⇒ matris

Genererar en matris med termer för ett system (eller en lista) av talföljder $ListOfDepVars(Var)=ListOrSystemOfExpr$ enligt följande: Ökar den oberoende variabeln Var från $Var0$ till $VarMax$ i steg om $VarStep$, utvärderar $ListOfDepVars(Var)$ för motsvarande värden på Var med formeln $ListOrSystemOfExpr$ och $MatrixOfInitTerms$ och ger resultaten som en matris.

De ursprungliga innehållen i Var är oförändrade när **seqGen()** har beräknats.

Det förinställda värdet på $VarStep = 1$.

Genererar de första 5 termerna i talföljden $u(n) = u(n-1)^2/2$, med $u(1)=2$ och $VarStep=1$.

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

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:

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u(n-1)}{2} + u(n-1)\right\}, n, \{u1, u2\}, \{1, 5\}, \left[\begin{array}{c} - \\ 2 \end{array}\right]\right) \\ \left[\begin{array}{ccccc} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{array}\right]$$

Obs: Tomrummet () i den initiala termmatrisen ovan används för att indikera att den initiala termen för $u1(n)$ beräknas med den explicita talföljdsformeln $u1(n)=1/n$.

seqn(*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 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 $nstep=1$

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

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

$$\left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right)$$

$$\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

setMode()

setMode(*modeNameInteger*, *settingInteger*)
 \Rightarrow *heltal*

setMode(*list*) \Rightarrow *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.

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.

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.

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 63.

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

Lägets namn	Lägesheltal	Heltal för inställningar
Display Digits (Visa siffror)	1	1=Float, 2=Float1, 3=Float2, 4=Float3, 5=Float4, 6=Float5, 7=Float6, 8=Float7, 9=Float8, 10=Float9, 11=Float10, 12=Float11, 13=Float12, 14=Fix0, 15=Fix1, 16=Fix2, 17=Fix3, 18=Fix4, 19=Fix5, 20=Fix6, 21=Fix7, 22=Fix8, 23=Fix9, 24=Fix10, 25=Fix11, 26=Fix12
Angle (Vinkel)	2	1=Radian, 2=Degree, 3=Gradian
Exponential Format (Exponentiellt format)	3	1=Normal, 2=Scientific, 3=Engineering
Real or Complex (Reellt eller Komplext)	4	1=Real, 2=Rectangular, 3=Polar
Auto or Approx. (Auto eller Ungefärlig)	5	1=Auto, 2=Approximate

Lägets namn	Läges-heltal	Heltal för inställningar
Vector Format (Vektorformat)	6	1=Rectangular, 2=Cylindrical, 3=Spherical
Base (Bas)	7	1=Decimal, 2=Hex, 3=Binary

shift()

Katalog > 

shift(*IntegerI* [, #ofShifts]) ⇒ *heltal*

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

0b0000000000000111101011000011010

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(*ListI* [, #ofShifts]) ⇒ *lista*

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

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 > 

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".

shift(String l [, #ofShifts]) ⇒ *sträng*

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

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är -1 (skifta ett tecken åt höger).

Tecken som infogas i början eller slutet av *string* av skiftningen får ett mellanslag.

shift("abcd")	" abc"
shift("abcd", -2)	" ab"
shift("abcd", 1)	"bcd "

sign()

Katalog > 

sign(List l) ⇒ *lista*

sign(Matrix l) ⇒ *matrix*

Om det komplexa formatläget är Real:

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 matrix, tecknen för alla element.

simult()

Katalog > 

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

Ger en kolumnvektor som innehåller lösningarna för ett system av linjära ekvationer.

Obs: Se även **linSolve()**, på sidan 81.

Lös för x och y:

$$x + 2y = 1$$

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

simult	$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} -3 \\ 2 \end{pmatrix}$
--------	--	---	---

Lösningen är x=-3 och y=2.

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

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:

$$ax + by = 1$$

$$cx + dy = 2$$

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(List1) ⇒ *lista*

I vinkelläget Grader:

sin(List1) ger en lista på sinus för alla element i *List1*.

sin()

tangente 

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

$\sin(\text{squareMatrix1}) \Rightarrow \text{kvadratMatrix}$

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

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

I vinkelläget Nygrader:

I vinkelläget Radianer:

I vinkelläget Radianer:

$$\sin \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$$

$\sin^{-1}()$

tangente 

$\sin^{-1}(\text{List1}) \Rightarrow \text{lista}$

$\sin^{-1}(\text{List1})$ 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(...)**.

$\sin^{-1}(\text{squareMatrix1}) \Rightarrow \text{kvadratMatrix}$

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

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

I vinkelläget Grader:

I vinkelläget Nygrader:

I vinkelläget Radianer:

I vinkelläget Radianer och i Rektangulärt komplex format:

$$\sin^{-1} \begin{pmatrix} 1 & 5 \\ 4 & 2 \end{pmatrix} = \begin{bmatrix} -0.174533 - 0.12198 \cdot i & 1.745533 - 2.35591 \cdot i \\ 1.39626 - 1.88473 \cdot i & 0.1745533 - 0.593162 \cdot i \end{bmatrix}$$

sinh()

Katalog > 

$\sinh(\text{List1}) \Rightarrow \text{lista}$

$$\sinh(1.2) = 1.50946$$
$$\sinh(\{0, 1, 2, 3\}) = \{0, 1.50946, 10.0179\}$$

sinh (*List1*) ger en lista på hyperboliska sinus för varje element i *List1*.

sinh(*squareMatrix1*) \Rightarrow *kvadratMatrix*

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

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

I vinkelåget Radianer:

$\sinh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$			

sinh⁻¹(*List1*) \Rightarrow *lista*

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

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

sinh⁻¹(*squareMatrix1*) \Rightarrow *kvadratMatrix*

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

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet innehåller alltid tal med flytande komma.

I vinkelåget Radianer:

$\sinh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$			

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 143.)

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 kategorikoder 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 206).

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 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>

SortA

Katalog > 

SortA *List1*[, *List2*] [, *List3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

SortA *Vector1*[, *Vector2*] [, *Vector3*] ...

SortA <i>list1</i>	Done
--------------------	------

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>	Done
--------------------------	------

<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 206.

SortD

Katalog > 

SortD *List1*[, *List2*] [, *List3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

SortD *Vector1*[, *Vector2*] [, *Vector3*] ...

$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
---------------------------------	---------------

Identisk med **SortA** med undantag för att **SortD** sorterar elementen i fallande ordning.

SortD <i>list1,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 206.

►Sphere

Katalog > 

Vector ►Sphere

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

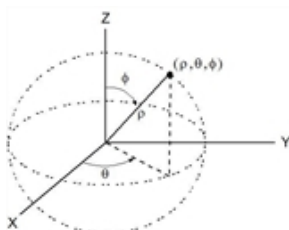
Visar rad- eller kolumnvektorn i sfärisk form $[\rho \angle \theta \angle \phi]$.

$[1 \ 2 \ 3] \text{►Sphere}$
$[3.74166 \ \angle 1.10715 \ \angle 0.640522]$

$\left(\begin{matrix} 2 & \angle \frac{\pi}{4} & 3 \end{matrix} \right) \text{►Sphere}$
$[3.60555 \ \angle 0.785398 \ \angle 0.588003]$

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()

`sqrt(List1)⇒lista`

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
"t"	0.998053
"Resid"	"{...}"

<i>stat.values</i>	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4,0.4,0.2,0,-,0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.ox	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.oy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.ox1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.ox2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.̄x
stat.b9	stat.FBlock	stat.̂p	stat.Σx ²	stat.̄x1
stat.b10	stat.Fcol	stat.̂p1	stat.Σxy	stat.̄x2
stat.bList	stat.FInteract	stat.̂p2	stat.Σy	stat.̄xDiff
stat.χ ²	stat.FreqReg	stat.̂pDiff	stat.Σy ²	stat.̄xList
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.ComplList	stat.LowerVal	stat.PValCol	stat.SEPred	stat.ȳ
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat.ȳ
stat.CookDist	stat.MaxX	stat.PValRow	stat.SESlope	stat.ȳList
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

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()

`stDevPop(List[, freqList]) ⇒ uttryck`

I vinkelläget Radianer och i Auto-läge:

Ger populationens 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 206.

`stDevPop(Matrix1[, freqMatrix]) ⇒ matris`

Ger en radvektor med populationens standardavvikelser för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

stDevSamp()

`stDevSamp(List[, freqList]) ⇒ uttryck`

Ger urvalets standardavvikelse för elementen i *List*.

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

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

stDevSamp(*MatrixI* [, *freqMatrix*]) ⇒ *matrix*

Ger en radvektor med urvalets standardavvikelser för kolumnerna i *MatrixI*.

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

Obs: *MatrixI* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 206.

Stop

Stop

Programmeringskommando: Avslutar programmet.

Stop är ej tillåtet i funktioner.

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

<i>i</i> :=0	0
Define <i>progI</i> ()=Prgm	<i>Done</i>
For <i>i</i> ,1,10,1	
If <i>i</i> =5	
Stop	
EndFor	
EndPrgm	
<i>progI</i> ()	<i>Done</i>
<i>i</i>	5

Store

Se → (store), på sidan 189.

string()

string(*Expr*) ⇒ *sträng*

Förenklar *Expr* och ger resultatet som en teckensträng.

subMat()Katalog > 

subMat(*Matrix1* [, *startRow*] [, *startCol*] [, *endRow*] [, *endCol*]) ⇒ *matrix*

Ger specificerad undermatrix av *Matrix1*.

Förvalinställningar: *startRow*=1, *startCol*=1, *endRow*=sista rad, *endCol*=sista kolumn.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
$\text{subMat}(m1,2,1,3,2)$	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
$\text{subMat}(m1,2,2)$	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Sum (Sigma)Se $\Sigma()$, på sidan 181.**sum()**Katalog > 

sum(*List* [, *Start*] [, *End*]) ⇒ *uttryck*

Ger summan av elementen i *List*.

Start och *end* är valfria. De specificerar ett område med element.

Varje tomt argument ger ett tomt resultat. Tomma element i *List* ignoreras. För mer information om tomma element, se på sidan 206.

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 206.

$\text{sum} \left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \right)$	$[5 \ 7 \ 9]$
$\text{sum} \left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right)$	$[12 \ 15 \ 18]$
$\text{sum} \left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, 2, 3 \right)$	$[11 \ 13 \ 15]$

sumIf()Katalog > 

sumIf(*List*, *Criteria* [, *SumList*]) ⇒ *värde*

Ger den totala kumulerade summan av alla element i *List* som uppfyller specificerade *Criteria*. Du kan också specificera en alternativ lista, *sumList*, för att ge elementen som skall kumuleras.

List kan vara ett uttryck, en lista eller en matris. *SumList*, om specificerad, måste ha samma dimension(er) som *List*.

Criteria kan vara:

- Ett värde, ett uttryck eller en sträng. Som exempel ackumulerar **34** endast de element i *List* som förenklas till värdet 34.
- Ett booleskt uttryck som innehåller symbolen **?** fungerar som platshållare för varje element. Som exempel ackumulerar **?<10** endast de element i *List* som är lägre än 10.

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 206.

Obs: Se även `countIf()`, på sidan 29.

Ger ett ekvationssystem formaterat som en lista. Du kan också skapa ett system med en mall.

$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=8 \end{cases}, x, y\right) \quad x=4 \text{ and } y=-4$$

T(transponera)Katalog > *MatrixI*T⇒*matrix*

Ger den komplexkonjugerade transponeringen av *MatrixI*.

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

tan() tangent**tan(ListI)**⇒*lista*

I vinkelläget Grader:

tan(ListI) ger en lista på tangensen för alla element i *ListI*.

I vinkelläget Nygrader:

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.

I vinkelläget Radianer:

tan(squareMatrixI)⇒*kvadratMatrix*

I vinkelläget Radianer:

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

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

$$\tan \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan⁻¹() tangent**tan⁻¹(ListI)**⇒*lista*

I vinkelläget Grader:

tan⁻¹(ListI) ger en lista på den inversa tangensen för varje element i *ListI*.

$$\tan^{-1}(1) \quad 45$$

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

I vinkelläget Nygrader:

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

$$\tan^{-1}(1) \quad 50$$

I vinkelläget Radianer:

$$\tan^{-1}(\{0,0.2,0.5\}) \quad \{0,0.197396,0.463648\}$$

tan⁻¹(*squareMatrix1*) ⇒ *kvadratMatrix*

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

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

I vinkelläget Radianer:

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

tanh(*List1*) ⇒ *lista*

tanh(*List1*) ger en lista på den hyperboliska tangensen för varje element i *List1*.

tanh(*squareMatrix1*) ⇒ *kvadratMatrix*

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

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

I vinkelläget Radianer:

$$\tanh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

tanh⁻¹(*List1*) ⇒ *lista*

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*

I Rektangulärt komplext format:

I vinkelläget Radianer och i Rektangulärt komplext format:

$\tanh^{-1}()$

Katalog > 

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.

$$\tanh^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} -0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.94730 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901 \end{bmatrix}$$

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

tCdf()

Katalog > 

tCdf(*lowBound*,*upBound*,*df*) \Rightarrow 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*.

Text

Katalog > 

TextfrågeSträng[, *DispFlagga*]

Programmeringskommando: Pausar programmet och visar teckensträngen *frågeSträng* i en dialogruta.

När användaren väljer **OK** fortsätter exekveringen av programmet.


Det valfria argumentet *flagga* kan vara ett valfritt uttryck.

- Om *DispFlagga* utelämnas eller beräknas till **1** sparas textmeddelandet i Räknares historik.
- Om *DispFlagga* beräknas till **0** sparas textmeddelandet inte i historiken.

Om programmet behöver ett svar som skrivs in av användaren, se **Request**, på sidan 123 eller **RequestStr**, på sidan 125.

Obs: Du kan använda detta kommando inom ett användardefinierat program, men inte inom en funktion.

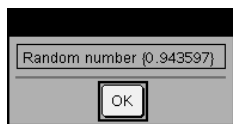
Definiera ett program som pausar för att visa vart och ett av fem slumpstal i en dialogruta.

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**.

```
Define text_demo()=Prgm
  For i,1,5
    stringfo:="Random number " &
string(rand(i))
    Text stringfo
  EndFor
EndPrgm
```

Kör programmet:
text_demo()

Exempel på en dialogruta:



Then

Se If, på sidan 67.

tIntervalKatalog > **tInterval** *List[,Freq[,CLevel]]*

(Indatalista)

tInterval $\bar{x},sx,n[,CLevel]$

(Summary stats indata)

Beräknar ett *t*-konfidsintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidsintervall 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 *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 143.)

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 206).

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 standardavvikelse för <i>List 1</i> och <i>List 2</i>
stat.n1, stat.n2	Antalet samplings i datasekvenser
stat.sp	Den sammanslagna standardavvikelsen. Beräknas när <i>Pooled</i> = YES.

tPdf()

tPdf(*XVal, df*) \Rightarrow *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*.

Ger spåret (summan av alla elementen på huvuddiagonalen) av *squareMatrix*.

Try

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 216).

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.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, [-1 \ 2 \ -3.1]\right)$$

Obs: Se även **ClrErr**, på sidan 22 och **PassErr**, på sidan 108.

```
Define prog1()=Prgm
```

```
Try
```

```
z:=z+1
```

```
Disp "z incremented."
```

```
Else
```

```
Disp "Sorry, z undefined."
```

```
EndTry
```

```
EndPrgm
```

Done

```
z:=1:prog1()
```

```
z incremented.
```

Done

```
DelVar z:prog1()
```

```
Sorry, z undefined.
```

Done

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 $\mu_0, List[, Freq[, Hypoth]]$

(Indatalista)

tTest $\mu_0, \bar{x}, sx, n, [Hypoth]$

(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 143.)

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 206).

Resultatvariabel	Beskrivning
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas

Resultatvariabel	Beskrivning
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

Katalog > 

tTest_2Samp *List1, List2[,Freq1[,Freq2
[,Hypoth[,Pooled]]]]*

(Indatalista)

tTest_2Samp $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2[,Hypoth
[,Pooled]]$

(Summary stats indata)

Beräknar ett 2-sampel *t*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

Testa $H_0: \mu_1 = \mu_2$, mot en av följande:

För $H_a: \mu_1 < \mu_2$, ställ *Hypoth*<0

För $H_a: \mu_1 \neq \mu_2$ (förinställning), 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 206).

Resultatvariabel	Beskrivning
stat.t	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos <i>t</i> -statistiken
stat. $\bar{x}1$, stat. $\bar{x}2$	Medelvärden hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i <i>List 1</i> och <i>List 2</i>

Resultatvariabel	Beskrivning
stat.n1, stat.n2	Storlek på urvalen
stat.sp	Den sammanslagna standardavvikelsen. Beräknad när <i>Pooled</i> =1.

tvmFV()

Katalog > 

tvmFV(*N,I,PV,Pmt,[PpY],[CpY],[PmtAt]*) \Rightarrow 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 158. Se även **amortTbl()**, på sidan 7.

tvmI()

Katalog > 

tvmI(*N,PV,Pmt,FV,[PpY],[CpY],[PmtAt]*) \Rightarrow värde

tvmI(240,100000,-1000,0,12,12) 10.5241

Finansiell funktion som beräknar räntesatsen per år.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 158. 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 158. 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

tvmPmt()Katalog > 

Finansiell funktion som beräknar beloppet på varje betalning.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 158. Se även **amortTbl()**, på sidan 7.

tvmPV()Katalog > 

tvmPV($N, I, Pmt, FV, [PpY], [CpY], [PmtAt]$) ⇒ 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 158. Se även **amortTbl()**, på sidan 7.

TVM-argument*	Beskrivning	Datatyp
N	Antal betalningsperioder	reellt tal
I	Årlig räntesats	reellt tal
PV	Nuvärde	reellt tal
Pmt	Betalningsbelopp	reellt tal
FV	Framtida värde	reellt tal
PpY	Antal betalningar per år, förinställning = 1	heltal > 0
CpY	Ränteperioder per år, förinställning = 1	heltal > 0
$PmtAt$	Betalning som skall betalas i slutet (end) eller i början (beginning) av varje period, förinställning = end	heltal (0 = end, 1 = beginning)

* Dessa argumentnamn avseende tidsjusterat pengavärde påminner om namnen på TVM-variablerna (till exempel, **tvm.pv** och **tvm.pmt**) som används av Finance Solver i applikationen *Calculator*. Finansiella funktioner lagrar dock inte sina argumentvärden eller resultat i TVM-variablerna.

TwoVarKatalog > 

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

Utför tvåvariabelstatistik. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 kategorikoder 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 206.

Resultatvariabel	Beskrivning
stat. \bar{x}	Medelvärde av x -värden
stat. x	Summa av x -värden
stat. x^2	Summa av x^2 -värden
stat. s_x	Standardavvikelse för x (sampling)
stat. σ_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 y^2 -värden

Resultatvariabel	Beskrivning
stat.sy	Standardavvikelse för y (sampling)
stat.y	Populationens standardavvikelse för y
stat.xy	Summa av $x \cdot y$ -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
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 > 

$\text{unitV}(\text{Vector1}) \Rightarrow \text{vektor}$

Ger en radenhets- eller kolumnenhetsvektor beroende på formen hos *Vector1*.

Vector1 måste vara antingen en enradsmatris eller en enkolumnsmatris.

unLock

Katalog > 

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

<i>a</i> :=65	65
---------------	----

unLock*Var*.

Lock <i>a</i>	Done
---------------	------

Låser upp den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.

getLockInfo(<i>a</i>)	1
-------------------------	---

<i>a</i> :=75	"Error: Variable is locked."
---------------	------------------------------

DelVar <i>a</i>	"Error: Variable is locked."
-----------------	------------------------------

Se **Lock**, på sidan 85 och **getLockInfo()**, på sidan 63.

Unlock <i>a</i>	Done
-----------------	------

<i>a</i> :=75	75
---------------	----

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

V

varPop()

Katalog > 

varPop(*List*[, *freqList*])⇒*uttryck*


Ger populationsvariansen för *List*.

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

Obs: *List* måste innehålla minst två element.

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 206.

varSamp()

Katalog > 

varSamp(*List*[, *freqList*])⇒*uttryck*

Ger sampelvariansen för *List*.

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

Obs: *List* måste innehålla minst två element.

Om ett element i endera listan är tomt ignoreras detta element och även motsvarande element i den andra listan ignoreras. För mer information om tomma element, se på sidan 206.

varSamp()

Katalog > 

$\text{varSamp}(\text{Matrix1}, \text{freqMatrix}) \Rightarrow \text{matrix}$

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

Varje *freqMatrix*-element räknar antalet konsekutiva förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader.

Om ett element i endera matrisen är tomt ignoreras detta element och även motsvarande element i den andra matrisen ignoreras. För mer information om tomma element, se på sidan 206.

$\text{varSamp} \left(\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{pmatrix} \right)$	$[4.75 \ 1.03 \ 4]$
$\text{varSamp} \left(\begin{pmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{pmatrix}, \begin{pmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{pmatrix} \right)$	$[3.91731 \ 2.08411]$

W

Wait

Katalog > 



Wait *tidISekunder*

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.

För att vänta 4 sekunder:

Wait 4

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"

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

warnCodes ()

warnCodes(*Expr1*, *StatusVar*) \Rightarrow 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 225.

when()

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.

$\text{when}(x < 0, x + 3) x = 5$	undef
-------------------------------------	-------

when()

Katalog > 

when() är användbar för att definiera rekursiva funktioner.

$\text{when}(n > 0, n \cdot \text{factoral}(n-1), 1) \rightarrow \text{factoral}(n)$	Done
$\text{factoral}(3)$	6
3!	6

While

Katalog > 

While Condition

Block

EndWhile

Exekverar påståendena i *Block* så länge *Condition* är sant.

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

Obs för att mata in exemplet: Se avsnittet 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
$\text{sum_of_recip}(3)$	$\frac{11}{6}$

X

xor

Katalog > 

BooleskUttr1 **xor** *BooleskUttr2* ger
Boolesk uttryck

true xor true	false
5 > 3 xor 3 > 5	true

BooleskLista1 **xor** *BooleskLista2* ger
Boolesk lista

BooleskMatris1 **xor** *BooleskMatris2* ger
Boolesk matris

Ger resultatet sant om *BooleanExpr1* är sant och *BooleanExpr2* är falskt, eller vice versa.

Ger resultatet falskt om båda argumenten är sanna eller falska. Ger ett förenklat booleskt uttryck om något av argumenten inte kan lösas om till sant eller falskt.

Obs: Se **eller**, på sidan 106.

Integer1 xor Integer2 \Rightarrow *heltal*

Jämför två reella heltal bit för bit med en **xor**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om en bit (men inte båda) är 1. Resultatet blir 0 om båda bitarna är 0 eller 1. Det erhållna värdet representerar bitresultaten och visas enligt det inställda basläget.

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

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

Obs: Se **eller**, på sidan 106.

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

0h7AC36 xor 0h3D5F	0h79169
--------------------	---------

I binärt basläge:

0b100101 xor 0b100	0b100001
--------------------	----------

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

Z

zInterval

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 143.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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_1Prop

Katalog > 

zInterval_1Prop $x, n [, CLevel]$

Beräknar ett 1-proportion z -konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

x är ett icke negativt heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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 143.)

$x1$ och $x2$ är icke negativa heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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 $\sigma_1, \sigma_2, List1, List2[, Freq1$
 $[, Freq2, [CLevel]]]$

(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 143.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 206).

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

Resultatvariabel	Beskrivning
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>
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.r1, stat.r2	Kända populationsstandardavvikelser för datasekvenserna <i>List 1</i> och <i>List 2</i>

zTest

Katalog > 

zTest $\mu0, \sigma, List, [Freq, Hypoth]$

(Indatalista)

zTest $\mu0, \sigma, \bar{x}, n, [Hypoth]$

(Summary stats indata)

Utför ett z-test med frekvensen *freqlist*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 206).

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 $p_0, x, n[, Hypoth]$

Beräknar ett 1-proportion z-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

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 206).

Resultatvariabel	Beskrivning
stat.p0	Hypotetisk populationsproportion
stat.z	Standardnormalvärde beräknat för proportionen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.p̂	Uppskattad urvalsproportion
stat.n	Urvalets storlek

zTest_2Prop $x_1, n_1, x_2, n_2[, Hypoth]$

Beräknar ett 2-proportion z-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 143.)

x_1 och x_2 är icke negativa heltal.

Testa $H_0: p_1 = p_2$ mot en av följande:

För $H_a: p_1 > p_2$, ställ *Hypoth*>0

För $H_a: p_1 \neq p_2$ (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 206).

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 samlingar i försöksomgång 1 och 2

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 143.)

Testa $H_0: \mu_1 = \mu_2$, mot en av följande:

För $H_a: \mu_1 < \mu_2$, ställ *Hypoth*<0

För $H_a: \mu_1 \neq \mu_2$ (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 206).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas

Resultatvariabel	Beskrivning
stat. \bar{x} 1, stat. \bar{x} 2	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)

$\boxed{+}$ tangent

Ger summan av de två argumenten.

56	56
56+4	60
60+4	64
64+4	68
68+4	72

$List1 + List2 \Rightarrow lista$

$Matrix1 + Matrix2 \Rightarrow matrix$

Ger en lista (eller matris) som innehåller summorna av motsvarande element i $List1$ och $List2$ (eller $Matrix1$ och $Matrix2$).

Argumenten måste ha samma dimensioner.

$15 + \{10, 15, 20\}$	$\{25, 30, 35\}$
$\{10, 15, 20\} + 15$	$\{25, 30, 35\}$

Obs: Använd .+ (punkt plus) för att lägga till ett uttryck till varje element.

$20 + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
---	--

-(subtrahera)

$\boxed{-}$ tangent

$List1 - List2 \Rightarrow lista$

$Matrix1 - Matrix2 \Rightarrow matrix$

Subtraherar varje element i $List2$ (eller $Matrix2$) från motsvarande element i $List1$ (eller $Matrix1$) och ger resultatet.

Argumenten måste ha samma dimensioner.

$15 - \{10, 15, 20\}$	$\{5, 0, -5\}$
$\{10, 15, 20\} - 15$	$\{-5, 0, 5\}$

Obs: Använd .- (punkt minus) för att subtrahera ett uttryck från varje element.

$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$
---	--

· (multiplicera)



Ger produkten av de två argumenten.

$List1 \cdot List2 \Rightarrow lista$

Ger en lista som innehåller produkterna av motsvarande element i $List1$ och $List2$.

Listorna måste ha samma dimensioner.

$Matrix1 \cdot Matrix2 \Rightarrow matrix$

Ger en matris över produkten av $Matrix1$ och $Matrix2$.

Antalet kolumner i $Matrix1$ måste vara lika med antalet rader i $Matrix2$.

Obs: Använd · (punkt multiplicera) för att multiplicera ett uttryck med varje element.

/ (dividera)



Obs: Se även **Fraction template**, på sidan 1.

$List1 / List2 \Rightarrow lista$

Ger en lista på kvoterna av $List1$ dividerat med $List2$.

$$\frac{\begin{Bmatrix} 1, 2, 3 \\ 4, 5, 6 \end{Bmatrix}}{\begin{Bmatrix} 0.25, \frac{2}{5}, \frac{1}{2} \end{Bmatrix}}$$

Listorna måste ha samma dimensioner.

Obs: Använd / (punkt dividera) för att dividera ett uttryck med varje element.

^ (potens)



$List1 \wedge List2 \Rightarrow lista$

Ger det första argumentet upphöjt till det andra argumentets potens.

Obs: Se även **Exponent template**, på sidan 1.

Ger, för en lista, elementen i $List1$ upphöjda till potensen för motsvarande element i $List2$.

^ (potens)

 tangent

I det reella området använder potenser i bråkform som har reducerade exponenter med udda nämnare den reella delen kontra principaldelen för komplext läge.

$$\{1,2,3,4\}^{-2} \quad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$$

squareMatrix1 ^ *integer* ⇒ *matrix*

Ger *squareMatrix1* upphöjd till potensen för *integer* (heltal).

squareMatrix1 måste vara en kvadratisk matris.

Om *integer* = -1 beräknas den inversa matrisen.

Om *integer* < -1 beräknas den inversa matrisen till en lämplig positiv potens.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 \quad \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \quad \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \quad \begin{bmatrix} 11 & -5 \\ 2 & 2 \\ -15 & 7 \\ 4 & 4 \end{bmatrix}$$

x² (kvadrat)

 tangent

Ger kvadraten på argumentet.

*List1*² ⇒ *lista*

Ger en lista med kvadraterna på elementen i *List1*.

*squareMatrix1*² ⇒ *matrix*

Ger en matris över kvadraten på *squareMatrix1*. Detta är inte detsamma som att beräkna kvadraten på varje element. Använd .^2 för att beräkna kvadraten på varje element.

$$4^2 \quad 16$$

$$\{2,4,6\}^2 \quad \{4,16,36\}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2 \quad \begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}.^2 \quad \begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$$

.+ (punkt addera)

 tangent

Matrix1 .+ *Matrix2* ⇒ *matrix*

Matrix1 .+ *Matrix2* ger en matris som är summan av varje par av motsvarande element i *Matrix1* och *Matrix2*.

.- (punkt subtrahera)

  tangenter

$Matrix1 .- Matrix2 \Rightarrow matrixx$

$Matrix1 .- Matrix2$ ger en matris som är skillnaden mellan varje par av motsvarande element i $Matrix1$ och $Matrix2$.

.· (punkt multiplicera)

  tangenter

$Matrix1 .· Matrix2 \Rightarrow matrix$

$Matrix1 .· Matrix2$ ger en matris som är produkten av varje par av motsvarande element i $Matrix1$ och $Matrix2$.

./ (punkt dividera)

  tangenter

$Matrix1 ./ Matrix2 \Rightarrow matrix$

$Matrix1 ./ Matrix2$ ger en matris som är kvoten av varje par av motsvarande element i $Matrix1$ och $Matrix2$.

.^ (punkt potens)

  tangenter

$Matrix1 .^ Matrix2 \Rightarrow matrix$

$Matrix1 .^ Matrix2$ ger en matris där varje element i $Matrix2$ är exponenten för motsvarande element i $Matrix1$.

-(negation)

  tangent

$-List1 \Rightarrow lista$

$-Matrix1 \Rightarrow matrix$

Ger argumentets negation.

Ger, för en lista eller matris, alla elementen negerade.

Om argumentet är ett binärt eller hexadecimalt heltal ger negationen tvåkomplementet.

I binärt basläge:

Viktigt: Noll, inte bokstaven O.

```
-0b100101
0b11111111111111111111111111111111▶
```

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

% (procent)

  **tangent**

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 närmevärde,

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.

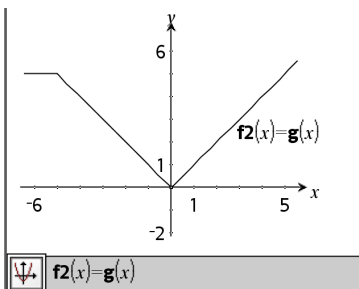
Obs för att mata in exemplet: Se avsnittet Räkna i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Exempel på funktion som använder matchande testsymboler: =, ≠, <, ≤, >, ≥

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

Done

Resultat från plottning av $g(x)$



≠ (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 \neq

< (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 $Expr2$.

Ger resultatet falskt om $Expr1$ bestäms vara större än eller lika med $Expr2$.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

≤ (mindre än eller lika med)

  tangenter

$Expr1 \leq Expr2 \Rightarrow$ Booleskt uttryck

Se exemplet “=” (lika med).

$List1 \leq List2 \Rightarrow$ Boolesk lista

$Matrix1 \leq Matrix2 \Rightarrow$ Boolesk matris

≤ (mindre än eller lika med)

ctrl = tangenter

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 <=

> (större än)

ctrl = 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 större än $Expr2$.

Ger resultatet falskt om $Expr1$ bestäms vara mindre än eller lika med $Expr2$.

Allt annat ger en förenklad form av ekvationen.

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 $Expr2$.

\geq (större än eller lika med)

  tangenter

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 $\gt;$

\Rightarrow (logisk implikation)

  knappar

BoolesktUttr1 \Rightarrow *BoolesktUttr2* ger
Booleskt uttryck

$5 > 3$ or $3 > 5$	true
--------------------	------

$5 > 3 \Rightarrow 3 > 5$	false
---------------------------	-------

BooleskLista1 \Rightarrow *BooleskLista2* 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** \langle argument1 \rangle or \langle argument2 \rangle 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

$5 > 3$ xor $3 > 5$	true
---------------------	------

$5 > 3 \Leftrightarrow 3 > 5$	false
-------------------------------	-------

BooleskLista1 \Leftrightarrow *BooleskLista2* ger
Boolesk lista

3 xor 4	7
---------	---

$3 \Leftrightarrow 4$	-8
-----------------------	----

BooleskMatris1 \Leftrightarrow *BooleskMatris2* ger
Boolesk matris

$\{1,2,3\}$ xor $\{3,2,1\}$	$\{2,0,2\}$
-----------------------------	-------------

$\{1,2,3\} \Leftrightarrow \{3,2,1\}$	$\{-3,-1,-3\}$
---------------------------------------	----------------

Heltal1 \Leftrightarrow *Heltal2* ger *Heltal*

⇔ (logisk dubbel implikation, XNOR)

ctrl = knappar

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 $\lt;=>$

! (fakultet)

?!▶ tangent

$List1! \Rightarrow lista$

$5!$ 120

$Matrix1! \Rightarrow matrix$

$\{\{5,4,3\}\}!$ $\{120,24,6\}$

Ger argumentets fakultet.

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$ $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Ger, för en lista eller matris, en lista eller matris med elementens fakulteter.

& (lägg till)

ctrl 📖 tangenter

$String1 \& String2 \Rightarrow sträng$

"Hello "&"Nick" "Hello Nick"

Ger en textsträng som är $String2$ bifogad till $String1$.

d() (derivata)

Katalog > 📖

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

√() (kvadratrot)

ctrl x² tangenter

$\sqrt{List1} \Rightarrow lista$

Ger kvadratroten ur argumentet.

Ger, för en lista, kvadratrötterna ur alla element i $List1$.

$\sqrt{}$ (kvadrattrot)

ctrl x² tangenter

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

Obs: Se även [Square root template](#), på sidan 1.

Π (prodSeq)

Katalog > 

$\Pi(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow$ 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](#) (Π), på sidan 5.

$\Pi(\text{Expr1}, \text{Var}, \text{Low}, \text{Low}-1) \Rightarrow 1$

$\Pi(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow 1/\Pi(\text{Expr1}, \text{Var}, \text{High}+1, \text{Low}-1)$ om *High* < *Low*-1

Produktformlerna som används har härletts från följande referens:

Ronald L. Graham, Donald E. Knuth och Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\prod_{k=4}^3 (k) \quad 1$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) \quad 6$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) \cdot \prod_{k=2}^4 \left(\frac{1}{k}\right) \quad \frac{1}{4}$$

Σ (sumSeq)

Katalog > 

$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow$ 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.

$$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{Low}-1) \Rightarrow 0$$

$$\Sigma(\text{Expr1}, \text{Var}, \text{Low}, \text{High}) \Rightarrow -\Sigma(\text{Expr1}, \text{Var}, \text{High}+1, \text{Low}-1) \text{ om } \text{High} < \text{Low}-1$$

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.

$$\sum_{k=4}^3 (k) = 0$$

$$\sum_{k=4}^1 (k) = -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) = 4$$

 $\Sigma\text{Int}()$

$\Sigma\text{Int}(\text{NPmt1}, \text{NPmt2}, \text{N}, \text{I}, \text{PV}, [\text{Pmt}], [\text{FV}], [\text{PpY}], [\text{CpY}], [\text{PmtAt}], [\text{roundValue}]) \Rightarrow \text{värde}$

$$\Sigma\text{Int}(1, 3, 12, 4.75, 20000, \dots, 12, 12) = -218.11$$

 ΣInt

$(\text{NPmt1}, \text{NPmt2}, \text{amortTable}) \Rightarrow \text{värde}$

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 158.

- Om du utelämnar Pmt används förinställningen $\text{Pmt}=\text{tvmpmt}(\text{N}, \text{I}, \text{PV}, \text{FV}, \text{PpY}, \text{CpY}, \text{PmtAt})$.
- Om du utelämnar FV används förinställningen $\text{FV}=0$.
- Förinställningarna av PpY , CpY och PmtAt är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

$\text{tbl}:=\text{amortTbl}(12, 12, 4.75, 20000, \dots, 12, 12)$			
0	0.	0.	20000.
1	-79.17	-1630.69	18369.3
2	-72.71	-1637.15	16732.2
3	-66.23	-1643.63	15088.5
4	-59.73	-1650.13	13438.4
5	-53.19	-1656.67	11781.7
6	-46.64	-1663.22	10118.5
7	-40.05	-1669.81	8448.7
8	-33.44	-1676.42	6772.28
9	-26.81	-1683.05	5089.23
10	-20.14	-1689.72	3399.51
11	-13.46	-1696.4	1703.11
12	-6.74	-1703.12	-0.01

$$\Sigma\text{Int}(1, 3, \text{tbl}) = -218.11$$

ΣInt(*NPmt1*,*NPmt2*,*amortTable*)

beräknar räntesumman baserat på amorteringstabellen *amortTable*.

Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 7.

Obs: Se även ΣPrn() nedan och Bal(), på sidan 15.

ΣPrn()

ΣPrn(*NPmt1*, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*])⇒värde

ΣPrn(1,3,12,4.75,20000,,12,12) -4911.47

ΣPrn

(*NPmt1*,*NPmt2*,*amortTable*)⇒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 158.

- 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.

tbl:=amortTbl(12,12,4.75,20000,,12,12)			
0	0.	0.	20000.
1	-79.17	-1630.69	18369.3
2	-72.71	-1637.15	16732.2
3	-66.23	-1643.63	15088.5
4	-59.73	-1650.13	13438.4
5	-53.19	-1656.67	11781.7
6	-46.64	-1663.22	10118.5
7	-40.05	-1669.81	8448.7
8	-33.44	-1676.42	6772.28
9	-26.81	-1683.05	5089.23
10	-20.14	-1689.72	3399.51
11	-13.46	-1696.4	1703.11
12	-6.74	-1703.12	-0.01

ΣPrn(1,3,tbl) -4911.47

Obs: Se även ΣInt() ovan och Bal(), på sidan 15.

(indirection)

  tangenter# *varNameString*

Skapar eller avser variabeln xyz.

Avser variabeln vars namn är *varNameString*. Detta låter dig använda strängar för att skapa variabelnamn inom en funktion.

$10 \rightarrow r$	10
"r" $\rightarrow sI$	"r"
#sI	10

Ger värdet på variabeln (r) vars namn är lagrat i variabeln s1.

E (grundpotensform)

 tangent*mantissa*E*exponent*

23000.	23000.
2300000000.+4.1E15	4.1E15
$3 \cdot 10^4$	30000

Skriver in ett tal i grundpotensform. Talet tolkas som *mantissa* × 10^{exponent}.

Tips: Om du vill skriva in en potens av 10 utan att orsaka ett decimalt resultat, använd 10^{integer}.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva @E. Skriv exempelvis 2.3@E4 för att mata in 2.3E4.

g (nygrad)

 tangent*ListI*g⇒*lista*

I vinkelläge Grader, Nygrader eller Radianer:

*MatrixI*g⇒*matrix*

Denna funktion ger dig ett sätt att specificera en vinkel i nygrader när du är i läge Grader eller Radianer.

Multipliserar i vinkelläget Radianer *Expr1* med $\pi/200$.

Multipliserar i vinkelläget Grader *Expr1* med $g/100$.

g (nygrad)

1 tangent

Ger i läget Nygrader *Expr1* oförändrat.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva @g.

r (radian)

1 tangent

List1^r ⇒ lista

I vinkelläge Grader, Nygrader eller Radianer:

Matrix1^r ⇒ matris

Denna funktion ger dig ett sätt att specificera en vinkel i radianer när du är i läge Grader eller Nygrader.

Multipliserar i vinkelläget Grader argumentet med $180/\pi$.

Ger i vinkelläget Radianer argumentet oförändrat.

Multipliserar i vinkelläget Nygrader argumentet med $200/\pi$.

Tips: Använd *r* om du vill framtvinga radianer i en funktionsdefinition oavsett det inställda läget när funktionen används.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva @r.

° (grader)

1 tangent

List1[°] ⇒ lista

I vinkelläge Grader, Nygrader eller Radianer:

Matrix1[°] ⇒ matris

I vinkelläget Radianer:

Denna funktion ger dig ett sätt att specificera en vinkel i grader när du är i läge Nygrader eller Radianer.

$$\cos\left\{\left\{0, \frac{\pi}{4}, 90^\circ, 30.12^\circ\right\}\right\}$$

$$\{1, 0.707107, 0., 0.864976\}$$

Multipliserar i vinkelläget Radianer argumentet med $\pi/180$.

Ger i vinkelläget Grader argumentet oförändrat.

Multipliserar i vinkelläget Nygrader argumentet med $10/9$.

° (grader)

1 tangent

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva @d.

° , ' , " (grad/minut/sekund)

ctrl tangent

$dd^{\circ}mm'ss.ss'' \Rightarrow$ uttryck

I vinkelläget Grader:

dd Ett positivt eller negativt tal

25°13'17.5"	25.2215
-------------	---------

mm Ett icke negativt tal

25°30'	$\frac{51}{2}$
--------	----------------

$ss.ss$ Ett icke negativt tal

	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)

ctrl tangent

$[Radius, \angle \theta_Angle] \Rightarrow$ vektor

I vinkelläget Radianer och med vektorformatet inställt på:

(polär indata)

rektangulär

$[Radius, \angle \theta_Angle, Z_Coordinate] \Rightarrow$ vektor

(cylindrisk indata)

cylindrisk

$[Radius, \angle \theta_Angle, \angle \theta_Angle] \Rightarrow$ vektor

(sfärisk indata)

sfärisk

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 @<.

∠ (vinkel)

ctrl  tangenter

(Magnitude ∠ Angle) ⇒ complexValue
(polär indata)

I vinkelläget Radianer och i Rektangulärt
komplex format:

Matar in ett komplext värde i ($r∠\theta$) polär
form. Angle tolkas enligt det inställda
vinkelläget.

$$5+3\cdot i - \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107\cdot i$$

_ (understrykningstecken som ett tomt element)

Se "Tomma element" (på sidan 206).

10^()

Katalog > 

$10^{\wedge}(List1) \Rightarrow lista$

Ger 10 upphöjd till argumentets potens.

Ger, för en lista, 10 upphöjd till potensen
för elementen i List1.

$10^{\wedge}(squareMatrix1) \Rightarrow kvadratMatrix$

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.

$$10^{\wedge 6} \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ -2 & 1 & \end{bmatrix} \begin{bmatrix} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{bmatrix}$$

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

^-1(inverterat värde)

Katalog > 

List1 ^-1 ⇒ lista

Ger argumentets inverterade värde.

Ger, för en lista, de inverterade värdena på
elementen i List1.

squareMatrix1 ^-1 ⇒ kvadratMatrix

Ger inversen av squareMatrix1.

squareMatrix1 måste vara en icke singular
kvadratisk matris.

Uttr | *BoolesktUttr1*
[**and***BoolesktUttr2*]...

Uttr | *BoolesktUttr1*
[**or***BoolesktUttr2*]...

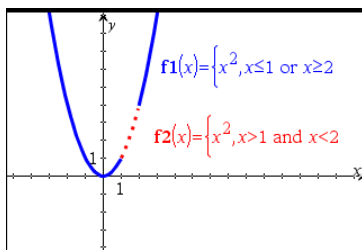
("|")-symbolen begränsning fungerar som en binär operator. Operanden till vänster om | är ett uttryck. Operanden till höger om | specificerar ett eller flera relationer som är avsedda att påverka förenklingen av uttrycket. Flera relationer efter | måste förbindas med ett logiskt "and" eller "or"-operatorer.

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 jämförelseoperatorn "inte lika med" (\neq eller \neq) för att utesluta ett specifikt värde från övervägning.

$$\text{solve}(x^2 - 1 = 0, x) | x \neq 1 \quad x = -1$$

→ (lagra)**ctrl var tangent**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva =: som kortkommando. Skriv exempelvis $\pi/4$ =: **minvar**.

:= (tilldela)**ctrl := tangent***Var := List**Var := Matrix**Function(Param1,...) := Expr**Function(Param1,...) := List**Function(Param1,...) := Matrix***© (kommentar)****ctrl © tangent****© [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**0 B tangent, 0 H tangent****0b** *binaryNumber*

I decimalt basläge:

 $0b10+0hF+10$

27

0h *hexadecimalNumber*

I binärt basläge:

 $0b10+0hF+10$

0b11011

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).

0b, 0h

0 B tangenter, 0 H tangenter

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-handenheter kommer att byta till detta grafikläge samtidigt som de kör grafikkommandon och byter tillbaka till kontexten i vilken programmet körs 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 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 grafiskskärmen

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 Räkna-appen - **disp** och **dispAt** - kommer att stödja kommandon i grafikkontexten. Texten från dessa kommandon kommer att skickas till Räkna-skärmen (inte på Grafik) och kommer att synas efter att programmet avslutas och systemet byter tillbaka till Räkna-appen

Rensa

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

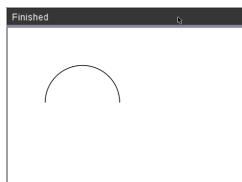
"Bågvinkeln" definierar bågens kurva.

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

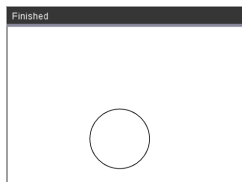
 Katalog > 
 CXII

DrawCircle *x, y, radie*

x, y: koordinat i mittpunkt

radie: cirkelns radie

DrawCircle 150,150,40



Se även: [FillCircle](#)

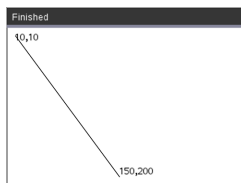
DrawLine $x1, y1, x2, y2$

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

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 $x1, y1, x2, y2, x3, y3...xn, yn$ kommer **INTE** automatiskt att bindas samman till $x1, y1$.

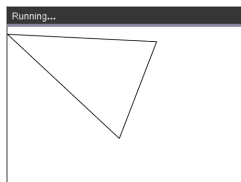
Uttryck som utvärderas till en lista med reella flyttal $xlist, ylist$

Uttryck som utvärderas till ett reellt flyttal $x1, y1...xn, yn$ = koordinater för polygonens hörn

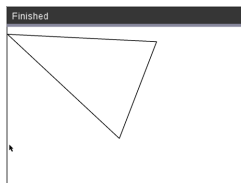
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

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*

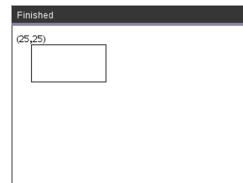
x, y: övre vänstra koordinaten i rektangeln

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

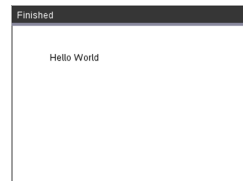
DrawText *x, y, exprOrString1*
[,exprOrString2]...

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"



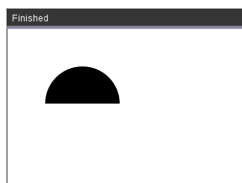
FillArc

 Katalog >  CXII

FillArc x, y , bredd, höjd, startVinkel, bågeVinkel

FillArc 50,50,100,100,0,180

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

FillCircle

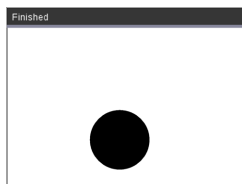
 Katalog >  CXII

FillCircle x, y , radie

FillCircle150,150,40

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.

Här!

FillPoly

 Katalog >  CXII

FillPoly $xlist, ylist$

eller

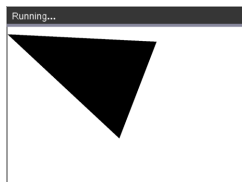
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

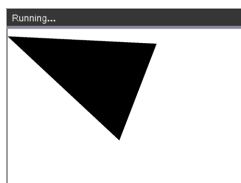
FillPoly $x1, y1, x2, y2, x3, y3...xn, yn$

FillPoly $xlist, ylist$

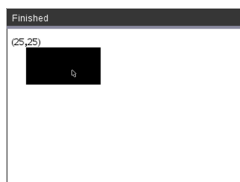
Obs: Linjen och färgen specificeras av [StällInFärg](#) och [StällInPenna](#)



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 höjdRita 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



G

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

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

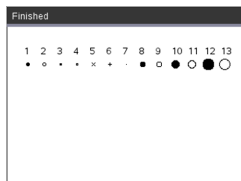
$form$: ett tal mellan 1 och 13 som specificerar form

- 1 - Fylld cirkel
- 2 - Tom cirkel
- 3 - Fylld kvadrat
- 4 - Tom kvadrat
- 5 - Kors
- 6 - Plus
- 7 - Tunn
- 8 - mediumpunkt, solid
- 9 - mediumpunkt, tom
- 10 - större punkt, solid
- 11 - större punkt, tom
- 12 - största punkt, solid
- 13 - största punkt, tom

PlotXY 100,100,1

For $n, 1, 13$ DrawText $1+22*n, 40, n$ PlotXY $5+22*n, 50, n$

EndFor



SetColor
 Katalog > 
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

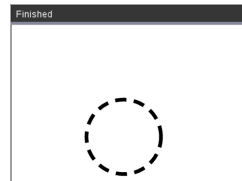
tjocklek: 1 <= tjocklek <= 3 | 1 är tunnast, 3 är tjockast

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

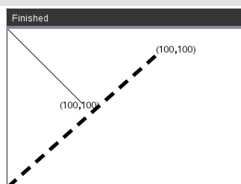
DrawLine 0,0,100,100

Om x_{\min} är större eller lika med x_{\max} eller om y_{\min} är större eller lika med y_{\max} 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

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 37 och `isVoid()`, på sidan 74.

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.

<code> _</code>	–
<code>gcd(100,_)</code>	–
<code>3+_</code>	–
<code>{5,_,10}</code>	<code>{3,6,9}</code> <code>{2,_,1}</code>

Lista argument som innehåller tomma element

Följande funktioner och kommandon ignorerar (hoppas över) tomma element som påträffas i listargument:

count, **countif**, **cumulativeSum**, **freqTable** → lista, **frekvens**, **max**, **medelvärde**, **median**, **produkt**, **stDevPop**, **stDevSamp**, **summa**, **sumif**, **varPop** och **varSamp** samt regressionsberäkningar, **OneVar**, **TwoVar** och **FiveNumSummary** statistik, konfidensintervaller och stat-tester

<code>sum({2,_,3,5,6,6})</code>	16.6
<code>median({1,2,_,_,3})</code>	2
<code>cumulativeSum({1,2,_,4,5})</code>	<code>{1,3,_,7,12}</code>
<code>cumulativeSum</code> $\begin{pmatrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{pmatrix}$	$\begin{bmatrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{bmatrix}$

Lista argument som innehåller tomma element

SortA och **SortD** flyttar alla tomma element inom det första argumentet till botten.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
SortA list1,list2	Done
list1	$\{1,3,4,5,_\}$
list2	$\{1,3,4,5,2\}$

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD list1,list2	Done
list1	$\{5,3,2,1,_\}$
list2	$\{5,3,2,1,4\}$

I regressionsberäkningar introducerar ett tomrum i en X- eller Y-lista ett tomrum i motsvarande element för residualen.

$l1:=\{1,2,3,4,5\}; l2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx l1,l2	Done
stat.Resid	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
stat.XReg	$\{1,_,3,4,5\}$
stat.YReg	$\{2,_,3,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1,1\}$

En utelämnad kategori i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
cat:={"M","M","F","F"}; incl:={"F"}	$\{ "F" \}$
LinRegMx l1,l2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1,1\}$

En frekvens på 0 i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx l1,l2,{1,0,1,1}	Done
stat.Resid	$\{0.069231,_, -0.276923, 0.207692\}$
stat.XReg	$\{1,_,4,5\}$
stat.YReg	$\{2,_,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1\}$

Kortkommandon för att mata in matematiska uttryck

Med kortkommandon kan du mata in element i matematiska uttryck genom att skriva i stället för att använda Katalogen eller Symbolpaletten. För att exempelvis mata in uttrycket $\sqrt{6}$ kan du skriva `sqrt(6)` på inmatningsraden. När du trycker på `enter` ändras uttrycket `sqrt(6)` till $\sqrt{6}$. Vissa kortkommandon kan användas från både handenheten och datorns tangentbord. Andra är i första hand användbara från datorns tangentbord.

Från handenheten eller datorns tangentbord

För att mata in detta:	Skriv detta kortkommando:
π	pi
θ	theta
∞	infinity
\leq	<=
\geq	>=
\neq	/=
\Rightarrow (logisk implikation)	=>
\Leftrightarrow (logisk dubbel implikation, XNOR)	<=>
\rightarrow (lagra operator)	=:
$ $ (absolutbelopp)	abs (...)
$\sqrt{()}$	sqrt (...)
$\Sigma()$ (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:
° (grader)	@d
g (nygrader)	@g
∠ (vinkel)	@<
► (omvandling)	@>
►Decimal, ►approxFraction (), och så vidare.	@>Decimal, @>approxFraction(), och så vidare.

EOS™-hierarki (Equation Operating System)

Detta avsnitt beskriver det ekvationsoperativsystem (EOS™) som används av TI-Nspire™ 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 operatören (#) 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 $4^4 3!$ utvärderas $3!$ först. Resultatet, 6, blir sedan exponenten för 4, vilket ger 4096.

Exponentberäkning

Exponentberäkning (^) och exponentberäkning element för element (.^) utvärderas från höger till vänster. Till exempel utvärderas uttrycket 2^3^2 på samma sätt som $2^{(3^2)}$. Man får resultatet 512. Detta är inte detsamma som $(2^3)^2$, vilket ger resultatet 64.

Negation

För att skriva in ett negativt tal, tryck på $\boxed{-}$ 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 ("|")-operatören begränsning ger en uppsättning av begränsningar som påverkar utvärderingen av argumentet som föregår operatören.

TI-Nspire CX II - TI-Basic programmeringsfunktioner

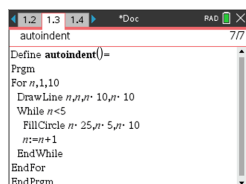
Autoindentering i Programeditor

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.



```
autoindent 77
Define autoindent()=
Prgm
For n,1,10
DrawLine n,n,n-10,n-10
While n<5
FillCircle n-25,n-5,n-10
n:=n+1
EndWhile
EndFor
EndPrgm
```

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

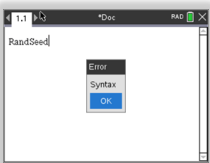
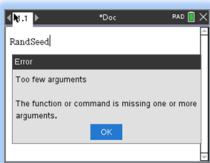
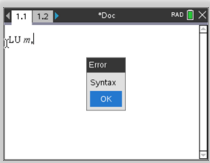
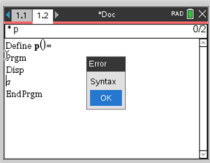
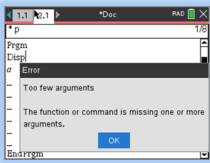
Fel

Feltillstånd	Nytt meddelande
Fel i villkorsats (If/While)	En villkorsats 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 endsats
Saknar EndFor	Förväntade EndFor men hittade en annan endsats
Saknar EndWhile	Förväntade EndWhile men hittade en annan endsats
Saknar EndLoop	Förväntade EndLoop men hittade en annan endsats

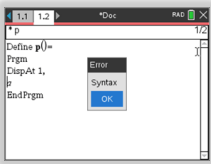
Feltillstånd	Nytt meddelande
Saknar EndTry	Förväntade EndTry men hittade en annan endsats
" Then " utelämnad efter If <condition>	Saknar If...Then
" Then " utelämnad efter Elseif <condition>	Then saknas i block: Elseif .
När " Then ", " Else " och " Elseif " påträffades utanför kontrollblock	Else ogiltigt 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

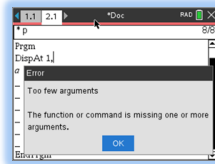
Om kommandon som förväntar sig ett eller flera argument anropas med en ofullständig lista över argument så kommer ett "**För få argument-fel**" att utfärdas istället för ett "**syntax**"-fel

Aktuellt uppträdande	Nytt CX II-uppträdande
 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.1'. The main display area contains the text 'RasidSeed'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Syntax'. There are 'Syntax' and 'OK' buttons in the dialog.</p>	 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.1'. The main display area contains the text 'RasidSeed'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Too few arguments. The function or command is missing one or more arguments.' There is an 'OK' button in the dialog.</p>
 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.2'. The main display area contains the text '[L1 m]'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Syntax'. There are 'Syntax' and 'OK' buttons in the dialog.</p>	 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.2'. The main display area contains the text '[L1 m]'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Too few arguments. The function or command is missing one or more arguments.' There is an 'OK' button in the dialog.</p>
 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.2'. The main display area contains the text '* p', 'Deltase p()=', 'Prgm', 'Disp', 'p', and 'EndPrgm'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Syntax'. There are 'Syntax' and 'OK' buttons in the dialog.</p>	 <p>A screenshot of the TI-84 Plus CE II calculator interface. The top status bar shows '1.1' and '1.2'. The main display area contains the text '* p', 'Prgm', 'Disp', and 'EndPrgm'. An error dialog box is displayed in the center, with the title 'Error' and the message 'Too few arguments. The function or command is missing one or more arguments.' There is an 'OK' button in the dialog.</p>

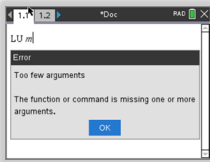
Aktuellt uppträdande




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	8987551792.261 _m/_F
_Fc	Faradays konstant	96485.33212 _coul/_mol
_g	Tyngdkraftens acceleration	9.80665 _m/_s ²
_Gc	Gravitationskonstanten	6.6743E-11 _m ³ /_kg/_s ²
_h	Plancks konstant	6.62607015E-34 _J _s
_k	Boltzmanns konstant	1.380649E-23 _J/_°K
_μ0	Permeabilitet vid vakuum	1.25663706212E-6 _N/_A ²
_μb	Bohrmagneton	9.274009994E-24 _J _m ² /_Wb
_Me	Elektronens vilomassa	9.1093837015E-31 _kg
_Mμ	Myonmassa	1.883531627E-28 _kg
_Mn	Neutronens vilomassa	1.67492749804E-27 _kg
_Mp	Protonens vilomassa	1.67262192369E-27 _kg
_Na	Avogadros tal	6.02214076E23 _/_mol
_q	Elektronens laddning	1.602176634E-19 _coul
_Rb	Bohrradie	5.29177210903E-11 _m
_Rc	Molar gas constant (Allmänna gaskonstanten)	8.314462618 _J/_mol/_°K
_Rdb	Rydbergs konstant	10973731.568160/_m
_Re	Elektronradie	2.8179403262E-15 _m
_u	Atommassa	1.6605390666E-27 _kg
_Vm	Molvolyum	2.241396954E-2 _m ³ /_mol
_ε0	Permittivitet vid vakuum	8.8541878128E-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 *errorCode*. Användardefinierade program och funktioner kan undersöka *errorCode* för att bestämma orsaken till ett fel. För ett exempel på användningen av *errorCode*, se exempel 2 under kommandot **Try**, på sidan 154.

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: <ul style="list-style-type: none">• inte börjar med en siffra• inte innehåller mellanslag eller specialtecken• inte innehåller understrykningstecken eller punkt på ogiltigt sätt• inte överskrider max. längd Se avsnittet Calculator (Räknare) i dokumentationen för mer information.
160	Argument must be an expression (Argumentet måste vara ett uttryck)
165	Batteries too low for sending or receiving (Batterierna är för svaga för sändning eller mottagning) Sätt i nya batterier före sändning eller mottagning.
170	Bound (Gräns)

Felkod	Beskrivning
	Den nedre gränsen måste vara mindre än den övre gränsen för att definiera sökintervallet.
180	Avsluta Tangenten <code>esc</code> eller <code>ctrl on</code> 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)

Felkod	Beskrivning
295	Excessive iteration (För många iterationer)
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.

Felkod	Beskrivning
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.
565	Invalid outside program (Ogiltigt 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 (Resursutmattnig)
673	Resource exhaustion (Resursutmattnig)
680	Missing ((Saknar " ("
690	Missing) (Saknar ") "
700	Missing " (Saknar " " "

Felkod	Beskrivning
710	Missing] (Saknar "] ")
720	Missing } (Saknar " } ")
730	Missing start or end of block syntax (Saknar start eller slut i blockets syntax)
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 slumpstyp tillåts ej vid grafritning)
860	Recursion too deep (Rekursionen för djup)
870	Reserved name or system variable (Reserverat namn eller systemvariabel)
900	Argument error (Argumentfel) Median-median-modellen kunde inte användas på datauppsättningen.
910	Syntaxfel
920	Text not found (Text hittades inte)
930	Too few arguments (För få argument) Funktionen eller kommandot saknar ett eller flera argument.
940	Too many arguments (För många argument) Uttrycket eller ekvationen innehåller för många argument och kan inte utvärderas.
950	Too many subscripts (För många nedsänkta tecken)
955	Too many undefined variables (För många odefinierade variabler)
960	Variable is not defined (Variabel ej definierad)

Felkod	Beskrivning
	<p>Inget värde är tillskrivet variabeln. Använd något av följande kommandon:</p> <ul style="list-style-type: none"> • sto → • := • Define (Definiera) <p>för att tilldela variabler värden.</p>
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	<p>Invalid variable name (Ogiltigt variabelnamn)</p> <p>Kontrollera att namnet inte överskrider max. längd.</p>
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. (Operatören 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	<p>Input argument must be numeric. (Inmatade argument måste vara numeriska.)</p> <p>Endast inmatningar som innehåller numeriska värden är tillåtna.</p>
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	<p>Function is not defined. (Funktion ej definierad.) Använd något av följande kommandon:</p> <ul style="list-style-type: none"> • Define (Definiera) • := • sto → <p>för att definiera en funktion.</p>
1100	Non-real calculation (Icke-reell beräkning)

Felkod	Beskrivning
	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)
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: <ul style="list-style-type: none"> • 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) <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • Kontrollera att biblioteket är i MyLib-mappen. • Uppdatera bibliotek. Se avsnittet Bibliotek i dokumentationen för mer information.

Felkod	Beskrivning
1200	<p>Biblioteksvariabel kunde ej hittas:</p> <ul style="list-style-type: none"> • Kontrollera att biblioteksvariabeln finns i det första problemet i biblioteket. • Kontrollera att biblioteksvariabeln har definierats som LibPub eller LibPriv. • Uppdatera bibliotek. <p>Se avsnittet Bibliotek i dokumentationen för mer information.</p>
1210	<p>Ogiltigt genvägsnamn till bibliotek.</p> <p>Kontrollera att namnet inte:</p> <ul style="list-style-type: none"> • innehåller en punkt • börjar med ett understrykningstecken • överskrider 16 tecken • är ett reserverat namn <p>Se avsnittet Bibliotek i dokumentationen för mer information.</p>
1220	<p>Områdesfel:</p> <p>Funktionerna tangentLine och normalLine stöder endast funktioner med reella värden.</p>
1230	<p>Områdesfel.</p> <p>Trigonometriska omvandlingsoperatorer stöds inte i vinkellägena Grader och Nygrader.</p>
1250	<p>Argumentfel</p> <p>Använd ett linjärt ekvationssystem.</p> <p>Exempel på ett linjärt ekvationssystem med variablerna x och y:</p> $3x+7y=5$ $2y-5x=-1$
1260	<p>Argumentfel:</p> <p>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.</p>
1270	<p>Argumentfel</p> <p>Derivatans ordning måste vara lika med 1 eller 2.</p>
1280	<p>Argumentfel</p> <p>Använd ett polynom i expanderad form i en variabel.</p>
1290	<p>Argumentfel</p>

Felkod	Beskrivning
	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: 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 163.

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(): <ul style="list-style-type: none">• 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.

Varningskod	Meddelande
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
10016	1^{undef} ersätts med 1
10017	För stort värde ersätts 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 garantins längd och villkor eller om produkttjänsten.

Begränsad garanti. Denna garanti påverkar inte dina lagstadgade rättigheter.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Index

'		
' , minutnotation	186	
-		
- , subtrahera[*]	172	
!		
! , faktulet	180	
"		
" , sekundnotation	186	
#		
# , indirection	184	
# , indirection, operator	211	
%		
% , procent	176	
&		
& , lägg till	180	
*		
* , multiplicera	173	
.		
.- , punkt subtraktion	175	
. * , punkt multiplikation	175	
./ , punkt division	175	
.^ , punkt potens	175	
+. , punkt addition	174	
:		
:= , tilldela	189	
^		
^-1 , inverterat värde	187	
^ , potens	173	
, operatör begränsning	188	
+		
+ , addera	172	
/		
/ , dividera[*]	173	
=		
≠ , inte lika med[*]	177	
= , lika med	176	
>		
> , större än	178	
∏		
∏ , produkt[*]	181	
∑		
∑ () , summa[*]	181	
∑Int ()	182	
∑Prn ()	183	
√		
√ , kvadratroten[*]	180	
≤		
≤ , mindre än eller lika med	177	
≥		
≥ , större än eller lika med	178	
▶		
▶ , konvertera till nygradvinkel[Grad]	66	
▶approxFraction ()	13	
▶Base10, visa som decimalt heltal		
[Bas 10]	17	
▶Base16, visa som hexadecimal[Bas		
16]	18	
▶Base2, visa som binär[Bas 2]	16	

►Cylind, visa som cylindrisk vektor [Cylind]	33	addera, +	172
►DD, visa som decimal vinkel[DD] ..	34	amorteringstabell, amortTbl()	7, 15
►Decimal, visa resultat som decimal [Decimal]	34	amortTbl(), amorteringstabell	7, 15
►DMS, visa som grad/minut/sekund [DMS]	41	and, Booleska operatörer	8
►Polar, visa som polär vektor[Polär]	109	andradderivata mall för	5
►Rad, omvandla till radianer	117	angle(), vinkel	9
►Rect, visa som ortogonal vektor ...	120	annars, Else	67
►Sphere, visa som sfärisk vektor [sfär]	142	ANOVA 2-vägs, 2-vägs variansanalys	10
		ANOVA, 1-vägs variansanalys	9
		Ans, sista svar	12
		antal dagar mellan datum, dbd() ...	33
		approx(), ungefärlig	12
		approxRational()	13
		arccos()	13
		arccosh()	13
		arccosinus, $\cos^{-1}()$	26
		arccot()	13
		arccoth()	13
		arccsc()	13
		arcscch()	14
		arcsec()	14
		arcsech()	14
		arcsin()	14
		arcsinh()	14
		arcsinus, $\sin^{-1}()$	139
		arctan()	14
		arctangens, $\tan^{-1}()$	149
		arctanh()	14
		argument i TVM-funktioner	158
		augment(), sammanfoga/slå ihop ..	14
		avgRC(), genomsnittlig ändringsfrekvens	15
		avrunda, round()	129
		avsluta om, EndIf	67
		avsluta om, EndIf	67
		avsluta, Exit	47
		B	
		Begär	123
		beräkning, ordning vid	210
		bestämd integral mall	6
		bibliotek skapa genvägar till objekt	76
		binär indikator, Ob	189
→, lagra	189		
⇒, logisk implikation[*]	179, 208		
⇔, logisk dubbel implikation[*]	179		
©			
©, kommentar	189		
°			
°, grader/minuter/sekunder[*]	186		
°, gradnotation[*]	185		
0			
Ob, binär indikator	189		
Oh, hexadecimal indikator	189		
1			
10^(), tiopotens	187		
2			
2-sampel F Test	55		
A			
abs(), absolutbelopp	7		
absolutbelopp mall	3-4		

visa, ►Base2	16	csch ⁻¹ (), invers hyperbolisk cosekant	31
binomCdf()	18, 72	csch(), hyperbolisk cosekant	31
binomPdf()	19	CubicReg, kubisk regression	31
blandade bråk, använda propFrac() med	113	cumulativeSum(), kumulativ summa	32
Booleska operatorer		Cycle, cykel	32
⇒	179, 208	cykel, Cycle	32
⇔	179		
and	8		
eller	106		
icke	102		
negerad	97		
nor	101		
xor	164		
bråk			
mall	1		
propFrac	113		
		D	
		d(), förstaderivata	180
		days between dates (dagar mellan datum), dbd()	33
		dbd(), days between dates (dagar mellan datum)	33
		decimal	
		heltalsvisning, ►Base10	17
		vinkelvisning, ►DD	34
		Define	34
		Define LibPriv	36
		Define LibPub	36
		define, Define	34
		Define, define	34
		defining	
		private function or program	36
		public function or program	36
		dela heltal, intDiv()	70
		deltmatrix, subMat()	148
		deltaList()	37
		DelVar, ta bort variabel	37
		delVoid(), ta bort tomma element derivata	37
		förstaderivata, d()	180
		numerisk derivata, nDeriv()	99
		numerisk derivata, nDerivative())	98
		det(), matrisdeterminant	38
		diag(), diagonal matris	38
		dim(), dimension	39
		dimension, dim()	39
		Disp, visa data	39, 132
		DispAt	39
		dividera, /	173
		dotP(), skalärprodukt	41
		E	
		e exponent	
		mall	2
		e till en potens, e^()	42, 47
		E, exponent	184

e [^] (), e till en potens	42	finansiella funktioner, tvnN()	157
eff), konvertera nominell till effektiv ränta	42	finansiella funktioner, tvnPmt() ...	157
effektiv ränta, eff()	42	finansiella funktioner, tvnPV()	158
egentligt bråk, propFrac	113	FiveNumSummary	51
egenvärde, eigVl()	43	floor(), golv	52
egenvektor, eigVc()	43	flytta fel, PassErr	108
eigVc(), egenvektor	43	For	52
eigVl(), egenvärde	43	for, For	52
Ekvationsoperativsystem (EOS)	210	For, for	52
ekvationsystem (2 ekvationer) mall	3	fördelningsfunktioner	
ekvationsystem (N ekvationer) mall	3	binomCdf()	18, 72
eller (boolesk), eller	106	binomPdf()	19
eller, Boolesk operator	106	invNorm()	72
else if, Elseif	44	invt()	73
Elseif, else if	44	Invx ² ()	71
end		normCdf()	102
for, EndFor	52	normPdf()	102
funktion, EndFunc	56	poissCdf()	109
end function, EndFunc	56	poissPdf()	109
EndTry, slut försök	154	tCdf()	151
EndWhile, slut medan	164	tPdf()	153
enhetsvektor, unitV()	160	χ ² 2way()	20
envariabelstatistik, OneVar	105	χ ² Cdf()	21
EOS (Ekvationsoperativsystem)	210	χ ² GOF()	21
euler(), Euler function	45	χ ² Pdf()	21
Exit, avsluta	47	format(), formatsträng	53
exp(), e till en potens	47	formatsträng, format()	53
exponent, E	184	försök, Try	154
exponenter		förstaderivata	
mall	1	mall för	5
exponentiell regression, ExpReg	48	fpart(), funktionsdel	53
expr(), sträng till uttryck	48	freqTable()	54
ExpReg, exponentiell regression	48	frequency()	54
		Frobenius norm, norm()	102
		Func, funktion	56
		Func, programfunktion	56
		functions	
		user-defined	34
		funktioner	
		del, fpart()	53
		programfunktion, Func	56
		funktioner och variabler	
		kopiera	24
		fyll	198-199
		G	
		g, nygrader	184
		gå till, Goto	66
		gcd(), största gemensamma delare	57
F			
factor(), faktor	49		
faktor, factor()	49		
fakultet, !	180		
fel och felsökning			
flytta fel, PassErr	108		
rensa fel, ClrErr	22		
Felkoder och meddelanden	225		
Fill, fylla matris	50		
finansiella funktioner, tvnFV()	157		
finansiella funktioner, tvnI()	157		

genomsnittlig ändringsfrekvens, avgRC()	15		
geomCdf()	57		
geomPdf()	57		
Get	58, 200		
getDenom(), hämta/ge nämnare	59		
getKey()	59		
getLangInfo(), hämta/ge språkinformation	62		
getLockInfo(), testar låsstatus hos en variabel eller variabelgrupp	63		
getMode(), hämta lägesinställningar	63		
getNum(), hämta/ge tal	64		
GetStr	64		
getType(), get type of variable	65		
getVarInfo(), hämta/ge variabelinformation	65		
golv, floor()	52		
Goto, gå till	66		
grader/minuter/sekunder	186		
gradnotation, °	185		
grupper, låsa och låsa upp	85, 161		
grupper, testa låsstatus	63		
H			
hämta/ge			
nämnare, getDenom()	59		
tal, getNum()	64		
variabelinformation, getVarInfo()	62, 65		
heltal, int()	70		
heltalsdel, iPart()	73		
hexadecimal			
indikator, Oh	189		
visa, ►Base16	18		
höger, right()	126		
höger, right()	70		
hyperbolisk			
arccosinus, cosh ⁻¹ ()	27		
arcsinus, sinh ⁻¹ ()	140		
arctangens, tanh ⁻¹ ()	150		
cosinus, cosh()	26		
sinus, sinh()	139		
tangens, tanh()	150		
			I
icke, Booleska operatörer	102		
identitetsmatris, identity()	67		
identity(), identitetsmatris	67		
if, om	67		
ifFn()	68		
imag(), imaginärdel	69		
imaginärdel, imag()	69		
indirection, #	184		
indirection, operator (#)	211		
inom sträng, inString()	69		
inställningar, hämta aktuella	63		
inString(), inom sträng	69		
int(), heltal	70		
intDiv(), dela heltal	70		
inte lika med, ≠	177		
interpolate(), interpolera	70		
invers kumulativ normalfördelning, invNorm()	72		
invers, ⁻¹	187		
inverterat värde, ⁻¹	187		
invF()	71		
invNorm(), invers kumulativ normalfördelning	72		
invt()	73		
Invχ ² ()	71		
iPart(), heltalsdel	73		
irr(), internränta			
internal rate of return, irr()	73		
isPrime(), primtest	74		
isVoid(), testa om sant	74		
			K
kombinationer, nCr()	98		
Kommandot Wait	162		
kommentar, ©	189		
komplex			
konjugat, conj()	23		
konstruera matris, constructMat()	23		
konvertera			
4Grad	66		
kopiera variabel eller funktion, CopyVar	24		
korrelationsmatris, corrMat()	25		
kortkommandon	208		
kortkommandon, tangentbord	208		
kubisk regression, CubicReg	31		

kumulativ summa, cumulativeSum()	32	ny, newList()	99
kvadratisk regression, QuadReg	114	produkt, product()	112
kvadratrot		sammanfoga/slå ihop, augment()	14
mall	1	skalärprodukt, dotP()	41
kvadratrot, √()	143, 180	skillnad, @list()	82
kvartär regression, QuartReg	115	skillnader i en lista, @list()	82
L			
lägen		sortera fallande, SortD	142
inställning, setMode()	134	sortera stigande, SortA	142
lägesinställningar, getMode()	63	summering, sum()	147
lägg till, &	180	vektorprodukt, crossP()	30
lagra		ln(), naturlig logaritm	82
symbol, &	189	LnReg, logaritmisk regression	83
längd på sträng	39	Local, lokal variabel	84
läsa upp variabler eller		Log	
variabelgrupper	161	mall	2
läsa variabler eller variabelgrupper	85	logaritmer	82
Låsa, låsa variabel eller		logaritmisk regression, LnReg	83
variabelgrupp	85	logisk dubbel implikation, ⇔	179
Lbl, märka	75	logisk implikation, ⇒	179, 208
lcm, minsta gemensamma multipel	75	Logistic, logistisk regression	86
left(), vänster	75	LogisticD, logistisk regression	86
LibPriv	36	logistisk regression, Logistic	86
LibPub	36	logistisk regression, LogisticD	86
libShortcut(), skapa genvägar till		lokal variabel, Local	84
biblioteksobjekt	76	lokal, Local	84
lika med, =	176	Loop, slinga	88
linjär regression, LinRegAx	77	LU, matris undre-övre uppdelning	88
linjär regression, LinRegBx	76, 78	M	
LinRegBx, linjär regression	76	mallar	
LinRegMx, linjär regression	77	absolutbelopp	3-4
LinRegtIntervals, linjär regression	78	andraderivata	5
LinRegtTest	80	bestämd integral	6
linSolve()	81	bråk	1
list►mat(), lista till matris	82	e exponent	2
lista till matris, list►mat()	82	ekvationsssystem (2 ekvationer)	3
lista, räkna poster	28	ekvationsssystem (N ekvationer)	3
lista, villkorligt räkna poster	29	exponent	1
listor		förstaderivata	5
kumulativ summa,		kvadratrot	1
cumulativeSum()	32	Log	2
lista till matris, list►mat()	82	matris (1 × 2)	4
matris till lista, mat►list()	89	matris (2 × 1)	4
maximum, max()	89	matris (2 × 2)	4
med tomta element	206	matris (m × n)	4
minimum, min()	92	n:te rot	1
mittsträng, mid()	92	produkt (P)	5

stegvis funktion (2 steg)	2	slump, randMat()	119
stegvis funktion (N steg)	2	summering, sum()	147
summa (G)	5	transponera, T	149
märka, Lbl	75	trappstegsform, ref()	121
mat►list(), matris till lista	89	undermatris, subMat()	147
matris (1 × 2)		undre-övre uppdelning, LU	88
mall	4	växla matrisrad, rowSwap()	130
matris (2 × 1)		max(), maximum	89
mall	4	maximum, max()	89
matris (2 × 2)		mean(), medelvärde	90
mall	4	med,	188
matris (m × n)		medan, While	164
mall	4	medelvärde, mean()	90
matris till lista, mat►list()	89	median(), median	90
matriser		median, median()	90
delmatris, subMat()	148	medium-medium-linjereggression,	
determinant, det()	38	MedMed	91
diagonal, diag()	38	MedMed, medium-medium-	
dimension, dim()	39	linjereggression	91
egenvärde, eigVl()	43	mid(), mittsträng	92
egenvektor, eigVc()	43	min(), minimum	92
fylla, Fill	50	mindre än eller lika med, {	177
identitet, identity()	67	minimum, min()	92
kolumndimension, colDim() ...	23	minsta gemensamma multipel, lcm .	75
kolumnnorm, colNorm()	23	minutnotation,	186
kumulativ summa,		mirr(), modifierad internränta	93
cumulativeSum()	32	mittsträng, mid()	92
lista till matris, list►mat()	82	mod(), modul	94
matris till lista, mat►list()	89	modifierad internränta, mirr()	93
maximum, max()	89	modul, mod()	94
minimum, min()	92	mRow(), matrisradoperation	94
ny, newMat()	99	mRowAdd(),	
produkt, product()	112	matrisradmultiplikation och	
punkt addition, .+	174	addition	94
punkt division, .P	175	Multipelt linjärt regression-t-test ...	96
punkt multiplikation, .*	175	Multiplitera, *	173
punkt potens, .^	175	MultReg	94
punkt subtration, .N	175	MultRegIntervals()	95
QR-faktorisering, QR	113	MultRegTests()	96
radaddition, rowAdd()	129		
raddimension, rowDim()	130	N	
radmultiplikation och addition,		n:te rot	
mRowAdd()	94	mall	1
radnorm rowNorm()	130	nand, Boolesk operator	97
radoperation, mRow()	94	när, when()	163
reducerad radtrappstegsform,		naturlig logaritm, ln()	82
rref()	130	nCr(), kombinationer	98
sammanfoga/slå ihop, augment(nDerivative(), numerisk derivata ...	98
)	14		

negation, skriva in negativa tal	211	\blacktriangleright Ry(), rektangulär y-koordinat	108
nettovärde, npv()	103	PassErr, flytta fel	108
newList(), ny lista	99	Pdf()	54
newMat(), ny matris	99	permutationer, nPr()	103
nfMax(), numeriskt funktionsmaximum	99	piecewise()	109
nfMin(), numeriskt funktionsminimum	99	poissCdf()	109
nInt(), numerisk integral	100	poissPdf()	109
nom), konvertera effektiv till nominell ränta	100	polär visa vektor, \blacktriangleright Polar	109
nominell ränta, nom()	100	polära koordinater, R \blacktriangleright θ ()	116-117
nor, Booleska operatörer	101	polyEval(), utvärdera polynom	110
norm(), Frobenius norm	102	polynom slump, randPoly()	119
normCdf()	102	utvärdera, polyEval()	110
normPdf()	102	PolyRoots()	110
nPr(), permutationer	103	potens, ^	173
npv(), nettovärde	103	potensregression, PowerReg ...	110-111, 123, 125, 151
nSolve(), numerisk lösning	104	PowerReg, potensregression	111
numeric derivata, nDeriv()	99	Prgm, definiera program	112
numerisk derivata, nDeriv()	99	primtalstest, isPrime()	74
derivata, nDerivative()	98	procent, %	176
integral, nInt()	100	prodSeq()	112
lösning, nSolve()	104	product(), produkt	112
ny lista, newList()	99	produkt (P) mall	5
matris, newMat()	99	produkt, product()	112
nygradsnotation, g	184	produkt, Π ()	181
O			
objekt skapa genvägar till bibliotek	76	program och programmering försök, Try	154
om, If	67	rensa fel, ClrErr	22
omvandla \blacktriangleright Rad	117	slut försök, EndTry	154
OneVar, envariabelstatistik	105	slut program, EndPrgm	112
operatörer ordning vid beräkning	210	visa I/O- skärm, Disp	132
operatören begränsning " "	188	visa I/O-fönster, Disp	39
operatören begränsning, ordning vid utvärdering	210	programmera definiera program, Prgm	112
ord(), numerisk teckenkod	107	flytta fel, PassErr	108
ortogonalvektorvisning, \blacktriangleright Rect	120	visa data, Disp	39
P			
\blacktriangleright Rx(), rektangulär x-koordinat	107	programmering visa data, Disp	132
		programs defining private library	36
		defining public library	36
		propFrac, egentligt bråk	113
		punkt addition, .+	174
		division, .P	175

multiplikation, *	175	sinus, SinReg	140
potens, ^	175	rektangulär x-koordinat, P►Rx()	107
subtraktion, .N	175	rektangulär y-koordinat, P►Ry()	108
Q			
QR-faktorisering, QR	113	remain(), rest	123
QR, QR-faktorisering	113	rensa	
QuadReg, kvadratisk regression	114	fel, ClrErr	22
QuartReg, kvartär regression	115	Rensa	194
R			
R, radian	185	RequestStr	125
R►Pr(), polära koordinater	117	rest, remain()	123
R►Pθ(), polära koordinater	116	resultat, statistik	143
radian, R	185	resultvärden, statistik	145
räkna poster i en lista villkorligt ,		retur, Return	126
countif()	29	Return, retur	126
räkna poster i en lista, count()	28	right(), höger	126
rand(), slumpstal	117	right, right()	45, 163
randBin, slumpstal	118	rita	195-197
randInt(), slumpsheltal	118	rk23(), Runge Kutta-funktion	126
randMat(), slumpmatris	119	rotate(), rotera	128
randNorm(), slumppnorm	119	rotera, rotate()	128
randPoly(), slumppolynom	119	round(), avrunda	129
randSamp()	119	rowAdd(), matrisradaddition	129
RandSeed, slumpstalsfrö	120	rowDim(), matrisraddimension	130
real(), reell	120	rowNorm(), matrisradnorm	130
reducerad radtrappstegsform, rref()	130	rowSwap(), växla matrisrad	130
reell, real()	120	rref(), reducerad trappstegsform	130
ref(), trappstegsform	121	S	
RefreshProbeVars	122	sammanfoga/slå ihop, augment()	14
regression		samtidiga ekvationer, simult()	137
exponentiell, ExpReg	48	sannolik täthet, normPdf()	102
kubisk, CubicReg	31	sannolikhet för normalfördelning,	
regressioner		normCdf()	102
kvadratisk, QuadReg	114	sannolikhet för student-t-fördelning,	
kvartär, QuartReg	115	tCdf()	151
linjär regression, LinRegAx	77	sec ⁻¹ (), invers sekansfunktion	131
linjär regression, LinRegBx	76, 78	sec(), sekansfunktion	131
logaritmisk, LnReg	83	sech ⁻¹ (), invers hyperbolisk	
Logistic	86	sekansfunktion	131
logistisk, Logistic	86	sech(), hyperbolisk sekansfunktion	131
medium-medium-linje,		sekundnotation, " "	186
MedMed	91	seq(), talföljd	132
MultReg	94	seqGen()	133
potensregression,		seqn()	134
PowerRe		sequence, seq()	133-134
g	110-111, 123, 125, 151	setMode(), ställ in läge	134
		sfärisk vektorvisning, ►Sphere	142
		shift(), skifta	136
		sign(), tecken	137

simult(), samtidiga ekvationer	137	varians, variance()	161
sin ⁻¹ (), arcsinus	139	stdDevPop(), standardavvikelse för population	145
sin(), sinus	138	stdDevSamp(), standardavvikelse för urval	145
sinh ⁻¹ (), hyperbolisk arcsinus	140	stegvis funktion (2 steg) mall	2
sinh(), hyperbolisk sinus	139	stegvis funktion (N steg) mall	2
SinReg, sinus regression	140	Stoppkommando	146
sinus, sin()	138	större än eller lika med, 	178
sinusregression, SinReg	140	större än, >	178
skalär		största gemensamma delare, gcd()	57
produkt, dotP()	41	sträng	
skifta, shift()	136	dimension, dim()	39
slinga, Loop	88	längd	39
slump		strängar	
matris, randMat()	119	använda för att skapa variabelnamn	211
norm, randNorm()	119	format, format()	53
polynom, randPoly()	119	formatera	53
slumpfrö, RandSeed	120	höger, right()	70, 126
slump sampel	119	indirection, #	184
slut		inom, inString	69
försök, EndTry	154	lägg till, &	180
medan, EndWhile	164	mittsträng, mid()	92
program, EndPrgm	112	numerisk teckenkod, ord()	107
slinga, EndLoop	88	rotera, rotate()	128
slut medan, EndWhile	164	skifta, shift()	136
slut slinga, EndLoop	88	sträng till uttryck, expr()	48
SortA, sortera stigande	142	teckensträng, char()	20
SortD, sortera fallande	142	uttryck till sträng, string()	146
sortera		vänster, left()	75
fallande, SortD	142	string(), uttryck till sträng	146
stigande, SortA	142	strings	
språk		right, right()	45, 163
hämta språkinformation	62	subMat(), delmatrix	148
sqrt(), kvadratroten	143	subMat(), undermatrix	147
ställ in		substitution med " " -operatorn	188
läge, setMode()	134	subtrahera, -	172
standardavvikelse, stdDev()	145, 161	sum(), summering	147
stat.results	143	sumlf()	147
stat.values	145	summa (G) mall	5
statistik		summa av kapitalbetalningar	183
envariabelstatistik, OneVar	105	summa av räntebetalningar	182
fakultet, !	180	summa, Σ()	181
kombinationer, nCr()	98	summering, sum()	147
medelvärde, mean()	90	sumSeq()	148
median, median()	90		
permutationer, nPr()	103		
slump norm, randNorm()	119		
slumptalsfrö, RandSeed	120		
standardavvikelse, stdDev()	145, 161		
tvåvariabelresultat, TwoVar	158		

svar (sista), Ans	12	tvåvariabelresultat, TwoVar	158
T		TVM-argument	158
t-test, tTest	155	tvmFV()	157
T, transponera	149	tvmI()	157
ta bort		tvmN()	157
tomma element från lista	37	tvmPmt()	157
variabel, DelVar	37	tvmPV()	158
tak, ceiling()	19, 29	TwoVar, tvåvariabelresultat	158
talföljd, seq()	132	U	
$\tan^{-1}()$, arctangens	149	undermatris, subMat()	147
tan(), tangens	149	ungefärlig, approx()	12
tangens, tan()	149	unitV(), enhetsvektor	160
$\tanh^{-1}()$, hyperbolisk arctangens ...	150	unLock, låsa upp variabel eller	
tanh(), hyperbolisk tangens	150	variabelgrupp	161
täthetsfunktion för student-t, tPdf()	153	user-defined functions	34
tCdf(), student		user-defined functions and	
fördelningssannolikhet	151	programs	36
tecken		uteslutning med " " -operatorn	188
numerisk kod, ord()	107	uttryck	
sträng, char()	20	sträng till uttryck, expr()	48
tecken, sign()	137	utvärdera polynom, polyEval()	110
teckensträng, char()	20	V	
test for void, isVoid()	74	vänster, left()	75
Test_2S, 2-sample F test	55	variabel	
Text, kommando	151	skapa namn från en	
tidsjusterat pengavärde, antal		teckensträng	211
betalningsperioder	157	variabler	
tidsjusterat pengavärde,		lokal, Local	84
betalningsbelopp	157	rensa alla enstaka bokstäver ...	22
tidsjusterat pengavärde, framtida		ta bort, DelVar	37
värde	157	variabler och funktioner	
tidsjusterat pengavärde, nuvärde ...	158	kopiera	24
tidsjusterat pengavärde, ränta	157	variabler, låsa och låsa upp	63, 85, 161
tlInterval, t konfidensintervall	152	varians, variance()	161
tlInterval_2Samp, 2sempel t-		varningskoder och meddelanden ...	225
konfidensintervall	153	varPop()	161
tiopotens, $10^{\wedge}()$	187	varSamp(), sampelvarians	161
tomma element	206	vektorer	
tomma element, ta bort	37	enhet, unitV()	160
tPdf(), studentt täthetsfunktion ...	153	skalärprodukt, dotP()	41
trace()	154	vektorprodukt, crossP()	30
transponera, T	149	visa cylindrisk vektor, \triangleright Cylind ..	33
trappstegsform, ref()	121	vektorprodukt, crossP()	30
Try, felhanteringskommando	154	vinkel, angle()	9
Try, försök	154	visa cylindrisk vektor, \triangleright Cylind	33
tTest, t-test	155	visa data, Disp	39, 132
tTest_2Samp, 2-sampel t-test	156		

visa som	
binär, ►Base2	16
cylindrisk vektor, ►Cylind	33
decimal vinkel, ►DD	34
decimalt heltal, ►Base10	17
grad/minut/sekund, ►DMS	41
hexadecimal, ►Base16	18
ortogonal vektor, ►Rect	120
polär vektor, ►Polar	109
sfärisk vektor, ►Sphere	142
visning i grad/minut/sekund, ►DMS	41
void, testa om	74

W

warnCodes(), Warning codes	163
when(), när	163
While, medan	164

X

x ² , kvadrat	174
XNOR	179
xor, Booleskt exklusivt eller	164

Z

zInterval, z konfidensintervall	165
zInterval_1Prop, 1-proportion z- konfidensintervall	166
zInterval_2Prop, 2-proportion z- konfidensintervall	166
zInterval_2Samp, 2-sampel z- konfidensintervall	167
zTest	168
zTest_1Prop, 1-proportion z-test	169
zTest_2Prop, 2-proportion z-test	169
zTest_2Samp, 2-sampel z-test	170

Δ

Δlist(), listskillnad	82
------------------------	----

χ

χ ² 2way	20
χ ² Cdf()	21
χ ² GOF	21
χ ² Pdf()	21