

TI-Nspire™ CX Referenzhandbuch

Weitere Informationen zu TI Technology finden Sie in der Online-Hilfe unter
education.ti.com/eguide.

Wichtige Informationen

Außer im Fall anderslautender Bestimmungen der Lizenz für das Programm gewährt Texas Instruments keine ausdrückliche oder implizite Garantie, inklusive aber nicht ausschließlich sämtlicher impliziter Garantien der Handelsfähigkeit und Eignung für einen bestimmten Zweck, bezüglich der Programme und der schriftlichen Dokumentationen, und stellt dieses Material nur im „Ist-Zustand“ zur Verfügung. Unter keinen Umständen kann Texas Instruments für besondere, direkte, indirekte oder zufällige Schäden bzw. Folgeschäden haftbar gemacht werden, die durch Erwerb oder Benutzung dieses Materials verursacht werden, und die einzige und exklusive Haftung von Texas Instruments, ungeachtet der Form der Beanstandung, kann den in der Programmlizenz festgesetzten Betrag nicht überschreiten. Zudem haftet Texas Instruments nicht für Forderungen anderer Parteien jeglicher Art gegen die Anwendung dieses Materials.

© 2020 Texas Instruments Incorporated

Die aktuellen Produkte können geringfügig von den Abbildungen abweichen.

Inhaltsverzeichnis

Vorlagen für Ausdrücke	1
Alphabetische Auflistung	7
A	7
B	16
C	20
D	38
E	48
F	56
G	64
I	75
L	83
M	100
N	109
O	118
P	121
Q	128
R	131
S	147
T	168
U	181
V	182
W	183
X	186
Z	187
Sonderzeichen	193
TI-Nspire™ CX II – Zeichenbefehle	218
Grafikprogrammierung	218
Grafikbildschirm	218
Standardansicht und Einstellungen	219
Fehlermeldungen des Grafikbildschirms	220
Im Grafikmodus ungültige Befehle	220
C	222
D	223
F	227
G	229
P	230
F:	232
U	234

Leere (ungültige) Elemente	235
Tastenkürzel zum Eingeben mathematischer Ausdrücke	237
Auswertungsreihenfolge in EOS™ (Equation Operating System)	239
TI-Nspire CX II – TI-Basic Programmierungsfunktionen	241
Automatisches Einrücken im Programmierungsseditor	241
Verbesserte Fehlermeldungen für TI-Basic	241
Konstanten und Werte	244
Fehlercodes und -meldungen	245
Warncodes und -meldungen	254
Allgemeine Informationen	256
Online-Hilfe	256
Kontakt mit TI Support aufnehmen	256
Service- und Garantieinformationen	256
Index	257

Vorlagen für Ausdrücke

Vorlagen für Ausdrücke bieten Ihnen eine einfache Möglichkeit, mathematische Ausdrücke in der mathematischen Standardschreibweise einzugeben. Wenn Sie eine Vorlage eingeben, wird sie in der Eingabezeile mit kleinen Blöcken an den Positionen angezeigt, an denen Sie Elemente eingeben können. Der Cursor zeigt, welches Element eingegeben werden kann.

Verwenden Sie die Pfeiltasten oder drücken Sie **tab**, um den Cursor zur jeweiligen Position der Elemente zu bewegen, und geben Sie für jedes Element einen Wert oder Ausdruck ein. Drücken Sie **enter** oder **ctrl enter**, um den Ausdruck auszuwerten.

Vorlage Bruch

ctrl ÷ Tasten



Hinweis: Siehe auch / (Dividieren), Seite 195.

Beispiel:

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

Vorlage Exponent

^ Taste



Hinweis: Geben Sie den ersten Wert ein, drücken Sie **^** und geben Sie dann den Exponenten ein. Um den Cursor auf die Grundlinie zurückzusetzen, drücken Sie die rechte Pfeiltaste (**▶**).

Hinweis: Siehe auch ^ (Potenz), Seite 196.

Beispiel:

$$2^3 = 8$$

Vorlage Quadratwurzel

ctrl x² Tasten



Hinweis: Siehe auch √() (Quadratwurzel), Seite 206.

Beispiel:

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

Vorlage n-te Wurzel

ctrl \wedge Tasten



Hinweis: Siehe auch **root()**, Seite 143.

Beispiel:

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

2

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

$\{2, 3, 2.46621\}$

Vorlage e Exponent

ex Tasten



Potenz zur natürlichen Basis e

Hinweis: Siehe auch **e^(())**, Seite 48.

Example:

$$e^1$$

2.71828182846

Vorlage Logarithmus

ctrl \log Taste



Berechnet den Logarithmus zu einer bestimmten Basis. Bei der Standardbasis 10 wird die Basis weggelassen.

Hinweis: Siehe auch **log()**, Seite 95.

Beispiel:

$$\log_4(2.)$$

0.5

Vorlage Stückweise (2 Teile)

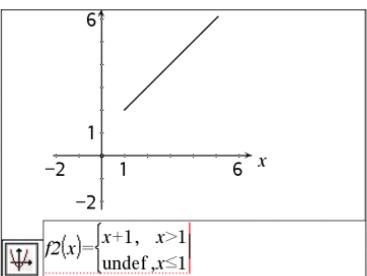
Katalog >



Ermöglicht es, Ausdrücke und Bedingungen für eine stückweise definierte Funktion aus zwei-Stücken zu erstellen. Um ein Stück hinzuzufügen, klicken Sie in die Vorlage und wiederholen die Vorlage.

Hinweis: Siehe auch **piecewise()**, Seite 122.

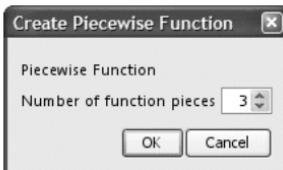
Beispiel:



Vorlage Stückweise (n Teile)

Katalog > 

Ermöglicht es, Ausdrücke und Bedingungen für eine stückweise definierte Funktion aus n -Teilen zu erstellen. Fragt nach n .



Hinweis: Siehe auch **piecewise()**, Seite 122.

Beispiel:

Siehe Beispiel für die Vorlage Stückweise (2 Teile).

Vorlage System von 2 Gleichungen

Katalog > 



Erzeugt ein System aus zwei linearen Gleichungen. Um einem vorhandenen System eine Zeile hinzuzufügen, klicken Sie in die Vorlage und wiederholen die Vorlage.

Hinweis: Siehe auch **system()**, Seite 168.

Beispiel:

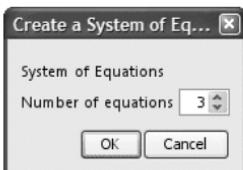
$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=-\frac{5}{2}$$

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

Vorlage System von n Gleichungen

Katalog > 

Ermöglicht es, ein System aus N linearen Gleichungen zu erzeugen. Fragt nach N .



Hinweis: Siehe auch **system()**, Seite 168.

Beispiel:

Siehe Beispiel für die Vorlage Gleichungssystem (2 Gleichungen).

Vorlage Absolutwert

Katalog > 



Hinweis: Siehe auch **abs()**, Seite 7.

Beispiel:

Vorlage Absolutwert

Katalog > 

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

Vorlage dd°mm'ss.ss"

Katalog > 

$$30^\circ 15' 10''$$

Ermöglicht es, Winkel im Format **dd°mm'ss.ss"** einzugeben, wobei **dd** für den Dezimalgrad, **mm** die Minuten und **ss.ss** die Sekunden steht.

Beispiel:

$$30^\circ 15' 10'' \quad 0.528011$$

Vorlage Matrix (2 x 2)

Katalog > 

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

Erzeugt eine 2 x 2 Matrix.

Beispiel:

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \cdot 5 \quad \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

Vorlage Matrix (1 x 2)

Katalog > 

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

Beispiel:

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

Vorlage Matrix (2 x 1)

Katalog > 

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

Beispiel:

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

Vorlage Matrix (m x n)

Katalog > 

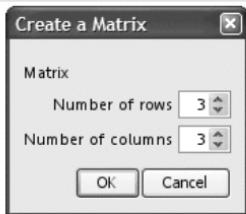
Die Vorlage wird angezeigt, nachdem Sie aufgefordert wurden, die Anzahl der Zeilen und Spalten anzugeben.

Beispiel:

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

Vorlage Matrix (m x n)

Katalog > 



Hinweis: Wenn Sie eine Matrix mit einer großen Zeilen- oder Spaltenanzahl erstellen, dauert es möglicherweise einen Augenblick, bis sie angezeigt wird.

Vorlage Summe (Σ)

Katalog > 

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

Beispiel:

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

Hinweis: Siehe auch $\Sigma()$ (**sumSeq**), Seite 207.

Vorlage Produkt (Π)

Katalog > 

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

Beispiel:

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

Hinweis: Siehe auch $\Pi()$ (**prodSeq**), Seite 207.

Vorlage Erste Ableitung

Katalog > 

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

Beispiel:

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

Vorlage Erste Ableitung

Katalog > 

Die Vorlage „Erste Ableitung“ lässt sich verwenden, um die erste Ableitung an einem Punkt numerisch durch automatische Ableitungsmethoden zu berechnen.

Hinweis: Siehe auch **d()** (**Ableitung**), Seite 205.

Vorlage Zweite Ableitung

Katalog > 

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

Die Vorlage „Zweite Ableitung“ lässt sich verwenden, um die zweite Ableitung an einem Punkt numerisch durch automatische Ableitungsmethoden zu berechnen.

Hinweis: Siehe auch **d()** (**Ableitung**), Seite 205.

Beispiel:

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

18

Vorlage Bestimmtes Integral

Katalog > 

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

Mit der Vorlage „Bestimmtes Integral“ können Sie das bestimmte Integral numerisch berechnen. Hierzu wird dieselbe Methode wie bei **nInt()** verwendet.

Hinweis: Siehe auch **nInt()**, Seite 112.

Beispiel:

$$\int_0^{10} x^2 \, dx$$

333.333

Alphabetische Auflistung

Elemente, deren Namen nicht alphabetisch sind (wie +, !, und >) finden Sie am Ende dieses Abschnitts (Seite 193). Wenn nicht anders angegeben, wurden sämtliche Beispiele im standardmäßigen Reset-Modus ausgeführt, wobei alle Variablen als nicht definiert angenommen wurden.

A

abs() (Absolutwert)

abs(Wert1)⇒Wert

abs(Liste1)⇒Liste

abs(Matrix1)⇒Matrix

Katalog >

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

$$|2^{-3 \cdot i}| \quad 3.60555$$

Gibt den Absolutwert des Arguments zurück.

Hinweis: Siehe auch **Vorlage Absolutwert**, Seite 3.

Ist das Argument eine komplexe Zahl, wird der Betrag der Zahl zurückgegeben.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

amortTbl()

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

Katalog >

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

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

Amortisationsfunktion, die eine Matrix als Amortisationstabellen für eine Reihe von TVM-Argumenten zurückgibt.

NPmt ist die Anzahl der Zahlungen, die in der Tabelle enthalten sein müssen. Die Tabelle beginnt mit der ersten Zahlung.

N, I, PV, Pmt, FV, PpY, CpY und PmtAt werden in der TVM-Argumentetabelle (Seite 179) beschrieben.

- Wenn Sie Pmt nicht angeben, wird standardmäßig Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt) eingesetzt.

- Wenn Sie FV nicht angeben, wird standardmäßig $FV=0$ eingesetzt.
- Die Standardwerte für PpY , CpY und $PmtAt$ sind dieselben wie bei den TVM-Funktionen.

WertRunden (roundValue) legt die Anzahl der Dezimalstellen für das Runden fest.
Standard=2.

Die Spalten werden in der Ergebnismatrix in der folgenden Reihenfolge ausgegeben:
Zahlungsnummer, Zinsanteil,
Tilgungsanteil, Saldo.

Der in Zeile n angezeigte Saldo ist der Saldo nach Zahlung n .

Sie können die ausgegebene Matrix als Eingabe für die anderen Amortisationsfunktionen $\SigmaInt()$ und $\SigmaPrn()$, Seite 208, und $bal()$, Seite 16, verwenden.

and (und)

Boolescher Ausdr1 and Boolescher Ausdr2 \Rightarrow Boolescher Ausdruck

Boolesche Liste1 and Boolesche Liste2
 \Rightarrow Boolesche Liste

Boolesche Matrix1 and Boolesche Matrix2 \Rightarrow Boolesche Matrix

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des ursprünglichen Terms zurück.

Ganzzahl1 and Ganzzahl2 \Rightarrow Ganzzahl

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer **and**-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn beide Bits 1 sind; andernfalls ist das Ergebnis 0. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Im Hex-Modus:

0h7AC36 and 0h3D5F	0h2C16
--------------------	--------

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 and 0b100	0b100
--------------------	-------

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 32-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen.

Im Dec-Modus:

37 and 0b100

4

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

angle() (Winkel)

angle(*Wert1*)⇒*Wert*

Gibt den Winkel des Arguments zurück, wobei das Argument als komplexe Zahl interpretiert wird.

Im Grad-Modus:

angle(0+2·i)

90

Im Neugrad-Modus:

angle(0+3·i)

100

Im Bogenmaß-Modus:

angle(1+i)

0.785398

angle({1+2·i, 3+0·i, 0-4·i})

{1.10715, 0., -1.5708}

angle(*Liste1*)⇒*Liste*

angle(*Matrix1*)⇒*Matrix*

Gibt als Liste oder Matrix die Winkel der Elemente aus *Liste1* oder *Matrix1* zurück, wobei jedes Element als komplexe Zahl interpretiert wird, die einen zweidimensionalen kartesischen Koordinatenpunkt darstellt.

ANOVA

ANOVA *Liste1, Liste2[, Liste3, ..., Liste20]*
[, *Flag*]

Führt eine einfache Varianzanalyse durch, um die Mittelwerte von zwei bis maximal 20 Grundgesamtheiten zu vergleichen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162)

Flag=0 für Daten, *Flag*=1 für Statistik

Ausgabevariable	Beschreibung
stat.F	Wert der F Statistik
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Gruppen-Freiheitsgrade
stat.SS	Summe der Fehlerquadrate zwischen den Gruppen
stat.MS	Mittlere Quadrate der Gruppen
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittleres Quadrat für die Fehler
stat.sp	Verteilte Standardabweichung
stat.xbarlist	Mittelwerte der Eingabelisten
stat.CLowerList	95 % Konfidenzintervalle für den Mittelwert jeder Eingabeliste
stat.CUpperList	95 % Konfidenzintervalle für den Mittelwert jeder Eingabeliste

ANOVA2way (ANOVA 2fach)

ANOVA2way *Liste1*,*Liste2*
[,*Liste3*,...,*Liste10*][,LevZei]

Berechnet eine zweifache Varianzanalyse, um die Mittelwerte von zwei bis maximal 10 Grundgesamtheiten zu vergleichen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162)

LevZei=0 für Block

LevZei=2,3,...,*Len*-1, für Faktor zwei, wobei
Len=length(*Liste1*)=length(*Liste2*) = ... =
length(*Liste10*) und *Len* / *LevZei* ∈ €
{2,3,...}

Ausgaben: Block-Design

Ausgabevariable	Beschreibung
stat.F	F Statistik des Spaltenfaktors
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade des Spaltenfaktors
stat.SS	Summe der Fehlerquadrate des Spaltenfaktors
stat.MS	Mittlere Quadrate für Spaltenfaktor
stat.FBlock	F Statistik für Faktor
stat.PValBlock	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat.dfBlock	Freiheitsgrade für Faktor
stat.SSBlock	Summe der Fehlerquadrate für Faktor
stat.MSBlock	Mittlere Quadrate für Faktor
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittlere Quadrate für die Fehler
stat.s	Standardabweichung des Fehlers

Ausgaben des SPALTENFAKTORS

Ausgabevariable	Beschreibung
stat.Fcol	F Statistik des Spaltenfaktors
stat.PValCol	Wahrscheinlichkeitswert des Spaltenfaktors
stat.dfCol	Freiheitsgrade des Spaltenfaktors
stat.SSCol	Summe der Fehlerquadrate des Spaltenfaktors
stat.MSCol	Mittlere Quadrate für Spaltenfaktor

Ausgaben des ZEILENFAKTORS

Ausgabevariable	Beschreibung
stat.Frow	F Statistik des Zeilenfaktors
stat.PValRow	Wahrscheinlichkeitswert des Zeilenfaktors
stat.dfRow	Freiheitsgrade des Zeilenfaktors

Ausgabevariable	Beschreibung
stat.SSRow	Summe der Fehlerquadrate des Zeilenfaktors
stat.MSRow	Mittlere Quadrate für Zeilenfaktor

INTERAKTIONS-Ausgaben

Ausgabevariable	Beschreibung
stat.FInteract	F Statistik der Interaktion
stat.PValInteract	Wahrscheinlichkeitswert der Interaktion
stat.dfInteract	Freiheitsgrade der Interaktion
stat.SSInteract	Summe der Fehlerquadrate der Interaktion
stat.MSInteract	Mittlere Quadrate für Interaktion

FEHLER-Ausgaben

Ausgabevariable	Beschreibung
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittlere Quadrate für die Fehler
s	Standardabweichung des Fehlers

Ans (Antwort)

ctrl **(→)** **Taste**

Ans⇒Wert

Gibt das Ergebnis des zuletzt ausgewerteten Ausdrucks zurück.

56	56
56+4	60
60+4	64

approx() (Approximieren)

Katalog >

approx(Wert1)⇒Zahl

Gibt die Auswertung des Arguments ungeachtet der aktuellen Einstellung des Modus **Auto oder Näherung** als Dezimalwert zurück, sofern möglich.

Gleichwertig damit ist die Eingabe des Arguments und Drücken von **ctrl enter**.

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

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

approx(Liste1)⇒Liste

approx(Matrix1)⇒Matrix

Gibt, sofern möglich, eine Liste oder *Matrix* zurück, in der jedes Element dezimal ausgewertet wurde.

►approxFraction()

Katalog >

Wert ►approxFraction([Tol])⇒Wert

Liste ►approxFraction([Tol])⇒Liste

Matrix ►approxFraction([Tol])⇒Matrix

Gibt die Eingabe als Bruch mit der Toleranz *Tol* zurück. Wird *tol* weggelassen, so wird die Toleranz 5.E-14 verwendet.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie @>**approxFraction** (...) eintippen.

$\frac{1}{2} + \frac{1}{3} + \tan(\pi)$	0.833333
0.8333333333333333	►approxFraction(5.E-14)
$\frac{5}{6}$	
{π, 1.5}	►approxFraction(5.E-14)

approxRational()

Katalog >

approxRational(Wert[, Tol])⇒Wert

approxRational(Liste[, Tol])⇒Liste

approxRational(Matrix[, Tol])⇒Matrix

Gibt das Argument als Bruch mit der Toleranz *Tol* zurück. Wird *tol* weggelassen, so wird die Toleranz 5.E-14 verwendet.

approxRational(0.333, 5·10 ⁻⁵)	$\frac{333}{1000}$
approxRational({0.2, 0.33, 4.125}, 5.E-14)	$\left\{\frac{1}{5}, \frac{33}{100}, \frac{33}{8}\right\}$

arccos()

Siehe $\cos^{-1}()$, Seite 29

arccosh()

Siehe $\cosh^{-1}()$, Seite 30.

arccot()

Siehe $\cot^{-1}()$, Seite 31.

arccoth()

Siehe $\coth^{-1}()$, Seite 32.

arccsc()

Siehe $\csc^{-1}()$, Seite 35.

arccsch()

Siehe $\csch^{-1}()$, Seite 35.

arcsec()

Siehe $\sec^{-1}()$, Seite 147.

arcsech()

Siehe $\sech^{-1}()$, Seite 148.

arcsin()

Siehe $\sin^{-1}()$, Seite 156.

arcsinh()

Siehe $\sinh^{-1}()$, Seite 158.

augment() (Erweitern)**Katalog** > **augment(Liste1, Liste2)⇒Liste**

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

Gibt eine neue Liste zurück, die durch Anfügen von *Liste2* ans Ende von *Liste1* erzeugt wurde.

augment(Matrix1, Matrix2)⇒Matrix

Gibt eine neue Matrix zurück, die durch Anfügen von *Matrix2* an *Matrix1* erzeugt wurde. Wenn das Zeichen „,” verwendet wird, müssen die Matrizen gleiche Zeilendimensionen besitzen, und *Matrix2* wird spaltenweise an *Matrix1* angefügt. Verändert weder *Matrix1* noch *Matrix2*.

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

avgRC() (Durchschnittliche Änderungsrate)**Katalog** > **avgRC(Ausdr1, Var [=Wert] [, Schritt])⇒Ausdruck**

x:=2 2

avgRC(Ausdr1, Var [=Wert] [, Liste1])⇒Liste

avgRC(x^2-x+2,x) 3.001

avgRC(Liste1, Var [=Wert] [, Schritt])⇒Liste

avgRC(x^2-x+2,x,,1) 3.1

avgRC(Matrix1, Var [=Wert] [, Schritt])⇒Matrix

avgRC(x^2-x+2,x,3) 6

Gibt den rechtsseitigen Differenzenquotienten zurück (durchschnittliche Änderungsrate).

Ausdr1 kann eine benutzerdefinierte Funktion sein (siehe **Func**).

Wenn *Wert* angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

Schritt ist der Schrittwert. Wird *Schritt* nicht angegeben, wird als Vorgabewert 0,001 benutzt.

Beachten Sie, dass die ähnliche Funktion **centralDiff()** den zentralen Differenzenquotienten benutzt.

B**bal()**

bal(*NPmt, NI, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]*)⇒*Wert*

bal(*NPmt, AmortTabelle*)⇒*Wert*

Amortisationsfunktion, die den Saldo nach einer angegebenen Zahlung berechnet.

N, I, PV, Pmt, FV, PpY, CpY und PmtAt werden in der TVM-Argumentetabelle (Seite 179) beschrieben.

NPmt bezeichnet die Zahlungsnummer, nach der die Daten berechnet werden sollen.

N, I, PV, Pmt, FV, PpY, CpY und PmtAt werden in der TVM-Argumentetabelle (Seite 179) beschrieben.

- Wenn Sie *Pmt* nicht angeben, wird standardmäßig *Pmt=tvmPmt* (*N,I,PV,FV,PpY,CpY,PmtAt*) eingesetzt.
- Wenn Sie *FV* nicht angeben, wird standardmäßig *FV=0* eingesetzt.
- Die Standardwerte für *PpY*, *CpY* und *PmtAt* sind dieselben wie bei den TVM-Funktionen.

bal(5,6,5.75,5000,,12,12)	833.11
---------------------------	--------

tbl:=amortTbl(6,6,5.75,5000,,12,12)

0	0.	0.	5000.
1	-23.35	-825.63	4174.37
2	-19.49	-829.49	3344.88
3	-15.62	-833.36	2511.52
4	-11.73	-837.25	1674.27
5	-7.82	-841.16	833.11
6	-3.89	-845.09	-11.98

bal(4, <i>tbl</i>)	1674.27
---------------------	---------

WertRunden (roundValue) legt die Anzahl der Dezimalstellen für das Runden fest.
Standard=2.

bal(NPmt,AmortTabelle) berechnet den Saldo nach jeder Zahlungsnummer *NPmt* auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle (amortTable)* muss eine Matrix in der unter **amortTbl()**, Seite 7, beschriebenen Form sein.

Hinweis: Siehe auch **ΣInt()** und **ΣPrn()**, Seite 208.

►Base2

Ganzzahl1 ►Base2⇒*Ganzzahl*

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Base2 eintippen.

Konvertiert *Ganzzahl1* in eine Binärzahl. Dual- oder Hexadezimalzahlen weisen stets das Präfix 0b bzw. 0h auf. Null (nicht Buchstabe O) und b oder h.

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

Ohne Präfix wird *Ganzzahl1* als Dezimalzahl behandelt (Basis 10). Das Ergebnis wird unabhängig vom Basis-Modus binär angezeigt.

Negative Zahlen werden als Binärikomplement angezeigt. Beispiel:

-1 wird angezeigt als

0hFFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 Einsen) im Binärmodus

-2⁶³ wird angezeigt als

0h8000000000000000 im Hex-Modus

256	►Base2	0b100000000
-----	--------	-------------

0h1F	►Base2	0b11111
------	--------	---------

0b100...000 (63 Nullen) im Binärmodus

Geben Sie eine dezimale ganze Zahl ein, die außerhalb des Bereichs einer 64-Bit-Dualform mit Vorzeichen liegt, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Die folgenden Beispiele verdeutlichen, wie diese Anpassung erfolgt:

2^{63} wird zu -2^{63} und wird angezeigt als

0h8000000000000000 im Hex-Modus

0b100...000 (63 Nullen) im Binärmodus

2^{64} wird zu 0 und wird angezeigt als

0h0 im Hex-Modus

0b0 im Binärmodus

$-2^{63} - 1$ wird zu $2^{63} - 1$ und wird angezeigt als

0h7FFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 1's) im Binärmodus

►Base10

Ganzzahl1 ►Base10⇒Ganzzahl

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Base10 eintippen.

Konvertiert Ganzzahl1 in eine Dezimalzahl (Basis 10). Ein binärer oder hexadezimaler Eintrag muss stets das Präfix 0b bzw. 0h aufweisen.

0b binäre_Zahl

0h hexadezimale_Zahl

0b10011 ►Base10

19

0h1F ►Base10

31

Null (nicht Buchstabe O) und b oder h.

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

Ohne Präfix wird *Ganzzahl1* als Dezimalzahl behandelt. Das Ergebnis wird unabhängig vom Basis-Modus dezimal angezeigt.

►Base16

Ganzzahl1 ►Base16⇒*Ganzzahl*

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Base16** eintippen.

256►Base16	0h100
0b111100001111►Base16	0hF0F

Wandelt *Ganzzahl1* in eine Hexadezimalzahl um. Dual- oder Hexadezimalzahlen weisen stets das Präfix Ob bzw. Oh auf.

Ob *binäre_Zahl*

Oh *hexadezimale_Zahl*

Null (nicht Buchstabe O) und b oder h.

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

Ohne Präfix wird *Ganzzahl1* als Dezimalzahl behandelt (Basis 10). Das Ergebnis wird unabhängig vom Basis-Modus hexadezimal angezeigt.

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ►Base2, Seite 17.

binomCdf()

binomCdf(n, p)⇒*Liste*

binomCdf()Katalog > **binomCdf**

$\{n, p, \text{untereGrenze}, \text{obereGrenze}\} \Rightarrow \text{Zahl}$,
wenn *untereGrenze* und *obereGrenze*
Zahlen sind, *Liste*, wenn *untereGrenze* und
obereGrenze Listen sind

binomCdf(*n,p,obereGrenze*) für $P(0 \leq X \leq \text{obereGrenze}) \Rightarrow \text{Zahl}$, wenn *obereGrenze*
eine Zahl ist, *Liste*, wenn *obereGrenze* eine
Liste ist

Berechnet die kumulative
Wahrscheinlichkeit für die diskrete
Binomialverteilung mit *n* Versuchen und der
Wahrscheinlichkeit *p* für einen Erfolg in
jedem Einzelversuch.

Für $P(X \leq \text{obereGrenze})$ setzen Sie
untereGrenze=0

binomPdf()Katalog > **binomPdf(*n,p*)** \Rightarrow *Liste*

binomPdf(*n,p,XWert*) \Rightarrow *Zahl*, wenn *XWert*
eine Zahl ist, *Liste*, wenn *XWert* eine Liste
ist

Berechnet die Wahrscheinlichkeit an einem
XWert für die diskrete Binomialverteilung
mit *n* Versuchen und der Wahrscheinlichkeit
p für den Erfolg in jedem Einzelversuch.

C**ceiling() (Obergrenze)**Katalog > **ceiling(*Wert1*)** \Rightarrow *Wert*ceiling(.456)

1.

Gibt die erste ganze Zahl zurück, die \geq dem
Argument ist.

Das Argument kann eine reelle oder eine
komplexe Zahl sein.

Hinweis: Siehe auch **floor()**.

ceiling() (Obergrenze)**Katalog > ****ceiling(Liste1)⇒Liste****ceiling(Matrix1)⇒Matrix**

Für jedes Element einer Liste oder Matrix wird die kleinste ganze Zahl, die größer oder gleich dem Element ist, zurückgegeben.

$\text{ceiling}(\{-3.1, 1.2, 5\})$	$\{-3, 1, 3\}$
$\text{ceiling}\begin{pmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{pmatrix}$	$\begin{pmatrix} 0 & -3 \cdot i \\ 2 & 4 \end{pmatrix}$

centralDiff()**Katalog > ****centralDiff(Ausdr1,Var [=Wert]
[,Schritt])⇒Ausdruck****centralDiff(Ausdr1,Var
[,Schritt]) | Var=Wert⇒Ausdruck****centralDiff(Ausdr1,Var [=Wert]
[,Liste])⇒Liste****centralDiff(Liste1,Var [=Wert]
[,Schritt])⇒Liste****centralDiff(Matrix1,Var [=Wert]
[,Schritt])⇒Matrix**

Gibt die numerische Ableitung unter Verwendung des zentralen Differenzenquotienten zurück.

Wenn *Wert* angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

Schritt ist der Schrittwert. Wird *Schritt* nicht angegeben, wird als Vorgabewert 0,001 benutzt.

Wenn Sie *Liste1* oder *Matrix1* verwenden, wird die Operation über die Werte in der Liste oder die Matrixelemente abgebildet.

Hinweis: Siehe auch .

char() (Zeichenstring)**Katalog > ****char(Ganzzahl)⇒Zeichen**

$\text{char}(38)$	$"\&"$
$\text{char}(65)$	$"A"$

Gibt ein Zeichenstring zurück, das das Zeichen mit der Nummer *Ganzzahl* aus dem Zeichensatz des Handhelds enthält. Der gültige Wertebereich für *Ganzzahl* ist 0–65535.

 χ^2 way

χ^2 way *BeobMatrix*

chi χ^2 way *BeobMatrix*

Berechnet eine χ^2 Testgröße auf Grundlage einer beobachteten Matrix *BeobMatrix*. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Matrix finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: sum(beobachtet - erwartet) ² /erwartet
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.ExpMat	Berechnete Kontingenztafel der erwarteten Häufigkeiten bei Annahme der Nullhypothese
stat.CompMat	Berechnete Matrix der Chi-Quadrat-Summanden in der Testgröße

 χ^2 Cdf()

χ^2 Cdf

(

untereGrenze

,*obereGrenze*,*Freigrad*) \Rightarrow Zahl, wenn
untereGrenze und *obereGrenze* Zahlen
 sind, Liste, wenn *untereGrenze* und
obereGrenze Listen sind

chi χ^2 Cdf

(

untereGrenze

,*obereGrenze*,*Freiheitsgrad*) \Rightarrow Zahl, wenn
untereGrenze und *obereGrenze* Zahlen
 sind, Liste, wenn *untereGrenze* und
obereGrenze Listen sind

Berechnet die Verteilungswahrscheinlichkeit
 χ^2 zwischen *untereGrenze* und
obereGrenze für die angegebenen
 Freiheitsgrade *FreiGrad*.

Für $P(X \leq \text{obereGrenze})$ setzen Sie
untereGrenze= 0.

Informationen zu den Auswirkungen leerer
 Elemente in einer Liste finden Sie unter
 "Leere (ungültige) Elemente" (Seite 235).

$\chi^2\text{GOF}$ *BeobListe*,*expListe*,*FreiGrad*

chi2GOF *BeobListe*,*expListe*,*FreiGrad*

Berechnet eine Testgröße, um zu
 überprüfen, ob die Stichprobendaten aus
 einer Grundgesamtheit stammen, die einer
 bestimmten Verteilung genügt. *obsList* ist
 eine Liste von Zählern und muss Ganzzahlen
 enthalten. Eine Zusammenfassung der
 Ergebnisse wird in der Variablen *stat.results*
 gespeichert. (Seite 162)

Informationen zu den Auswirkungen leerer
 Elemente in einer Liste finden Sie unter
 "Leere (ungültige) Elemente" (Seite 235).

Ausgabeverable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: sum((beobachtet - erwartet) ² /erwartet)
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.Complist	Liste der Chi-Quadrat-Summanden in der Testgröße

$\chi^2\text{Pdf}()$

Katalog > 

$\chi^2\text{Pdf}(X\text{Wert}, \text{FreiGrad}) \Rightarrow \text{Zahl}$, wenn
 $X\text{Wert}$ eine Zahl ist, *Liste*, wenn $X\text{Wert}$ eine
Liste ist

$\text{chi2Pdf}(X\text{Wert}, \text{FreiGrad}) \Rightarrow \text{Zahl}$, wenn
 $X\text{Wert}$ eine Zahl ist, *Liste*, wenn $X\text{Wert}$
eine Liste ist

Berechnet die
Wahrscheinlichkeitsdichtefunktion (Pdf)
einer χ^2 -Verteilung an einem bestimmten
 $X\text{Wert}$ für die vorgegebenen Freiheitsgrade
FreiGrad.

Informationen zu den Auswirkungen leerer
Elemente in einer Liste finden Sie unter
"Leere (ungültige) Elemente" (Seite 235).

ClearAZ (LöscheAZ)

Katalog > 

ClearAZ

Löscht alle Variablen mit einem Zeichen im
aktuellen Problembereich.

Wenn eine oder mehrere Variablen
gesperrt sind, wird bei diesem Befehl eine
Fehlermeldung angezeigt und es werden
nur die nicht gesperrten Variablen gelöscht.
Siehe **unLock**, Seite 182

$5 \rightarrow b$	5
b	5
ClearAZ	<i>Done</i>
b	"Error: Variable is not defined"

ClrErr (LöFehler)

Katalog > 

ClrErr

Löscht den Fehlerstatus und setzt die
Systemvariable *FehlerCode (errCode)* auf
Null.

Ein Beispiel für **ClrErr** finden Sie als Beispiel
2 im Abschnitt zum Befehl **Versuche (Try)**,
Seite 175.

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** (**Ügebefehler**) verwenden. Wenn der Fehler verarbeitet oder ignoriert werden soll, verwenden Sie **ClrErr**. Wenn nicht bekannt ist, was mit dem Fehler zu tun ist, verwenden Sie **PassErr**, um ihn an den nächsten Error Handler zu übergeben. Wenn keine weiteren **Try...Else...EndTry** Error Handler unerledigt sind, wird das Fehlerdialogfeld als normal angezeigt.

Hinweis: Siehe auch **PassErr**, Seite 122, und **Try**, Seite 175.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

colAugment() (Spaltenerweiterung)

colAugment(*Matrix1*, *Matrix2*) \Rightarrow **Matrix**

Gibt eine neue Matrix zurück, die durch Anfügen von *Matrix2* an *Matrix1* erzeugt wurde. Die Matrizen müssen gleiche Spaltendimensionen haben, und *Matrix2* wird zeilenweise an *Matrix1* angefügt. Verändert weder *Matrix1* noch *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,m2</i>)	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

colDim() (Spaltendimension)

colDim(*Matrix*) \Rightarrow **Ausdruck**

Gibt die Anzahl der Spalten von *Matrix* zurück.

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

colNorm() (Spaltennorm)

colNorm(*Matrix*) \Rightarrow **Ausdruck**

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

Gibt das Maximum der Summen der absoluten Elementwerte der Spalten von *Matrix* zurück.

Hinweis: Undefinierte Matrixelemente sind nicht zulässig. Siehe auch **rowNorm()**.

conj() (Komplex Konjugierte)

conj(Wert I) \Rightarrow Wert

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

conj(Liste I) \Rightarrow Liste

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

conj(Matrix I) \Rightarrow Matrix

Gibt das komplex Konjugierte des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

constructMat()

constructMat
(Ausdr, Var1, Var2, AnzZeilen, AnzSpalten)
 \Rightarrow Matrix

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

Gibt eine Matrix auf der Basis der Argumente zurück.

Ausdr ist ein Ausdruck in Variablen Var1 und Var2. Die Elemente in der resultierenden Matrix ergeben sich durch Berechnung von Ausdr für jeden inkrementierten Wert von Var1 und Var2.

Var1 wird automatisch von 1 bis AnzZeilen inkrementiert. In jeder Zeile wird Var2 inkrementiert von 1 bis AnzSpalten.

CopyVar

Katalog >

CopyVar *Var1, Var2*

CopyVar *Var1., Var2.*

CopyVar *Var1, Var2* kopiert den Wert der Variablen *Var1* auf die Variable *Var2* und erstellt ggf. *Var2*. Variable *Var1* muss einen Wert haben.

Wenn *Var1* der Name einer vorhandenen benutzerdefinierten Funktion ist, wird die Definition dieser Funktion nach Funktion *Var2* kopiert. Funktion *Var1* muss definiert sein.

Var1 muss die Benennungsregeln für Variablen erfüllen oder muss ein indirekter Ausdruck sein, der sich zu einem Variablenamen vereinfachen lässt, der den Regeln entspricht.

CopyVar *Var1., Var2.* kopiert alle Mitglieder der *Var1.*-Variablengruppe auf die *Var2.*-Gruppe und erstellt ggf. *Var2..*

Var1. muss der Name einer bestehenden Variablengruppe sein, wie die Statistikergebnisse *stat.* *nn* oder Variablen, die mit der Funktion **LibShortcut()** erstellt wurden. Wenn *Var2.* schon vorhanden ist, ersetzt dieser Befehl alle Mitglieder, die zu beiden Gruppen gehören, und fügt die Mitglieder hinzu, die noch nicht vorhanden sind. Wenn einer oder mehrere Teile von *Var2.* gesperrt ist/sind, wird kein Teil von *Var2.* geändert.

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

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

corrMat() (Korrelationsmatrix)

Katalog >

corrMat(*Liste1, Liste2[, ..., Liste20]*)

Berechnet die Korrelationsmatrix für die erweiterte Matrix [*Liste1* *Liste2* ... *Liste20*].

cos() (Kosinus)

Taste

cos(*Wert1*) \Rightarrow *Wert*

Im Grad-Modus:

cos() (Kosinus)

trig Taste

cos(Liste1)⇒Liste

cos(Wert1) gibt den Kosinus des Arguments als Wert zurück.

cos(Liste1) gibt in Form einer Liste für jedes Element in *Liste1* den Kosinus zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um den Winkelmodus vorübergend aufzuheben.

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

Im Bogenmaß-Modus:

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

cos(Quadratmatrix1)⇒Quadratmatrix

Gibt den Matrix-Kosinus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Kosinus jedes einzelnen Elements.

Wenn eine skalare Funktion f(A) auf *Quadratmatrix1* (A) angewendet wird, erfolgt die Berechnung des Ergebnisses durch den Algorithmus:

Berechnung der Eigenwerte (λ_i) und Eigenvektoren (V_i) von A.

Quadratmatrix1 muss diagonalisierbar sein. Sie darf auch keine symbolischen Variablen ohne zugewiesene Werte enthalten.

Bildung der Matrizen:

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

cos() (Kosinus)

trig Taste

Dann ist $A = X B X^{-1}$ und $f(A) = X f(B) X^{-1}$.

Beispiel: $\cos(A) = X \cos(B) X^{-1}$, wobei:

$\cos(B) =$

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

Alle Berechnungen werden unter Verwendung von Fließkomma-Operationen ausgeführt.

cos⁻¹⁽⁾ (Arkuskosinus)

trig Taste

$\cos^{-1}(Wert\,I) \Rightarrow Wert$

Im Grad-Modus:

$\cos^{-1}(Liste\,I) \Rightarrow Liste$

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

0.

$\cos^{-1}(Wert\,I)$ gibt den Winkel zurück, dessen Kosinus $Wert\,I$ ist.

Im Neugrad-Modus:

$\cos^{-1}(Liste\,I)$ gibt in Form einer Liste für jedes Element aus $Liste\,I$ den inversen Kosinus zurück.

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

100.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Im Bogenmaß-Modus:

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

{1.5708, 1.36944, 1.0472}

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie `arccos (...)` eintippen.

$\cos^{-1}(Quadratmatrix\,I) \Rightarrow Quadratmatrix$

Gibt den inversen Matrix-Kosinus von $Quadratmatrix\,I$ zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Kosinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

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

$Quadratmatrix\,I$ muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ▲ und ▶, um den Cursor zu bewegen.

cosh() (Cosinus hyperbolicus)

Katalog >

cosh(Wert1)⇒Wert**cosh(Liste1)⇒Liste**

cosh(Wert1) gibt den Cosinus hyperbolicus des Arguments zurück.

cosh(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den Cosinus hyperbolicus zurück.

cosh(Quadratmatrix1)⇒Quadratmatrix

Gibt den Matrix-Cosinus hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Cosinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Grad-Modus:

$$\cosh\left(\left(\frac{\pi}{4}\right)r\right) \quad 1.74671\text{e}19$$

Im Bogenmaß-Modus:

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

cosh⁻¹() (Arkuskosinus hyperbolicus)

Katalog >

cosh⁻¹(Wert1)⇒Wert

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

cosh⁻¹(Liste1)⇒Liste

$$\cosh^{-1}\{1,2,1,3\} \quad \{0,1.37286,1.76275\}$$

cosh⁻¹(Wert1) gibt den inversen Cosinus hyperbolicus des Arguments zurück.

cosh⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Cosinus hyperbolicus zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccosh (...)** eintippen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

cosh⁻¹⁽⁾ (Arkuskosinus hyperbolicus)

Katalog >

cosh⁻¹(Quadratmatrix I) \Rightarrow Quadratmatrix

Gibt den inversen Matrix-Cosinus hyperbolicus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Cosinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

cot() (Kotangens)

Taste

cot(WertI) \Rightarrow Wert

Im Grad-Modus:

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

cot(ListeI) \Rightarrow Liste

Gibt den Kotangens von *WertI* oder eine Liste der Kotangens aller Elemente in *ListeI* zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um den Winkelmodus vorübergend aufzuheben.

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

cot⁻¹⁽⁾ (Arkuskotangens)

Taste

cot⁻¹(WertI) \Rightarrow Wert

Im Grad-Modus:

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

cot⁻¹(ListeI) \Rightarrow Liste

Gibt entweder den Winkel, dessen Kotangens *WertI* ist, oder eine Liste der inversen Kotangens aller Elemente in *ListeI* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

cot⁻¹() (Arkuskotangens)

trig Taste

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie `arccot (...)` eintippen.

cot⁻¹(1)

0.785398

coth() (Kotangens hyperbolicus)

Katalog >

coth(Wert1)⇒Wert

coth(1.2)

1.19954

coth(Liste1)⇒Liste

coth({1,3,2})

{1.31304,1.00333}

Gibt den hyperbolischen Kotangens von *Ausdr1* oder eine Liste der hyperbolischen Kotangens aller Elemente in *Liste1* zurück.

coth⁻¹() (Arkuskotangens hyperbolicus)

Katalog >

coth⁻¹(Wert1)⇒Wert

coth⁻¹(3.5)

0.293893

coth⁻¹(Liste1)⇒Liste

coth⁻¹({-2,2,1,6})

{-0.549306,0.518046,0.168236}

Gibt den inversen hyperbolischen Kotangens von *Wert1* oder eine Liste der inversen hyperbolischen Kotangens aller Elemente in *Liste1* zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie `arccoth (...)` eintippen.

count() (zähl)

Katalog >

count(Wert1oderListe1 [,Wert2oderListe2 [...]])⇒Wert

count(2,4,6)

3

count({2,4,6})

3

count(2,{4,6},{8 10},[12 14])

7

Gibt die kumulierte Anzahl aller Elemente in den Argumenten zurück, deren Auswertungsergebnisse numerische Werte sind.

Jedes Argument kann ein Ausdruck, ein Wert, eine Liste oder eine Matrix sein. Sie können Datenarten mischen und Argumente unterschiedlicher Dimensionen verwenden.

Für eine Liste, eine Matrix oder einen Zellenbereich wird jedes Element daraufhin ausgewertet, ob es in die Zählung eingeschlossen werden soll.

count() (zähl)

Katalog >

Innerhalb der Lists & Spreadsheet Applikation können Sie anstelle eines beliebigen Arguments auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

countIf()

Katalog >

countIf(Liste,Kriterien)⇒Wert

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

Gibt die kumulierte Anzahl aller Elemente in der *Liste* zurück, die die festgelegten *Kriterien* erfüllen.

Zählt die Anzahl der Elemente, die 3 entsprechen.

Kriterien können sein:

- Ein Wert, ein Ausdruck oder eine Zeichenfolge. So zählt zum Beispiel 3 nur Elemente in der *Liste*, die vereinfacht den Wert 3 ergeben.
- Ein Boolescher Ausdruck, der das Sonderzeichen ? als Platzhalter für jedes Element verwendet. Beispielsweise zählt ?<5 nur die Elemente in der *Liste*, die kleiner als 5 sind.

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

Zählt die Anzahl der Elemente, die "def." entsprechen

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

Zählt 1 und 3.

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

Zählt 3, 5 und 7.

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

Zählt 1, 3, 7 und 9.

Innerhalb der Lists & Spreadsheet

Applikation können Sie anstelle der *Liste* auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente in der Liste werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Hinweis: Siehe auch **sumIf()**, Seite 167, und **frequency()**, Seite 62.

Katalog >

cPolyRoots()

cPolyRoots(Poly,Var)⇒Liste

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

cPolyRoots(KoeffListe)⇒Liste

Die erste Syntax **cPolyRoots(Poly,Var)** gibt eine Liste mit komplexen Wurzeln des Polynoms *Poly* bezüglich der Variablen *Var* zurück.

Poly muss dabei ein Polynom in entwickelter Form in einer Variablen sein. Verwenden Sie keine nicht-entwickelten Formen wie z. B. $y^2 \cdot y + 1$ oder $x \cdot x + 2 \cdot x + 1$

Die zweite Syntax **cPolyRoots(KoeffListe)** liefert eine Liste mit komplexen Wurzeln für die Koeffizienten in *KoeffListe*.

Hinweis: Siehe auch **polyRoots()**, Seite 124.

crossP() (Kreuzprodukt)

crossP(Liste1, Liste2)⇒Liste

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

Gibt das Kreuzprodukt von *Liste1* und *Liste2* als Liste zurück.

Liste1 und *Liste2* müssen die gleiche Dimension besitzen, die entweder 2 oder 3 sein muss.

crossP(Vektor1, Vektor2)⇒Vektor

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

Gibt einen Zeilen- oder Spaltenvektor zurück (je nach den Argumenten), der das Kreuzprodukt von *Vektor1* und *Vektor2* ist.

Entweder müssen *Vektor1* und *Vektor2* beide Zeilenvektoren oder beide Spaltenvektoren sein. Beide Vektoren müssen die gleiche Dimension besitzen, die entweder 2 oder 3 sein muss.

csc() (Kosekans)

csc(Wert1)⇒Wert

Im Grad-Modus:

csc(Liste1)⇒Liste

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

Gibt den Kosekans von *Wert1* oder eine Liste der Konsekans aller Elemente in *Liste1* zurück.

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

csc⁻¹() (Inverser Kosekans)**csc⁻¹(Wert1) ⇒ Wert**

Im Grad-Modus:

csc⁻¹(Liste1) ⇒ Listecsc⁻¹(1)

90.

Gibt entweder den Winkel, dessen Kosekans *Wert1* entspricht, oder eine Liste der inversen Kosekans aller Elemente in *Liste1* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccsc (...)** eintippen.

Im Neugrad-Modus:

csc⁻¹(1)

100.

Im Bogenmaß-Modus:

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

csch() (Kosekans hyperbolicus)**csch(Wert1) ⇒ Wert**

$$\text{csch}(3) \quad 0.099822$$

csch(Liste1) ⇒ Liste

$$\text{csch}(\{1, 2, 1, 4\}) \quad \{0.850918, 0.248641, 0.036644\}$$

Gibt den hyperbolischen Kosekans von *Wert1* oder eine Liste der hyperbolischen Kosekans aller Elemente in *Liste1* zurück.

csch⁻¹() (Inverser Kosekans hyperbolicus)**csch⁻¹(Wert1) ⇒ Wert**

$$\text{csch}^{-1}(1) \quad 0.881374$$

csch⁻¹(Liste1) ⇒ Liste

$$\text{csch}^{-1}(\{1, 2, 1, 3\}) \quad \{0.881374, 0.459815, 0.32745\}$$

csch⁻¹⁽⁾ (Inverser Kosekans hyperbolicus)

Katalog > 

Gibt den inversen hyperbolischen Kosekans von *Wert1* oder eine Liste der inversen hyperbolischen Kosekans aller Elemente in *Liste1* zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccsch (...)** eintippen.

CubicReg (Kubische Regression)

Katalog > 

CubicReg *X, Y[, [Häuf] [, Kategorie, Mit]]*

Berechnet die kubische polynomiale Regression $y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt *X* und *Y* an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

cumulativeSum() (kumulierteSumme)

Katalog > 

cumulativeSum(Liste1)⇒Liste

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

Gibt eine Liste der kumulierten Summen der Elemente aus *Liste1* zurück, wobei bei Element 1 begonnen wird.

cumulativeSum(Matrix1)⇒Matrix

Gibt eine Matrix der kumulierten Summen der Elemente aus *Matrix1* zurück. Jedes Element ist die kumulierte Summe der Spalte von oben nach unten.

Ein leeres (ungültiges) Element in *Liste1* oder *Matrix1* erzeugt ein ungültiges Element in der resultierenden Liste oder Matrix. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

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

cumulativeSum(*m1*)

$$\begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 12 \end{bmatrix}$$

Cycle (Zyklus)

Katalog > 

Cycle (Zyklus)

Übergibt die Programmsteuerung sofort an die nächste Wiederholung der aktuellen Schleife (**For**, **While** oder **Loop**).

Funktionslisting, das die ganzen Zahlen von 1 bis 100 summiert und dabei 50 überspringt.

Cycle (Zyklus)

Katalog >

Cycle ist außerhalb dieser drei Schleifenstrukturen (**For**, **While** oder **Loop**) nicht zulässig.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

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

Done

g()	5000
-----	------

►Cylind (Zylindervektor)

Katalog >

Vektor ►Cylind

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Cylind** eintippen.

[2 2 3]►Cylind	[2.82843 ∠ 0.785398 3.]
----------------	-------------------------

Zeigt den Zeilen- oder Spaltenvektor in Zylinderkoordinaten [r,∠θ, z] an.

Vektor muss genau drei Elemente besitzen. Er kann entweder ein Zeilen- oder Spaltenvektor sein.

D

dbd()

Katalog >

dbd(Datum1,Datum2)⇒Wert

Zählt die tatsächlichen Tage und gibt die Anzahl der Tage zwischen *Datum1* und *Datum2* zurück.

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

Datum1 und *Datum2* können Zahlen oder Zahlenlisten innerhalb des Datumsbereichs des Standardkalenders sein. Wenn sowohl *Datum1* als auch *Datum2* Listen sind, müssen sie dieselbe Länge haben.

Datum1 und *Datum2* müssen innerhalb der Jahre 1950 und 2049 liegen.

Sie können Datumseingaben in zwei Formaten vornehmen. Die Datumsformate unterscheiden sich in der Anordnung der Dezimalstellen.

MM.TTJJ (üblicherweise in den USA verwendetes Format)

TTMM.JJ (üblicherweise in Europa verwendetes Format)

►DD (Dezimalwinkel)

Zahl ►DD⇒Wert

Liste1 ►DD⇒Liste

Matrix1 ►DD⇒Matrix

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>DD eintippen.

Gibt das Dezimaläquivalent des Arguments zurück. Das Argument ist eine Zahl, eine Liste oder eine Matrix, die gemäß der Moduseinstellung als Neugrad, Bogenmaß oder Grad interpretiert wird.

Im Grad-Modus:

{1.5°}►DD	1.5°
{45°22'14.3"}►DD	45.3706°
{ {45°22'14.3", 60°0'0"} }►DD	{45.3706°, 60°}

Im Neugrad-Modus:

1►DD	90°
------	-----

Im Bogenmaß-Modus:

{1.5}►DD	85.9437°
----------	----------

►Decimal (Dezimal)

Wert1 ►Decimal⇒Wert

Liste1 ►Decimal⇒Wert

Matrix1 ►Decimal⇒Wert

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Decimal eintippen.

Zeigt das Argument in Dezimalform an. Dieser Operator kann nur am Ende der Eingabezeile verwendet werden.

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

Definie**Define Var = Expression****Define Function(Param1, Param2, ...) = Expression**Definiert die Variable *Var* oder die benutzerdefinierte Funktion *Function*.

Parameter wie z.B. *Param1* enthalten Platzhalter zur Übergabe von Argumenten an die Funktion. Beim Aufrufen benutzerdefinierter Funktionen müssen Sie Argumente angeben (z.B. Werte oder Variablen), die zu den Parametern passen. Beim Aufruf wertet die Funktion *Ausdruck (Expression)* unter Verwendung der übergebenen Parameter aus.

Var und *Funktion (Function)* dürfen nicht der Name einer Systemvariablen oder einer integrierten Funktion / eines integrierten Befehls sein.

Hinweis: Diese Form von **Definiere (Define)** ist gleichwertig mit der Ausführung folgenden Ausdrucks: *expression* → *Function(Param1,Param2)*.

Define Function(Param1, Param2, ...) = Func Block EndFunc
Define Program(Param1, Param2, ...) = Prgm Block EndPrgm

In dieser Form kann die benutzerdefinierte Funktion bzw. das benutzerdefinierte Programm einen Block mit mehreren Anweisungen ausführen.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen in separaten Zeilen sein. *Block* kann auch Ausdrücke und Anweisungen enthalten (wie **If**, **Then**, **Else** und **For**).

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
------------	---

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Hinweis: Siehe auch **Definiere LibPriv (Define LibPriv)**, Seite 41, und **Definiere LibPub (Define LibPub)**, Seite 41.

Define g(x,y)=Prgm

```
If x>y Then
Disp x," greater than ",y
Else
Disp x," not greater than ",y
EndIf
EndPrgm
```

Done

g(3,-7)

3 greater than -7

Done

Definiere LibPriv (Define LibPriv)

Define LibPriv *Var = Expression*

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

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

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

Funktioniert wie **Define**, definiert jedoch eine Variable, eine Funktion oder ein Programm für eine private Bibliothek. Private Funktionen und Programme werden im Katalog nicht angezeigt.

Hinweis: Siehe auch **Definiere (Define)**, Seite 40, und **Definiere LibPub (Define LibPub)**, Seite 41.

Definiere LibPub (Define LibPub)

Define LibPub *Var = Expression*

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

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

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

Funktioniert wie **Definiere (Define)**, definiert jedoch eine Variable, eine Funktion oder ein Programm für eine öffentliche Bibliothek. Öffentliche Funktionen und Programme werden im Katalog angezeigt, nachdem die Bibliothek gespeichert und aktualisiert wurde.

Hinweis: Siehe auch **Definiere (Define)**, Seite 40, und **Definiere LibPriv (Define LibPriv)**, Seite 41.

deltaList()Siehe Δ List(), Seite 91.**DelVar****DelVar** Var1[, Var2] [, Var3] ...

2 → a 2

DelVar Var.(a+2)² 16

Löscht die angegebene Variable oder Variablengruppe im Speicher.

DelVar a Done

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

Wenn eine oder mehrere Variablen gesperrt sind, wird bei diesem Befehl eine Fehlermeldung angezeigt und es werden nur die nicht gesperrten Variablen gelöscht. Siehe **unLock**, Seite 182.

DelVar

Katalog >

DelVar *Var.* löscht alle Mitglieder der Variablengruppe *Var.* (wie die Statistikergebnisse *stat.nn* oder Variablen, die mit der Funktion **LibShortcut()** erstellt wurden). Der Punkt (.) in dieser Form des Befehls **DelVar** begrenzt ihn auf das Löschen einer Variablengruppe; die einfache Variable *Var* ist nicht davon betroffen.

<i>aa.a:=45</i>	45
<i>aa.b:=5.67</i>	5.67
<i>aa.c:=78.9</i>	78.9
<i>getVarInfo()</i>	$\begin{bmatrix} aa.a & "NUM" & "[1]" \\ aa.b & "NUM" & "[1]" \\ aa.c & "NUM" & "[1]" \end{bmatrix}$
DelVar <i>aa.</i>	<i>Done</i>
getVarInfo()	"NONE"

delVoid()

Katalog >

delVoid(*Liste1*) \Rightarrow *Liste*

Gibt eine Liste mit dem Inhalt von *Liste1* aus, wobei alle leeren (ungültigen) Elemente entfernt sind.

Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

<i>delVoid({1,void,3})</i>	{1,3}
----------------------------	-------

det() (Matrixdeterminante)

Katalog >

det(*Quadratmatrix*[,
Toleranz]) \Rightarrow *Ausdruck*

Gibt die Determinante von *Quadratmatrix* zurück.

Jedes Matrixelement wird wahlweise als 0 behandelt, wenn sein Absolutwert kleiner als *Toleranz* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Toleranz* ignoriert.

<i>det</i> ($\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$)	-2
$\begin{bmatrix} 1.\text{e}20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow mat1$	$\begin{bmatrix} 1.\text{e}20 & 1 \\ 0 & 1 \end{bmatrix}$
<i>det</i> (<i>mat1</i>)	0
<i>det</i> (<i>mat1</i> ,1)	1.e20

- Wenn Sie **ctrl** **enter** verwenden oder den Modus **Autom.** oder **Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Toleranz* weggelassen oder nicht verwendet, so wird die Standardtoleranz

det() (Matrixdeterminante)**Katalog >**

folgendermaßen berechnet:

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

diag() (Matrixdiagonale)**Katalog >****diag(Liste)⇒Matrix****diag([2 4 6])**

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

diag(Zeilensmatrix)⇒Matrix**diag(Spaltenmatrix)⇒Matrix**

Gibt eine Matrix mit den Werten der Argumentliste oder der Matrix in der Hauptdiagonalen zurück.

diag(Quadratmatrix)⇒Zeilensmatrix

Gibt eine Zeilensmatrix zurück, die die Elemente der Hauptdiagonalen von Quadratmatrix enthält.

$$\begin{array}{|c c c|} \hline & [4 & 6 & 8] & [4 & 6 & 8] \\ & [1 & 2 & 3] & [1 & 2 & 3] \\ & [5 & 7 & 9] & [5 & 7 & 9] \\ \hline \text{diag}(Ans) & & [4 & 2 & 9] \\ \hline \end{array}$$

Quadratmatrix muss eine quadratische Matrix sein.

dim() (Dimension)**Katalog >****dim(Liste)⇒Ganzzahl****dim({0,1,2})****3**

Gibt die Dimension von Liste zurück.

dim(Matrix)⇒Liste

Gibt die Dimensionen von Matrix als Liste mit zwei Elementen zurück {Zeilen, Spalten}.

$$\dim \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{bmatrix} \quad \{3,2\}$$

dim(String)⇒Ganzzahl**dim("Hello")****5**

Gibt die Anzahl der in der Zeichenkette String enthaltenen Zeichen zurück.

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

Disp (Zeige)

Katalog >

Disp AusdruckOderString1 [, AusdruckOderString2] ...

Zeigt die Argumente im *Calculator* Protokoll an. Die Argumente werden hintereinander angezeigt, dabei werden Leerzeichen zur Trennung verwendet.

Dies ist vor allem bei Programmen und Funktionen nützlich, um die Anzeige von Zwischenberechnungen zu gewährleisten.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define chars(start,end)=Prgm

For i,start,end
Disp i," ",char(i)
EndFor
EndPrgm

Done

chars(240,243)

240 ð

241 ñ

242 ö

243 ö

Done

DispAt

Katalog >

DispAt int,Term1 [,Term2 ...] ...

Mit **DispAt** können Sie die Zeile festlegen, in der der angegebene Term oder die angegebene Zeichenkette auf dem Bildschirm angezeigt wird.

Die Zeilennummer kann als Term angegeben werden.

Beachten Sie, dass die Zeilennummer nicht für den gesamten Bildschirm gedacht ist, sondern für den Bereich unmittelbar nach dem Befehl/Programm.

Dieser Befehl ermöglicht die dashboardähnliche Ausgabe von Programmen, bei denen der Wert eines Terms oder von einer Sensormessung in der gleichen Zeile aktualisiert wird.

DispAt und **Disp** können im gleichen Programm verwendet werden.

DispAt

Beispiel

```
Define dispat_demo()
Prgm
For n,1,5
DispAt n,"Line ",n
EndFor
EndPrgm
```

Line 1
Line 2
Line 3
Line 4
Line 5

Done

```
"dispat_demo" stored si
Define dispat_demo()
Prgm
For n,1,5
DispAt 3,"Line ",n
EndFor
EndPrgm
```

Line 5

Done

Hinweis: Die maximale Anzahl ist auf 8 eingestellt, da diese Zahl einem Bildschirm voller Zeilen auf dem Handheld-Bildschirm entspricht – soweit die Zeilen über keine mathematischen 2D-Ausdrücke verfügen. Die genaue Anzahl der Zeilen hängt vom Inhalt der angezeigten Informationen ab.

Erläuternde Beispiele:

Define z()=	Beenden von
Prgm	z()
For n,1,3	Iteration 1: Zeile 1: N:1 Zeile 2: Hallo
DispAt 1,,N: „n	
Disp „Hallo“	
EndFor	Iteration 2: Zeile 1: N:2 Zeile 2: Hallo
EndPrgm	Zeile 3: Hallo
	Iteration 3: Zeile 1: N:3 Zeile 2: Hallo Zeile 3: Hallo Zeile 4: Hallo
Define z1()=	z1()
Prgm	Zeile 1: N:3
For n,1,3	Zeile 2: Hallo
DispAt 1,,N: „n	Zeile 3: Hallo
EndFor	Zeile 4: Hallo
	Zeile 5: Hallo
For n,1,4	
Disp „Hallo“	
EndFor	
EndPrgm	

Fehlermeldungen:

Fehlermeldung	Beschreibung
DispAt Zeilennummer muss zwischen 1 und 8 liegen	Term bewertet die Zeilennummer außerhalb des Bereichs 1-8 (inklusiv)
Zu wenig Argumente	Der Funktion oder dem Befehl fehlen ein oder mehr Argumente.
Keine Argumente	Entspricht dem aktuellen Dialog 'Syntaxfehler'
Zu viele Argumente	Argument eingrenzen. Gleicher Fehler

Fehlermeldung	Beschreibung
Ungültiger Datentyp	Erstes Argument muss eine Zahl sein.
Ungültig: DispAt ungültig	„Hallo Welt“ Datentypfehler für die Lücke wird verworfen (falls die Rückmeldung definiert ist)

►DMS (GMS)

Katalog >

Zahl ►DMS

Im Grad-Modus:

Liste ►DMS

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

Matrix ►DMS

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>DMS eintippen.

Interpretiert den Parameter als Winkel und zeigt die entsprechenden GMS-Werte (engl. DMS) an (GGGGGG°MM'SS.ss"). Siehe °, ', " (Seite 212) zur Erläuterung des DMS-Formats (Grad, Minuten, Sekunden).

Hinweis: ►DMS wandelt Bogenmaß in Grad um, wenn es im Bogenmaß-Modus benutzt wird. Folgt auf die Eingabe das Grad-Symbol °, wird keine Umwandlung vorgenommen. Sie können ►DMS nur am Ende einer Eingabezeile benutzen.

dotP() (Skalarprodukt)

Katalog >

dotP(Liste1, Liste2)⇒Ausdruck

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

17

Gibt das Skalarprodukt zweier Listen zurück.

dotP(Vektor1, Vektor2)⇒Ausdruck

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

32

Gibt das Skalarprodukt zweier Vektoren zurück.

Es müssen beide Zeilenvektoren oder beide Spaltenvektoren sein.

e^A()**e^x Taste****e^{A(Wert1)}⇒Wert**

Gibt e hoch Wert1 zurück.

Hinweis: Siehe auch Vorlage **e Exponent**, Seite 2.**Hinweis:** Das Drücken von **e^x** zum Anzeigen von e^A ist nicht das gleiche wie das Drücken von **E** auf der Tastatur.

Sie können eine komplexe Zahl in der polaren Form $r e^{i\theta}$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

e^{A(Liste1)}⇒Liste**e¹**

2.71828

e^{3^2}

8103.08

Gibt e hoch jedes Element der Liste1 zurück.

e^{A(Quadratmatrix1)}⇒Quadratmatrix**e^{1,1.,0.5}** {2.71828,2.71828,1.64872}

Ergebnis den Matrix-Exponenten von Quadratmatrix1. Dies ist nicht gleichbedeutend mit der Berechnung von e hoch jedes Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

eff()**Katalog >** **eff(Nominalzinssatz, CpY)⇒Wert****eff(5.75,12)**

5.90398

Finanzfunktion, die den Nominalzinssatz Nominalzinssatz in einen jährlichen Effektivsatz konvertiert, wobei CpY als die Anzahl der Verzinsungsperioden pro Jahr gegeben ist.

Nominalzinssatz muss eine reelle Zahl sein und CpY muss eine reelle Zahl > 0 sein.

Hinweis: Siehe auch **nom()**, Seite 113.

eigVc() (Eigenvektor)

eigVc(Quadratmatrix)⇒Matrix

Ergibt eine Matrix, welche die Eigenvektoren für eine reelle oder komplexe *Quadratmatrix* enthält, wobei jede Spalte des Ergebnisses zu einem Eigenwert gehört. Beachten Sie, dass ein Eigenvektor nicht eindeutig ist; er kann durch einen konstanten Faktor skaliert werden. Die Eigenvektoren sind normiert, d. h. wenn $V = [x_1, x_2, \dots, x_n]$, dann:

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

Quadratmatrix wird zunächst mit Ähnlichkeitstransformationen bearbeitet, bis die Zeilen- und Spaltennormen so nahe wie möglich bei demselben Wert liegen. Die *Quadratmatrix* wird dann auf die obere Hessenberg-Form reduziert, und die Eigenvektoren werden mit einer Schur-Faktorisierung berechnet.

Im Komplex-Formatmodus "kartesisch":

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

eigVc(m1)

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

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleright und \blacktriangleright , um den Cursor zu bewegen.

eigVi() (Eigenwert)

eigVi(Quadratmatrix)⇒Liste

Ergibt eine Liste von Eigenwerten einer reellen oder komplexen *Quadratmatrix*.

Quadratmatrix wird zunächst mit Ähnlichkeitstransformationen bearbeitet, bis die Zeilen- und Spaltennormen so nahe wie möglich bei demselben Wert liegen. Die *Quadratmatrix* wird dann auf die obere Hessenberg-Form reduziert, und die Eigenwerte werden aus der oberen Hessenberg-Matrix berechnet.

Im Komplex-Formatmodus "kartesisch":

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

eigVi(m1)

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

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleright und \blacktriangleright , um den Cursor zu bewegen.

Else

Siehe If, Seite 75.

Elsef**If Boolescher Ausdr1 Then** **Block1****Elsef Boolescher Ausdr2 Then** **Block2**

:

Elsef Boolescher AusdrN Then **BlockN****EndIf**

:

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define $g(x) = \text{Func}$ If $x \leq -5$ Then

Return 5

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

Return 3

EndIf

EndFunc

*Done***EndFor****Siehe For, Seite 59.****EndFunc****Siehe Func, Seite 63.****EndIf****Siehe If, Seite 75.****EndLoop****Siehe Loop, Seite 98.****EndWhile****Siehe While, Seite 185.****EndPrgm****Siehe Prgm, Seite 126.**

euler ()**Katalog > **

euler(*Ausdr, Var, abhVar, {Var0, VarMax}, abhVar0, VarSchritt [, eulerSchritt]*) \Rightarrow Matrix

euler(*AusdrSystem, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt [, eulerSchritt]*) \Rightarrow Matrix

euler(*AusdrListe, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt [, eulerSchritt]*) \Rightarrow Matrix

Verwendet die Euler-Methode zum Lösen des Systems

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

mit $abhVar(Var0)=abhVar0$ auf dem Intervall $[Var0, VarMax]$. Gibt eine Matrix zurück, deren erste Zeile die Ausgabewerte von *Var* definiert und deren zweite Zeile den Wert der ersten Lösungskomponente an den entsprechenden *Var*-Werten definiert usw.

Ausdr ist die rechte Seite, die die gewöhnliche Differentialgleichung (ODE) definiert.

AusdrSystem ist das System rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

AusdrListe ist eine Liste rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

Var ist die unabhängige Variable.

ListeAbhVar ist eine Liste abhängiger Variablen.

Differentialgleichung:

$$y' = 0.001 \cdot y \cdot (100 - y) \text{ und } y(0) = 10$$

$$\begin{aligned} \text{euler}\left(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9 & 11.8712 & 12.9174 & 14.042 \end{bmatrix} \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

Gleichungssystem:

$$\begin{cases} y_1' = -y_1 + 0.1 \cdot y_1 \cdot y_2 \\ y_2' = 3 \cdot y_2 - y_1 \cdot y_2 \end{cases}$$

$$\text{mit } y_1(0) = 2 \text{ und } y_2(0) = 5$$

$$\begin{aligned} \text{euler}\left(\begin{cases} -y_1 + 0.1 \cdot y_1 \cdot y_2 \\ 3 \cdot y_2 - y_1 \cdot y_2 \end{cases}, t, \{y_1, y_2\}, \{0, 5\}, \{2, 5\}, 1\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix} \end{aligned}$$

$\{Var0, VarMax\}$ ist eine Liste mit zwei Elementen, die die Funktion anweist, von $Var0$ zu $VarMax$ zu integrieren.

ListeAbhVar0 ist eine Liste von Anfangswerten für abhängige Variablen.

VarSchritt ist eine Zahl ungleich Null, sodass $\text{sign}(VarSchritt) = \text{sign}(VarMax - Var0)$ und Lösungen an $Var0 + i \cdot VarSchritt$ für alle $i=0,1,2,\dots$ zurückgegeben werden, sodass $Var0 + i \cdot VarSchritt$ in $[var0,VarMax]$ ist (möglicherweise gibt es keinen Lösungswert an $VarMax$).

eulerSchritt ist eine positive ganze Zahl (standardmäßig 1), welche die Anzahl der Euler-Schritte zwischen Ausgabewerten bestimmt. Die tatsächliche von der Euler-Methode verwendete Schrittgröße ist $VarSchritt/eulerSchritt$.

eval ()

Hub-Menü

eval(Expr) \Rightarrow Zeichenfolge

eval() ist nur im TI-Innovator™ Hub Befehlsargument von Programmierbefehlen **Get**, **GetStr** und **Send** gültig. Die Software wertet den Ausdruck *Expr* aus und ersetzt die Anweisung **eval()** mit dem Ergebnis als Zeichenfolge.

Das Argument *Expr* muss zu einer reellen Zahl vereinfachbar sein.

Stellen Sie das blaue Element von RGB LED auf halbe Intensität ein.

<i>lum:=127</i>	127
Send "SET COLOR,BLUE eval(lum)"	<i>Done</i>

Setzen Sie das blaue Element auf AUS zurück.

Send "SET COLOR,BLUE OFF"	<i>Done</i>
---------------------------	-------------

Argument **eval()** muss zu einer reellen Zahl vereinfachbar sein.

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

Programm zum Einblenden des roten Elements

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

Führen Sie das Programm aus.

fadein()	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)"	<i>Done</i>
<i>iostr.SendAns "SET COLOR.BLUE ON TIME 2"</i>	<i>Done</i>

Obwohl **eval()** sein Ergebnis nicht anzeigt, können Sie die resultierende Hub-Zeichenfolge nach Ausführen des Befehls durch Prüfung einer beliebigen der folgenden speziellen Variablen anzeigen.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

Hinweis: Siehe auch **Get** (Seite 65), **GetStr** (Seite 73) und **Send** (Seite 148).

Exit (Abbruch)

Katalog >

Exit (Abbruch)

Beendet den aktuellen **For**, **While**, oder **Loop** Block.

Exit ist außerhalb dieser drei Schleifenstrukturen (**For**, **While** oder **Loop**) nicht zulässig.

Hinweis zum Eingeben des Beispiele:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Funktionslisting:

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

<i>g()</i>	21
------------	----

exp() (e hoch x)

Taste

exp(Wert1)⇒Wert

Gibt e hoch Wert1 zurück.

Hinweis: Siehe auch Vorlage e Exponent, Seite 2.

Sie können eine komplexe Zahl in der polaren Form $r \text{ei} \theta$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

exp(Liste1)⇒Liste

Gibt e hoch jedes Element der Liste1 zurück.

exp(Quadratmatrix1)⇒Quadratmatrix

Ergibt den Matrix-Exponenten von Quadratmatrix1. Dies ist nicht gleichbedeutend mit der Berechnung von e hoch jedes Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

e ¹	2.71828
e ³ ²	8103.08

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

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

expr() (String in Ausdruck)

Katalog >

expr(String)⇒Ausdruck

Gibt die in String enthaltene Zeichenkette als Ausdruck zurück und führt diesen sofort aus.

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

ExpReg (Exponentielle Regression)

Katalog >

ExpReg X, Y [, [Häuf][, Kategorie, Mit]]

Berechnet die exponentielle Regression $y = a \cdot (b)^x$ auf Listen X und Y mit der Häufigkeit $Häuf$. Eine Zusammenfassung der Ergebnisse wird in den Variablen `stat.results` gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt X und Y an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 235).

Ausgabevariable	Beschreibung
<code>stat.RegEqn</code>	Regressionsgleichung: $a \cdot (b)^x$
<code>stat.a</code> , <code>stat.b</code>	Regressionskoeffizienten
<code>stat.r²</code>	Koeffizient der linearen Bestimmtheit für transformierte Daten
<code>stat.r</code>	Korrelationskoeffizient für transformierte Daten (x , $\ln(y)$)
<code>stat.Resid</code>	Mit dem exponentiellen Modell verknüpfte Residuen
<code>stat.ResidTrans</code>	Residuum für die lineare Anpassung der transformierten Daten.
<code>stat.XReg</code>	Liste der Datenpunkte in der modifizierten X List, die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde

AusgabevARIABLE	Beschreibung
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

F

factor() (Faktorisiere)

Katalog > 

factor(*RationaleZahl*) ergibt die rationale Zahl in Primfaktoren zerlegt. Bei zusammengesetzten Zahlen nimmt die Berechnungsdauer exponentiell mit der Anzahl an Stellen im zweitgrößten Faktor zu. Das Faktorisieren einer 30-stelligen ganzen Zahl kann beispielsweise länger als einen Tag dauern und das Faktorisieren einer 100-stelligen Zahl mehr als ein Jahrhundert.

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

So halten Sie eine Berechnung manuell an:

- **Handheld:** Halten Sie die Taste  **on** gedrückt und drücken Sie mehrmals **[enter]**.
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Möchten Sie hingegen lediglich feststellen, ob es sich bei einer Zahl um eine Primzahl handelt, verwenden Sie **isPrime()**. Dieser Vorgang ist wesentlich schneller, insbesondere dann, wenn *RationaleZahl* keine Primzahl ist und der zweitgrößte Faktor mehr als fünf Stellen aufweist.

FCdf()

Katalog >

FCdf

(

UntGrenze

,

ObGrenze

,FreiGradZähler,FreiGradNenner)⇒Zahl,
wenn UntGrenze und ObGrenze Zahlen
sind, Liste, wenn UntGrenze und
ObGrenze Listen sind

FCdf

(

UntGrenze

,

ObGrenze

,FreiGradZähler,FreiGradNenner)⇒Zahl,
wenn UntGrenze und ObGrenze Zahlen
sind, Liste, wenn UntGrenze und
ObGrenze Listen sind

Berechnet die F

Verteilungswahrscheinlichkeit zwischen
UntereGrenze und ObereGrenze für die
angegebenen FreiGradZähler
(Freiheitsgrade) und FreiGradNenner.

Für $P(X \leq \text{ObereGrenze})$, UntGrenze =0
setzen.

Fill (Füllen)

Katalog >

Fill Zahl, MatrixVar⇒Matrix

Ersetzt jedes Element in der Variablen
MatrixVar durch Zahl.

MatrixVar muss bereits vorhanden sein.

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

Fill Zahl, ListeVar⇒Liste

Ersetzt jedes Element in der Variablen
ListeVar durch Zahl.

ListeVar muss bereits vorhanden sein.

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

FiveNumSummary

Katalog >

FiveNumSummary X[,[Häuf]
,Kategorie,Mit]]

Bietet eine gekürzte Version der Statistik mit 1 Variablen auf Liste X . Eine Zusammenfassung der Ergebnisse wird in der Variablen `stat.results` gespeichert. (Seite 162.)

X stellt eine Liste mit den Daten dar.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden X -Wert an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen X , *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Ausgabevariable	Beschreibung
<code>stat.MinX</code>	Minimum der x-Werte
<code>stat.Q1X</code>	1. Quartil von x
<code>stat.MedianX</code>	Median von x
<code>stat.Q3X</code>	3. Quartil von x
<code>stat.MaxX</code>	Maximum der x-Werte

floor() (Untergrenze)

Katalog > 

`floor(Wert1)` \Rightarrow Ganzzahl

`floor(-2.14)`

-3.

Gibt die größte ganze Zahl zurück, die \leq dem Argument ist. Diese Funktion ist identisch mit `int()`.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

floor(Liste I)⇒Liste

floor(Matrix I)⇒Matrix

Für jedes Element einer Liste oder Matrix wird die größte ganze Zahl, die kleiner oder gleich dem Element ist, zurückgegeben.

Hinweis: Siehe auch **ceiling()** und **int()**.

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

$$\text{floor}\left(\begin{pmatrix} 1.2 & 3.4 \\ 2.5 & 4.8 \end{pmatrix}\right) = \begin{pmatrix} 1, & 3 \\ 2, & 4 \end{pmatrix}$$

For

For Var, Von, Bis [, Schritt]

Block

EndFor

Führt die in *Block* befindlichen Anweisungen für jeden Wert von *Var* zwischen *Von* und *Bis* aus, wobei der Wert bei jedem Durchlauf um *Schritt* inkrementiert wird.

Var darf keine Systemvariable sein.

Schritt kann positiv oder negativ sein. Der Standardwert ist 1.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen sein, die durch ":" getrennt sind.

Hinweis zum Eingeben des Beispieles:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define g()=Func Done

Local tempsum,step,i

0→tempsum

1→step

For i,1,100,step

tempsum+i→tempsum

EndFor

EndFunc

g() 5050

format() (Format)**format(*Wert*[, *FormatString*]) \Rightarrow *String***

Gibt *Wert* als Zeichenkette im Format der Formatvorlage zurück.

FormatString ist eine Zeichenkette und muss diese Form besitzen: "F[n]", "S[n]", "E[n]", "G[n][c]", wobei [] optionale Teile bedeutet.

F[n]: Festes Format. n ist die Anzahl der angezeigten Nachkommastellen (nach dem Dezimalpunkt).

S[n]: Wissenschaftliches Format. n ist die Anzahl der angezeigten Nachkommastellen (nach dem Dezimalpunkt).

E[n]: Technisches Format. n ist die Anzahl der Stellen, die auf die erste signifikante Ziffer folgen. Der Exponent wird auf ein Vielfaches von 3 gesetzt, und der Dezimalpunkt wird um null, eine oder zwei Stellen nach rechts verschoben.

G[n][c]: Wie Festes Format, unterteilt jedoch auch die Stellen links des Dezimaltrennzeichens in Dreiergruppen. c ist das Gruppentrennzeichen und ist auf "Komma" voreingestellt. Wenn c auf "Punkt" gesetzt wird, wird das Dezimaltrennzeichen zum Komma.

[Rc]: Jeder der vorstehenden Formateinstellungen kann als Suffix das Flag Rc nachgestellt werden, wobei c ein einzelnes Zeichen ist, das den Dezimalpunkt ersetzt.

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

fPart() (Funktionsteil)**fPart(*AusdrI*) \Rightarrow *Ausdruck***

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

fPart(*ListeI*) \Rightarrow *Liste***fPart(*MatrixI*) \Rightarrow *Matrix***

Gibt den Bruchanteil des Arguments zurück.

Bei einer Liste bzw. Matrix werden die Bruchanteile aller Elemente zurückgegeben.

fPart() (Funktionsteil)

Katalog >

Das Argument kann eine reelle oder eine komplexe Zahl sein.

F Pdf()

Katalog >

F Pdf

(

XWert

,FreiGradZähler,FreiGradNenner)⇒Zahl,

wenn XWert eine Zahl ist, Liste, wenn

XWert eine Liste ist

F Pdf

(

XWert

,FreiGradZähler,FreiGradNenner)⇒Zahl,

wenn XWert eine Zahl ist, Liste, wenn

XWert eine Liste ist

Berechnet die F

Verteilungswahrscheinlichkeit bei XWert für

die angegebenen FreiGradZähler

(Freiheitsgrade) und FreiGradNenner.

freqTable►list()

Katalog >

freqTable►list

(Liste1,HäufGanzzahlListe)⇒Liste

Gibt eine Liste zurück, die die Elemente von Liste1 erweitert gemäß den Häufigkeiten in HäufGanzzahlListe enthält. Diese Funktion kann zum Erstellen einer Häufigkeitstabelle für die Applikation 'Data & Statistics' verwendet werden.

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

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

Liste1 kann eine beliebige gültige Liste sein.

HäufGanzzahlListe muss die gleiche Dimension wie Liste1 haben und darf nur nicht-negative Ganzzahlelemente enthalten. Jedes Element gibt an, wie oft das entsprechende Liste1-Element in der Ergebnisliste wiederholt wird. Der Wert 0 schließt das entsprechende Liste1-Element aus.

freqTable@>list()

Katalog >

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie `freqTable@>list(...)` eintippen

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

frequency() (Häufigkeit)

Katalog >

frequency(Liste1,binsListe)⇒Liste

Gibt eine Liste zurück, die die Zähler der Elemente in *Liste1* enthält. Die Zähler basieren auf Bereichen (bins), die Sie in *binsListe* definieren.

Wenn *binsListe* { $b(1)$, $b(2)$, ..., $b(n)$ } ist, sind die festgelegten Bereiche $\{? \leq b(1), b(1) < ? \leq b(2), \dots, b(n-1) < ? \leq b(n), b(n) > ?\}$. Die Ergebnisliste enthält ein Element mehr als die *binsListe*.

Jedes Element des Ergebnisses entspricht der Anzahl der Elemente aus *Liste1*, die im Bereich dieser bins liegen. Ausgedrückt in Form der **countIf()** Funktion ist das Ergebnis $\{ \text{countIf}(\text{Liste}, ? \leq b(1)), \text{countIf}(\text{Liste}, b(1) < ? \leq b(2)), \dots, \text{countIf}(\text{Liste}, b(n-1) < ? \leq b(n)), \text{countIf}(\text{Liste}, b(n) > ?) \}$.

Elemente von *Liste1*, die nicht "in einem bin platziert" werden können, werden ignoriert. Leere (ungültige) Elemente werden ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Innerhalb der Lists & Spreadsheet Applikation können Sie für beide Argumente Zellenbereiche verwenden.

Hinweis: Siehe auch **countIf()**, Seite 33.

<code>datalist:= {1,2,e,3,π,4,5,6,"hello",7}</code>	<code>{1,2,2.71828,3,3.14159,4,5,6,"hello",7}</code>
<code>frequency(datalist,{2.5,4.5})</code>	<code>{2,4,3}</code>

Erklärung des Ergebnisses:

2 Elemente aus *Datenliste (Datalist)* sind ≤ 2.5

4 Elemente aus *Datenliste* sind > 2.5 und ≤ 4.5

3 Elemente aus *Datenliste* sind > 4.5

Das Element "Hallo" ist eine Zeichenfolge und kann nicht in einem der definierten bins platziert werden.

FTest_2Samp (Zwei-Stichproben F-Test)

Katalog >

FTest_2Samp *Liste1*,*Liste2*[,*Häufigkeit1*,
[*Häufigkeit2*[,*Hypoth*]]]

FTest_2Samp (Zwei-Stichproben F-Test)

Katalog >

FTest_2Samp *Liste1, Liste2[, Häufigkeit1 [, Häufigkeit2[, Hypoth]]]*

(Datenlisteneingabe)

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

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

(Zusammenfassende statistische Eingabe)

Führt einen F -Test mit zwei Stichproben durch. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Für $H_a: \sigma_1 > \sigma_2$ setzen Sie *Hypoth>0*

Für $H_a: \sigma_1 \neq \sigma_2$ (Standard) setzen Sie *Hypoth =0*

Für $H_a: \sigma_1 < \sigma_2$ setzen Sie *Hypoth<0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
Statistik.F	Berechnete Ü Statistik für die Datenfolge
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.dfNumer	Freiheitsgrade des Zählers = n1-1
stat.dfDenom	Freiheitsgrade des Nenners = n2-1
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.x1_bar	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.x2_bar	
stat.n1, stat.n2	Stichprobenumfang

Func

Katalog >

Func
 Block
 EndFunc

Definieren Sie eine stückweise definierte Funktion:

Func

Katalog >

Vorlage zur Erstellung einer benutzerdefinierten Funktion.

Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen ":" voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein. Die Funktion kann die Anweisung **Zurückgeben (Return)** verwenden, um ein bestimmtes Ergebnis zurückzugeben.

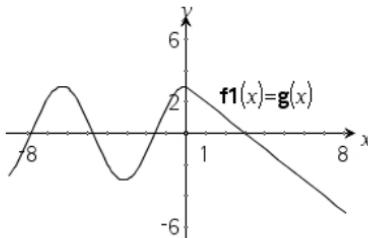
Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define $g(x) = \text{Func}$

Done

```
If  $x < 0$  Then  
    Return  $3 \cdot \cos(x)$   
Else  
    Return  $3 - x$   
EndIf  
EndFunc
```

Ergebnis der graphischen Darstellung $g(x)$



G

gcd() (Größter gemeinsamer Teiler)

Katalog >

gcd(Zahl1, Zahl2)⇒Ausdruck

gcd(18,33)

3

Gibt den größten gemeinsamen Teiler der beiden Argumente zurück. Der **gcd** zweier Brüche ist der **gcd** ihrer Zähler dividiert durch das kleinste gemeinsame Vielfache (**lcm**) ihrer Nenner.

In den Modi Auto oder Approximiert ist der **gcd** von Fließkommabrüchen 1,0.

gcd(Liste1, Liste2)⇒Liste

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

Gibt die größten gemeinsamen Teiler der einander entsprechenden Elemente von *Liste1* und *Liste2* zurück.

gcd(Matrix1, Matrix2)⇒Matrix

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

Gibt die größten gemeinsamen Teiler der einander entsprechenden Elemente von *Matrix1* und *Matrix2* zurück.

geomCdf

$(p, \text{untereGrenze}, \text{obereGrenze}) \Rightarrow \text{Zahl}$,
wenn *untereGrenze* und *obereGrenze*
Zahlen sind, *Liste*, wenn *untereGrenze* und
obereGrenze Listen sind

geomCdf(*p, obereGrenze*) für $P(1 \leq X \leq \text{obereGrenze}) \Rightarrow \text{Zahl}$, wenn *obereGrenze*
eine Zahl ist, *Liste*, wenn *obereGrenze* eine
Liste ist

Berechnet die kumulative geometrische
Wahrscheinlichkeit von *UntereGrenze* bis
ObereGrenze mit der angegebenen
Erfolgswahrscheinlichkeit *p*.

Für $P(X \leq \text{obereGrenze})$ setzen Sie
untereGrenze = 1.

geomPdf(*p, XWert*) $\Rightarrow \text{Zahl}$, wenn *XWert*
eine Zahl ist, *Liste*, wenn *XWert* eine Liste
ist

Berechnet die Wahrscheinlichkeit an einem
XWert, die Anzahl der Einzelversuche, bis
der erste Erfolg eingetreten ist, für die
diskrete geometrische Verteilung mit der
vorgegebenen Erfolgswahrscheinlichkeit *p*.

Get**Hub-Menü**

Get[*EingabeString*,] *Var*[, *statusVar*]

Get[*EingabeString*,] *Fkt*(*arg1*, ...*argn*)
[, *statusVar*]

Programmierbefehl: Ruft einen Wert von
einem verbundenen TI-Innovator™ Hub ab
und weist den Wert der Variablen *var* zu.

Der Wert muss angefordert werden:

- Im Voraus durch einen Befehl
Send "READ ...".
– oder –

Beispiel: Fordern Sie den aktuellen Wert des
integrierten Lichtpegelsensors des Hub an.

Verwenden Sie **Get**, um den Wert abzurufen,
und weisen Sie ihn der Variablen *lightval* zu.

Send "READ BRIGHTNESS"	Done
Get <i>lightval</i>	Done
<i>lightval</i>	0.347922

Betten Sie die Anforderung READ in den
Befehl **Get** ein.

- Durch Einbetten einer Anforderung "**READ ...**" als optionales Argument von *promptString*. Bei dieser Methode können Sie einen einzelnen Befehl verwenden, um den Wert anzufordern und abzurufen.

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

Implizite Vereinfachung findet statt. Zum Beispiel wird eine empfangene Zeichenfolge „123“ als numerischer Wert interpretiert. Um die Zeichenfolge beizubehalten, verwenden Sie **GetStr** statt **Get**.

Wenn Sie das optionale Argument von *statusVar* einbeziehen, wird ihm ein Wert auf Basis des Erfolgs der Operation zugewiesen. Ein Wert von null bedeutet, dass keine Daten empfangen wurden.

In der zweiten Synthax ermöglicht das Argument von *Fkt()* es einem Programm, die empfangene Zeichenfolge als Funktionsdefinition zu speichern. Diese Syntax verhält sich so, als hätte das Programm den folgenden Befehl ausgeführt:

Definiere *Fkt(arg1, ...argn) = empfanger String*

Anschließend kann das Programm die so definierte Funktion *Fkt()* nutzen.

Hinweis: Sie können den Befehl **Get** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

Hinweis: Siehe auch **GetStr**, Seite 73 und **Send**, Seite 148.

getDenom() (Nenner holen)

Katalog >

getDenom(Bruch1)⇒Wert

Transformiert das Argument in einen Ausdruck mit gekürztem gemeinsamem Nenner und gibt dann den Nenner zurück.

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

getKey()

Katalog >

getKey([01]) ⇒ returnType

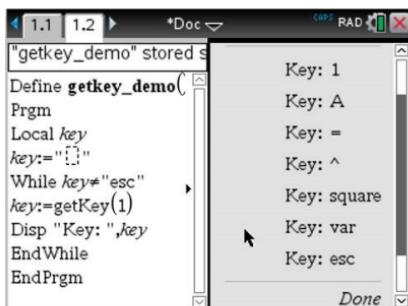
Beschreibung: getKey() – ermöglicht ein TI-Basic-Programm zum Holen von Tastatureingaben – Handheld, Desktop und Emulator auf Desktop.

Beispiel:

- gedrückteTaste := getKey() gibt eine Taste oder eine leere Zeichenkette zurück, wenn keine Taste gedrückt wurde. Dieser Aufruf wird umgehend zurückgegeben.
- gedrückteTaste := getKey(1) wartet bis eine Taste gedrückt wird. Dieser Aufruf pausiert die Ausführung des Programms, bis eine Taste gedrückt wird.

getKey()

Beispiel:



Handhabung von Tastenbefehlungen:

Handheld/Emulatortaste	Desktop	Rückgabewert
Esc	Esc	„Esc“
Touchpad – Oben klicken	–	„nach oben“
Ein	–	„Hauptmenü“
Scratch Apps	–	„Scratchpad“
Touchpad – Linksklick	–	„links“
Touchpad – Mittig klicken	–	„Mittelpunkt“

Handheld/Emulatortaste	Desktop	Rückgabewert
Touchpad – Rechtsklick	–	„rechts“
Dok	–	„Dok“
Tab	Tab	„Tab“
Touchpad – Unten klicken	Abwärtspfeil	„nach unten“
Menü	–	„Menü“
Strg	Strg	keine Rückgabe
Verschieben (Shift)	Verschieben (Shift)	keine Rückgabe
Var	–	„var“
Entf	–	„del“
=	=	"="
Trigonometrie	–	„Trigonometrie“
0 bis 9	0-9	„0“ ... „9“
Vorlagen	–	„Vorlage“
Katalog	–	„cat“
^	^	"^"
X^2	–	„Quadrat“
/ (Divisionstaste)	/	" / "
* (Multiplikationstaste)	*	" * "
e^x	–	„Ausdr“
10^x	–	„10power“
+	+	" + "
-	-	" - "
((" ("
))	") "
.	.	" . "
(-)	–	„-“ (Negativ-Zeichen)
Eingabetaste	Eingabetaste	„Eingabe“

Handheld/Emulatortaste	Desktop	Rückgabewert
Osteuropa	–	„E“ Exponentialform (wissenschaftliche Schreibweise E)
a – z	a-z	Alpha = Buchstabe gedrückt (Kleinschreibung) „a“ – „z“)
Umschalt a-z	Umschalt a-z	Alpha = Buchstabe gedrückt „A“ – „Z“
		Hinweis: Strg-Umschalt ergibt Feststelltaste
?!	–	"?!"
pi	–	„pi“
Flag	–	keine Rückgabe
,	,	„,“
Return	–	„Rückgabe“
Leerzeichen	Leerzeichen	„ “ (Leerzeichen)
Unzugänglich	Tasten für Sonderzeichen wie @,!,& etc.	Das Zeichen wird zurückgegeben
–	Funktionstasten	Kein zurückgegebenes Zeichen
–	Besondere Desktop-Bedientasten	Kein zurückgegebenes Zeichen
Unzugänglich	Sonstige Desktop-Tasten, die nicht auf dem Calculator zur Verfügung stehen, während getKey() auf eine Tastenbetätigung wartet. ({{, };; :; ...})	Gleiches Zeichen wie in Notes (nicht in einem math. Feld)

Hinweis: Es ist wichtig zu beachten, dass das Vorhandensein von `getKey()` in einem Programm die Art und Weise ändert, wie sicher Ereignisse durch das System gehandhabt werden. Einige davon werden unten beschrieben.

Programm beenden und Ereignis handhaben – Auf gleiche Art als sollte der Benutzer das Programm verlassen, indem er die **EIN**-Taste drückt

„Support“ unten bedeutet – System arbeitet wie erwartet – Programm läuft weiter.

Ereignis	Handheld-Gerät	Desktop – TI-Nspire™ Schülersoftware
Schnellumfrage	Programm beenden, Ereignis handhaben	Entspricht dem Handheld (nur TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software)
Verwaltung Remote-Datei (Einschl. Versenden der Datei 'Press-to-Test verlassen' von einem anderen Handheld oder Desktop-Handheld)	Programm beenden, Ereignis handhaben	Entspricht dem Handheld. (nur TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software)
Klasse beenden	Programm beenden, Ereignis handhaben	Support (nur TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software)
Ereignis	Handheld-Gerät	Desktop – TI-Nspire™ Alle Versionen
TI-Innovator™ Hub verbinden/trennen	Support – Kann erfolgreich Befehle an den TI-Innovator™ Hub geben. Nachdem Sie das Programm verlassen haben, arbeitet der TI-Innovator™ Hub noch mit dem Handheld weiter.	Entspricht dem Handheld

getLangInfo()

Katalog > 

getLangInfo()=>Zeichenkette

getLangInfo()

"en"

Gibt eine Zeichenkette zurück, die der Abkürzung der gegenwärtig aktiven Sprache entspricht. Sie können den Befehl zum Beispiel in einem Programm oder einer Funktion zum Bestimmen der aktuellen Sprache verwenden.

Englisch = "en"

Dänisch = "da"

getLangInfo()

Katalog >

Deutsch = "de"

Finnisch = "fi"

Französisch = "fr"

Italienisch = "it"

Holländisch = "nl"

Holländisch (Belgien) = "nl_BE"

Norwegisch = "no"

Portugiesisch = "pt"

Spanisch = "es"

Schwedisch = "sv"

getLockInfo()

Katalog >

getLockInfo(Var)⇒Wert

Gibt den aktuellen Gesperrt/Entsperrt-Status der Variablen *Var* aus.

Wert =0: *Var* ist nicht gesperrt oder ist nicht vorhanden.

Wert =1: *Var* ist gesperrt und kann nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 95, und **unlock**, Seite 182.

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

getMode()

Katalog >

getMode(ModusNameGanzzahl)⇒Wert

getMode(0)⇒Liste

getMode(ModusNameGanzzahl) gibt einen Wert zurück, der die aktuelle Einstellung des Modus *ModusNameGanzzahl* darstellt.

getMode(0) gibt eine Liste mit Zahlenpaaren zurück. Jedes Paar enthält eine Modus-Ganzzahl und eine Einstellungs-Ganzzahl.

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

Eine Auflistung der Modi und ihrer Einstellungen finden Sie in der nachstehenden Tabelle.

Wenn Sie die Einstellungen mit **getMode(0)** → *var* speichern, können Sie **setMode(var)** in einer Funktion oder in einem Programm verwenden, um die Einstellungen nur innerhalb der Ausführung dieser Funktion bzw. dieses Programms vorübergehend wiederherzustellen. Siehe **setMode()**, Seite 151.

Modus Name	Modus Ganzzahl	Einstellen von Ganzzahlen
Angezeigte Ziffern	1	1 =Fließ, 2 =Fließ 1, 3 =Fließ 2, 4 =Fließ 3, 5 =Fließ 4, 6 =Fließ 5, 7 =Fließ 6, 8 =Fließ 7, 9 =Fließ 8, 10 =Fließ 9, 11 =Fließ 10, 12 =Fließ 11, 13 =Fließ 12, 14 =Fix 0, 15 =Fix 1, 16 =Fix 2, 17 =Fix 3, 18 =Fix 4, 19 =Fix 5, 20 =Fix 6, 21 =Fix 7, 22 =Fix 8, 23 =Fix 9, 24 =Fix 10, 25 =Fix 11, 26 =Fix 12
Winkel	2	1 =Bogenmaß, 2 =Grad, 3 =Neugrad
Exponentialformat	3	1 =Normal, 2 =Wissenschaftlich, 3 =Technisch
Reell oder komplex	4	1 =Reell, 2 =Kartesisch, 3 =Polar
Auto oder Approx.	5	1 =Auto, 2 =Approximiert
Vektorformat	6	1 =Kartesisch, 2 =Zylindrisch, 3 =Sphärisch
Basis	7	1 =Dezimal, 2 =Hex, 3 =Binär

getNum() (Zähler holen)**getNum(Bruch1)⇒Wert**

Transformiert das Argument in einen Ausdruck mit gekürztem gemeinsamem Nenner und gibt dann den Zähler zurück.

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

GetStr[*EingabeString*,] *Var*[, *statusVar*]

Zum Beispiel siehe **Get**.

GetStr[*EingabeString*,] *Fkt*(*arg1*, ...*argn*)
[, *statusVar*]

Programmierbefehl: Verhält sich genauso wie der Befehl **Get**, der abgerufene Wert wird aber immer als Zeichenfolge interpretiert. Der Befehl **Get** interpretiert die Antwort hingegen als Ausdruck, es sei denn, sie ist in Anführungszeichen ("") gesetzt.

Hinweis: Siehe auch **Get**, Seite 65 und **Send**, Seite 148.

getType()

Katalog >

getType(*var*) \Rightarrow String

Gibt eine Zeichenkette zurück, die den Datentyp einer Variablen *var* angeibt.

Wenn *var* nicht definiert ist, wird die Zeichenkette „NONE“ zurückgegeben.

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

getVarInfo()

Katalog >

getVarInfo() \Rightarrow Matrix oder String

getVarInfo(BiblioNameString) \Rightarrow Matrix oder String

getVarInfo() gibt eine Informationsmatrix (Name, Typ, Erreichbarkeit einer Variablen in der Bibliothek und Gesperrt/Entsperrt-Status) für alle Variablen und Bibliotheksobjekte zurück, die im aktuellen Problem definiert sind.

Wenn keine Variablen definiert sind, gibt **getVarInfo()** die Zeichenfolge "KEINE" (NONE) zurück.

getVarInfo()	"NONE"
Define x=5	Done
Lock x	Done
Define LibPriv y={1,2,3}	Done
Define LibPub z(x)=3·x ² -x	Done
getVarInfo()	$\begin{bmatrix} x & \text{"NUM"} & \text{"["} & 1 \\ y & \text{"LIST"} & \text{"LibPriv"} & 0 \\ z & \text{"FUNC"} & \text{"LibPub"} & 0 \end{bmatrix}$
getVarInfo(tmp3)	"Error: Argument must be a string"
getVarInfo("tmp3")	$\begin{bmatrix} volcyL2 & \text{"NONE"} & \text{"LibPub"} & 0 \end{bmatrix}$

`getVarInfo(BiblioNameString)`gibt eine Matrix zurück, die Informationen zu allen Bibliotheksobjekten enthält, die in der Bibliothek *BiblioNameString* definiert sind. *BiblioNameString* muss eine Zeichenfolge (in Anführungszeichen eingeschlossener Text) oder eine Zeichenfolgenvariable sein.

Wenn die Bibliothek *BiblioNameString* nicht existiert, wird ein Fehler angezeigt.

Beachten Sie das Beispiel links, in dem das Ergebnis von `getVarInfo()` der Variablen *vs* zugewiesen wird. Beim Versuch, Zeile 2 oder Zeile 3 von *vs* anzuzeigen, wird der Fehler *„Liste oder Matrix ungültig“* zurückgegeben, weil mindestens eines der Elemente in diesen Zeilen (Variable *b* zum Beispiel) eine Matrix ergibt.

Dieser Fehler kann auch auftreten, wenn *Ans* zum Neuberechnen eines `getVarInfo()`-Ergebnisses verwendet wird.

Das System liefert den obigen Fehler, weil die aktuelle Version der Software keine verallgemeinerte Matrixstruktur unterstützt, bei der ein Element einer Matrix eine Matrix oder Liste sein kann.

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

Goto (Gehe zu)

Goto MarkeName

Setzt die Programmausführung bei der Marke *MarkeName* fort.

MarkeName muss im selben Programm mit der Anweisung **Lbl** definiert worden sein.

Hinweis zum Eingeben des Beispiele:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define `g()=Func`

Done

Local *temp,i*

$0 \rightarrow temp$

$1 \rightarrow i$

Lbl *top*

$temp+i \rightarrow temp$

If $i < 10$ Then

$i+1 \rightarrow i$

Goto *top*

EndIf

Return *temp*

EndFunc

`g()`

55

►Grad (Neugrad)

Katalog >

Ausdr1 ►Grad ⇒ *Ausdruck*

Wandelt *Ausdr1* ins Winkelmaß Neugrad um.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Grad eintippen.

Im Grad-Modus:

(1.5) ►Grad $(1.66667)^g$

Im Bogenmaß-Modus:

(1.5) ►Grad $(95.493)^g$

/

identity()

Katalog >

identity(Ganze Zahl) ⇒ Matrix

Gibt die Einheitsmatrix mit der Dimension *Ganzzahl* zurück.

Ganzzahl muss eine positive ganze Zahl sein.

identity(4)

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

If

Katalog >

**If BooleanExpr
Anweisungen**

Define $g(x)=\text{Func}$ Done
 If $x < 0$ Then
 Return x^2
 EndIf
 EndFunc

**If BooleanExpr Then
Block
EndIf**

$g(-2)$ 4

Wenn Boolescher Ausdruck wahr ergibt, wird die Einzelanweisung Anweisung oder der Anweisungsblock Block ausgeführt und danach mit Endif fortgefahren.

Wenn Boolescher Ausdruck falsch ergibt, wird das Programm fortgesetzt, ohne dass die Einzelanweisung bzw. der Anweisungsblock ausgeführt werden.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen sein, die durch ":" getrennt sind Zeichen.

Hinweis zum Eingeben des Beispiels:
 Anweisungen für die Eingabe von
 mehrzeiligen Programm- und
 Funktionsdefinitionen finden Sie im
 Abschnitt „Calculator“ des
 Produkthandbuchs.

```
If BooleanExpr Then
  Block1
Else
  Block2
EndIf
```

Wenn Boolescher Ausdruck wahr ergibt,
 wird Block1 ausgeführt und dann Block2
 übersprungen.

Wenn Boolescher Ausdruck falsch ergibt,
 wird Block1 übersprungen, aber Block2
 ausgeführt.

Block1 und Block2 können einzelne
 Anweisungen sein.

```
If BooleanExpr1 Then
  Block1
ElseIf BooleanExpr2 Then
  Block2
:
ElseIf BooleanExprN Then
  BlockN
EndIf
```

Gestattet Programmverzweigungen. Wenn
 Boolescher Ausdruck1 wahr ergibt, wird
 Block1 ausgeführt. Wenn Boolescher
 Ausdruck1 falsch ergibt, wird Boolescher
 Ausdruck2 bewertet usw.

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

g(12)	12
g(-12)	12

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

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

ifFn()

ifFn(BoolescherAusdruck,Wert_wenn_wahr [,Wert_wenn_falsch [,Wert_wenn_unbekannt]]) ⇒ Ausdruck, Liste oder Matrix

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

Testwert von 1 ist kleiner als 2.5, somit wird
 das entsprechende

Wert_wenn_wahr-Element von 5 in die
 Ergebnisliste kopiert.

Wertet den Booleschen Ausdruck

BoolescherAusdruck (oder jedes einzelne Element von *BoolescherAusdruck*) aus und erstellt ein Ergebnis auf der Grundlage folgender Regeln:

- *BoolescherAusdruck* kann einen Einzelwert, eine Liste oder eine Matrix testen.
- Wenn ein Element von *BoolescherAusdruck* als wahr bewertet wird, wird das entsprechende Element aus *Wert_wenn_wahr* zurückgegeben.
- Wenn ein Element von *BoolescherAusdruck* als falsch bewertet wird, wird das entsprechende Element aus *Wert_wenn_falsch* zurückgegeben. Wenn Sie *Wert_wenn_falsch* weglassen, wird Undef zurückgegeben.
- Wenn ein Element von *BoolescherAusdruck* weder wahr noch falsch ist, wird das entsprechende Element aus *Wert_wenn_unbekannt* zurückgegeben. Wenn Sie *Wert_wenn_unbekannt* weglassen, wird Undef zurückgegeben.
- Wenn das zweite, dritte oder vierte Argument der Funktion **ifFn()** ein einzelnen Ausdruck ist, wird der Boolesche Test für jede Position in *BoolescherAusdruck* durchgeführt.

Hinweis: Wenn die vereinfachte Anweisung *BoolescherAusdruck* eine Liste oder Matrix einbezieht, müssen alle anderen Listen- oder Matrixanweisungen dieselbe(n) Dimension(en) haben, und auch das Ergebnis wird dieselben(n) Dimension(en) haben.

Testwert von **2** ist kleiner als 2.5, somit wird das entsprechende

Wert_wenn_wahr-Element von **6** in die Ergebnisliste kopiert.

Testwert von **3** ist nicht kleiner als 2.5, somit wird das entsprechende *Wert_wenn_falsch*-Element von **10** in die Ergebnisliste kopiert.

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

Wert_wenn_wahr ist ein einzelner Wert und "entspricht" einer beliebigen ausgewählten Position.

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

Wert_wenn_falsch ist nicht spezifiziert. Undef wird verwendet.

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

Ein aus *Wert_wenn_wahr* ausgewähltes Element. Ein aus *Wert_wenn_unbekannt* ausgewähltes Element.

imag(*ValueI***)** ⇒ *Wert*

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

2

Gibt den Imaginärteil des Arguments zurück.

imag()**Katalog >** **imag(List I) ⇒ Liste**

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

Gibt eine Liste der Imaginärteile der Elemente zurück.

imag(Matrix I) ⇒ Matriximag($\begin{bmatrix} 1 & 2 \\ i \cdot 3 & i \cdot 4 \end{bmatrix}$) $\begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$

Gibt eine Matrix der Imaginärteile der Elemente zurück.

Umleitung**Siehe #(), Seite 210.****inString()****Katalog >** **inString(Quellstring, Teilstring[, Start])**
⇒ GanzzahlinString("Hello there","the") 7
inString("ABCEFG","D") 0Gibt die Position des Zeichens von *Quellstring* zurück, an der das erste Vorkommen von *Teilstring* beginnt.Start legt fest (sofern angegeben), an welcher Zeichenposition innerhalb von *Quellstring* die Suche beginnt. Vorgabe = 1 (das erste Zeichen von *Quellstring*).Enthält *Quellstring* die Zeichenkette *Teilstring* nicht oder ist *Start* > Länge von *Quellstring*, wird Null zurückgegeben.**int()****Katalog >** **int(Value) ⇒ Ganzzahl**
int(List I) ⇒ Liste
int(Matrix I) ⇒ Matrixint(-2.5) -3.
int([-1.234 0 0.37]) [-2. 0 0.]Gibt die größte ganze Zahl zurück, die kleiner oder gleich dem Argument ist. Diese Funktion ist identisch mit **floor()**.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

int()

Katalog >

Für eine Liste oder Matrix wird für jedes Element die größte ganze Zahl zurückgegeben, die kleiner oder gleich dem Element ist.

intDiv()

Katalog >

intDiv(Zahl1, Zahl2) \Rightarrow Ganzzahl
intDiv(Liste1, Liste2) \Rightarrow Liste
intDiv(Matrix1, Matrix2) \Rightarrow Matrix

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

Gibt den mit Vorzeichen versehenen ganzzahligen Teil von $(Zahl1 \div Zahl2)$ zurück.

Für eine Liste oder Matrix wird für jedes Elementpaar der mit Vorzeichen versehene ganzzahlige Teil von $(\text{Argument1} \div \text{Argument2})$ zurückgegeben.

Interpolieren ()

Katalog >

Interpolieren(xWert, xListe, yListe, yStrListe) \Rightarrow Liste

Diese Funktion tut folgendes:

Bei gegebenen $xListe$, $yListe=f(xListe)$ und $yStrListe=f'(xListe)$ für eine unbekannte Funktion f wird eine kubische Interpolierende zur Approximierung der Funktion f bei $xWert$ verwendet. Es wird angenommen, dass $xListe$ eine Liste monoton steigender oder fallender Zahlen ist; jedoch kann diese Funktion auch einen Wert zurückgeben, wenn dies nicht der Fall ist. Diese Funktion geht $xListe$ durch und sucht nach einem Intervall $[xListe[i], xListe[i+1]]$, das $xWert$ enthält. Wenn sie ein solches Intervall findet, gibt sie einen interpolierten Wert für $f(xWert)$ zurück; anderenfalls gibt sie **zurück.undefined**.

$xListe$, $yListe$ und $yStrListe$ müssen die gleiche Dimension ≥ 2 besitzen und Ausdrücke enthalten, die zu Zahlen vereinfachbar sind.

Differentialgleichung:
 $y' = -3y + 6t + 5$ und $y(0) = 5$

$rk := rk23[-3y + 6t + 5, t, y, \{0, 10\}, 5, 1]$
$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 5. & 3.19499 & 5.00394 & 6.99957 & 9.00593 \end{bmatrix}$
10

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleright und \blacktriangleright , um den Cursor zu bewegen.

Verwenden Sie die Funktion **interpolate()**, um die Funktionswerte für die Liste $xWert$ zu berechnen:

$xvalueList := seq(i, i, 0, 10, 0.5)$
$\{0, 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6., 6.5, 7., 7.5, 8., 8.5, 9., 9.5, 10.\}$
$xList := mat\triangleright list(rk[1])$
$\{0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.\}$
$yList := mat\triangleright list(rk[2])$
$\{5., 3.19499, 5.00394, 6.99957, 9.00593, 10.9978, 12.9925, 15.98819, 19.00129, 22.98221, 26.0066, 30.99957, 36.99957, 43.99957, 51.99957, 60.99957, 70.99957, 81.99957, 93.99957, 106.99957, 121.99957, 137.99957, 154.99957, 172.99957, 191.99957, 211.99957, 232.99957, 254.99957, 277.99957, 301.99957, 326.99957, 352.99957, 379.99957, 407.99957, 436.99957, 466.99957, 497.99957, 529.99957, 562.99957, 596.99957, 631.99957, 667.99957, 704.99957, 742.99957, 781.99957, 821.99957, 862.99957, 904.99957, 947.99957, 991.99957, 1036.99957, 1082.99957, 1129.99957, 1177.99957, 1226.99957, 1276.99957, 1327.99957, 1379.99957, 1432.99957, 1486.99957, 1541.99957, 1597.99957, 1654.99957, 1712.99957, 1771.99957, 1831.99957, 1892.99957, 1954.99957, 2017.99957, 2081.99957, 2146.99957, 2212.99957, 2279.99957, 2347.99957, 2416.99957, 2486.99957, 2557.99957, 2629.99957, 2702.99957, 2776.99957, 2851.99957, 2927.99957, 3004.99957, 3082.99957, 3161.99957, 3241.99957, 3322.99957, 3404.99957, 3487.99957, 3571.99957, 3656.99957, 3742.99957, 3829.99957, 3917.99957, 4006.99957, 4096.99957, 4187.99957, 4279.99957, 4372.99957, 4466.99957, 4561.99957, 4657.99957, 4754.99957, 4852.99957, 4951.99957, 5051.99957, 5152.99957, 5254.99957, 5357.99957, 5461.99957, 5566.99957, 5672.99957, 5779.99957, 5887.99957, 5996.99957, 6106.99957, 6217.99957, 6329.99957, 6442.99957, 6556.99957, 6671.99957, 6787.99957, 6894.99957, 6999.99957, 7106.99957, 7214.99957, 7323.99957, 7433.99957, 7544.99957, 7656.99957, 7769.99957, 7883.99957, 7998.99957, 8114.99957, 8231.99957, 8349.99957, 8468.99957, 8588.99957, 8709.99957, 8831.99957, 8954.99957, 9078.99957, 9203.99957, 9329.99957, 9456.99957, 9584.99957, 9713.99957, 9843.99957, 9974.99957, 10106.99957, 10239.99957, 10373.99957, 10508.99957, 10644.99957, 10781.99957, 10919.99957, 11058.99957, 11208.99957, 11359.99957, 11511.99957, 11664.99957, 11818.99957, 11973.99957, 12129.99957, 12286.99957, 12444.99957, 12603.99957, 12763.99957, 12924.99957, 13086.99957, 13249.99957, 13413.99957, 13578.99957, 13744.99957, 13911.99957, 14079.99957, 14248.99957, 14418.99957, 14589.99957, 14761.99957, 14934.99957, 15108.99957, 15283.99957, 15459.99957, 15636.99957, 15814.99957, 16093.99957, 16373.99957, 16654.99957, 16936.99957, 17219.99957, 17503.99957, 17788.99957, 18074.99957, 18361.99957, 18649.99957, 18938.99957, 19228.99957, 19519.99957, 19811.99957, 20104.99957, 20408.99957, 20713.99957, 21020.99957, 21328.99957, 21638.99957, 21950.99957, 22263.99957, 22577.99957, 22893.99957, 23210.99957, 23528.99957, 23848.99957, 24169.99957, 24492.99957, 24816.99957, 25141.99957, 25467.99957, 25795.99957, 26124.99957, 26454.99957, 26785.99957, 27117.99957, 27451.99957, 27786.99957, 28122.99957, 28459.99957, 28797.99957, 29136.99957, 29476.99957, 29817.99957, 30159.99957, 30503.99957, 30848.99957, 31194.99957, 31541.99957, 31889.99957, 32238.99957, 32588.99957, 32939.99957, 33291.99957, 33644.99957, 34000.99957, 34357.99957, 34716.99957, 35076.99957, 35437.99957, 35799.99957, 36163.99957, 36527.99957, 36893.99957, 37260.99957, 37628.99957, 38007.99957, 38387.99957, 38768.99957, 39151.99957, 39535.99957, 39920.99957, 40306.99957, 40693.99957, 41082.99957, 41472.99957, 41863.99957, 42255.99957, 42648.99957, 43043.99957, 43439.99957, 43836.99957, 44234.99957, 44633.99957, 45033.99957, 45434.99957, 45836.99957, 46239.99957, 46644.99957, 47050.99957, 47457.99957, 47866.99957, 48276.99957, 48687.99957, 49099.99957, 49512.99957, 49926.99957, 50341.99957, 50757.99957, 51175.99957, 51594.99957, 51914.99957, 52235.99957, 52557.99957, 52881.99957, 53206.99957, 53532.99957, 53859.99957, 54188.99957, 54518.99957, 54849.99957, 55182.99957, 55516.99957, 55852.99957, 56189.99957, 56528.99957, 56869.99957, 57212.99957, 57556.99957, 57892.99957, 58229.99957, 58568.99957, 58909.99957, 59252.99957, 59596.99957, 59942.99957, 60289.99957, 60638.99957, 61089.99957, 61442.99957, 61806.99957, 62172.99957, 62540.99957, 62909.99957, 63280.99957, 63652.99957, 64026.99957, 64392.99957, 64760.99957, 65129.99957, 65499.99957, 65861.99957, 66224.99957, 66588.99957, 66953.99957, 67319.99957, 67687.99957, 68056.99957, 68426.99957, 68797.99957, 69169.99957, 69542.99957, 69916.99957, 70291.99957, 70667.99957, 71045.99957, 71424.99957, 71804.99957, 72185.99957, 72567.99957, 72951.99957, 73335.99957, 73720.99957, 74106.99957, 74493.99957, 74882.99957, 75272.99957, 75663.99957, 76055.99957, 76448.99957, 76843.99957, 77238.99957, 77635.99957, 78033.99957, 78432.99957, 78832.99957, 79233.99957, 79635.99957, 80038.99957, 80443.99957, 80849.99957, 81256.99957, 81664.99957, 82073.99957, 82483.99957, 82894.99957, 83306.99957, 83719.99957, 84133.99957, 84548.99957, 84964.99957, 85381.99957, 85799.99957, 86219.99957, 86641.99957, 87064.99957, 87488.99957, 87913.99957, 88339.99957, 88766.99957, 89204.99957, 89643.99957, 90083.99957, 90524.99957, 90966.99957, 91409.99957, 91853.99957, 92307.99957, 92762.99957, 93218.99957, 93675.99957, 94133.99957, 94592.99957, 94952.99957, 95313.99957, 95675.99957, 96038.99957, 96393.99957, 96749.99957, 97106.99957, 97464.99957, 97823.99957, 98183.99957, 98543.99957, 98904.99957, 99266.99957, 99629.99957, 99993.99957, 100357.99957, 100722.99957, 101088.99957, 101456.99957, 101825.99957, 102195.99957, 102566.99957, 102938.99957, 103312.99957, 103686.99957, 104062.99957, 104439.99957, 104817.99957, 105196.99957, 105576.99957, 105957.99957, 106339.99957, 106722.99957, 107106.99957, 107491.99957, 107877.99957, 108265.99957, 108654.99957, 109044.99957, 109435.99957, 109826.99957, 110218.99957, 110612.99957, 111006.99957, 111401.99957, 111797.99957, 112194.99957, 112592.99957, 112991.99957, 113392.99957, 113794.99957, 114197.99957, 114502.99957, 114808.99957, 115116.99957, 115425.99957, 115735.99957, 116046.99957, 116358.99957, 116671.99957, 117085.99957, 117500.99957, 117916.99957, 118333.99957, 118751.99957, 119170.99957, 119590.99957, 119912.99957, 120235.99957, 120559.99957, 120884.99957, 121210.99957, 121537.99957, 121866.99957, 122206.99957, 122547.99957, 122889.99957, 123233.99957, 123578.99957, 123925.99957, 124273.99957, 124623.99957, 125074.99957, 125426.99957, 125779.99957, 126134.99957, 126490.99957, 126847.99957, 127206.99957, 127566.99957, 127927.99957, 128289.99957, 128653.99957, 129020.99957, 129389.99957, 129761.99957, 130135.99957, 130511.99957, 130889.99957, 131269.99957, 131652.99957, 132037.99957, 132424.99957, 132813.99957, 133204.99957, 133597.99957, 133993.99957, 134391.99957, 134792.99957, 135196.99957, 135502.99957, 135811.99957, 136122.99957, 136435.99957, 136749.99957, 137065.99957, 137383.99957, 137702.99957, 138023.99957, 138346.99957, 138669.99957, 139004.99957, 139340.99957, 139677.99957, 140016.99957, 140357.99957, 140700.99957, 141044.99957, 141390.99957, 141737.99957, 142086.99957, 142437.99957, 142789.99957, 143143.99957, 143500.99957, 143859.99957, 144221.99957, 144585.99957, 144951.99957, 145320.99957, 145691.99957, 146064.99957, 146439.99957, 146816.99957, 147195.99957, 147576.99957, 147959.99957, 148344.99957, 148731.99957, 149120.99957, 149511.99957, 149904.99957, 150300.99957, 150698.99957, 151099.99957, 151493.99957, 151889.99957, 152288.99957, 152689.99957, 153093.99957, 153500.99957, 153909.99957, 154321.99957, 154736.99957, 155153.99957, 155573.99957, 155996.99957, 156422.99957, 156851.99957, 157283.99957, 157717.99957, 158153.99957, 158592.99957, 159034.99957, 159479.99957, 159926.99957, 160375.99957, 160826.99957, 161279.99957, 161735.99957, 162194.99957, 162656.99957, 163121.99957, 163588.99957, 164058.99957, 164531.99957, 165006.99957, 165484.99957, 165965.99957, 166448.99957, 166934.99957, 167423.99957, 167915.99957, 168409.99957, 168906.99957, 169406.99957, 169909.99957, 170415.99957, 170923.99957, 171434.99957, 171947.99957, 172462.99957, 172980.99957, 173501.99957, 174025.99957, 174552.99957, 175082.99957, 175615.99957, 176151.99957, 176689.99957, 177229.99957, 177772.99957, 178317.99957, 178865.99957, 179416.99957, 180069.99957, 180725.99957, 181384.99957, 182046.99957, 182711.99957, 183378.99957, 184048.99957, 184721.99957, 185397.99957, 186076.99957, 186758.99957, 187443.99957, 188131.99957, 188821.99957, 189514.99957, 190211.99957, 190911.99957, 191614.99957, 192320.99957, 193030.99957, 193743.99957, 194460.99957, 195179.99957, 195892.99957, 196608.99957, 197327.99957, 198049.99957, 198774.99957, 199502.99957, 200233.99957, 200967.99957, 201704.99957, 202444.99957, 203187.99957, 203933.99957, 204682.99957, 205434.99957, 206189.99957, 206947.99957, 207708.99957, 208472.99957, 209240.99957, 210009.99957, 210782.99957, 211558.99957, 212337.99957, 213119.99957, 213894.99957, 214672.99957, 215453.99957, 216237.99957, 217024.99957, 217814.99957, 218607.99957, 219403.99957, 220192.99957, 220984.99957, 221779.99957, 222577.99957, 223378.99957, 224182.99957, 224989.99957, 225799.99957, 226612.99957, 227427.99957, 228245.99957, 229065.99957, 230888.99957, 231714.99957, 232543.99957, 233375.99957, 234209.99957, 235046.99957, 235886.99957, 236729.99957, 237575.99957, 238424.99957, 239276.99957, 240131.99957, 241088.99957, 242048.99957, 243011.99957, 244077.99957, 245146.99957, 246218.99957, 247293.99957, 248371.99957, 249452.99957, 250536.99957, 251623.99957, 252713.99957, 253806.99957, 254902.99957, 256001.99957, 257103.99957, 258208.99957, 259316.99957, 260427.99957, 261541.99957, 262658.99957, 263778.99957, 264891.99957, 265998.99957, 267108.99957, 268211.99957, 269316.99957, 270424.99957, 271534.99957, 272646.99957, 273761.99957, 274878.99957, 275998.99957, 277111.99957, 278226.99957, 279344.99957, 280464.99957, 281586.99957, 282709.99957, 283834.99957, 284961.99957, 286090.99957, 287221.99957, 288355.99957, 289491.99957, 290629.99957, 291769.99957, 292912.99957, 294057.99957, 295205.99957, 296356.99957, 297509.99957, 298665.99957, 299823.99957, 301084.99957, 302247.99957, 303413.99957, 304582.99957, 305754.99957, 306928.99957, 308105.99957, 309284.99957, 310466.99957, 311651.99957, 312838.99957, 314028.99957, 315221.99957, 316416.99957, 317614.99957, 318814.99957, 319916.99957, 321021.999$

Interpolieren()

Katalog >

xWert kann eine Zahl oder eine Zahlenliste sein.

invχ²()

Katalog >

invχ²(Fläche,FreiGrad)

invChi2(Fläche,FreiGrad)

Berechnet die inverse kumulative χ^2 (Chi-Quadrat) Wahrscheinlichkeitsfunktion, die durch Freiheitsgrade *FreiGrad* für eine bestimmte *Fläche* unter der Kurve festgelegt ist.

invF()

Katalog >

invF(Fläche,FreiGradZähler,FreiGradNenner)

invF(Fläche,FreiGradZähler,FreiGradNenner)

Berechnet die inverse kumulative F Verteilungsfunktion, die durch *FreiGradZähler* und *FreiGradNenner* für eine bestimmte *Fläche* unter der Kurve festgelegt ist.

invBinom()

Katalog >

invBinom(CumulativeProb,NumTrials,Prob,OutputForm)⇒ Skalar oder Matrix

Die Funktion gibt anhand der angegebenen Zahl von Versuchen (*NumTrials*) und der Erfolgswahrscheinlichkeit jedes Versuches (*Prob*), die Mindestanzahl erfolgreicher Versuche *k* aus, so dass die kumulative Wahrscheinlichkeit für *k* größer oder gleich der gegebenen kumulativen Wahrscheinlichkeit (*CumulativeProb*) ist.

OutputForm=0, gibt Ergebnis als Skalar (Standard) an.

Beispiel: Mary und Kevin spielen ein Würfelspiel. Mary soll raten, wie häufig bei 30 Mal würfeln die Zahl 6 angezeigt wird. Sollte die Zahl 6 genauso häufig oder weniger angezeigt werden, gewinnt Mary. Je niedriger die Zahl, die sie schätzt, desto höher ist ihr Gewinn. Was ist die niedrigste Zahl, die Mary angeben kann, wenn sie eine Gewinnwahrscheinlichkeit von mehr als 77 % erzielen möchte?

invBinom()

Katalog >

OutputForm=1, gibt Ergebnis als Matrix an.

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

invBinomN()

Katalog >

invBinomN(*CumulativeProb, Prob, NumSuccess, OutputForm*) \Rightarrow Skalar oder Matrix

Die Funktion gibt anhand der Erfolgswahrscheinlichkeit bei jedem Versuch (*Prob*) und der Anzahl der tatsächlichen Erfolge (*NumSuccess*) die Mindestanzahl an Versuchen *N*, aus, so dass die kumulative Wahrscheinlichkeit für *x* kleiner oder gleich der gegebenen kumulativen Wahrscheinlichkeit (*CumulativeProb*) ist.

OutputForm=0, gibt Ergebnis als Skalar (Standard) an.

OutputForm=1, gibt Ergebnis als Matrix an.

Beispiel: Monique übt Zielwürfe auf das Netz. Aus Erfahrung weiß sie, dass sie mit einer Wahrscheinlichkeit von 70 % trifft. Sie hat vor, so lange zu üben, bis sie 50 Mal getroffen hat. Wie häufig muss sie werfen, um sicherzustellen, dass die Wahrscheinlichkeit, 50 Mal zu treffen größer als 0,99 ist?

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()

Katalog >

invNorm(*Fläche[,μ[σ]]*)

Berechnet die inverse Summennormalverteilungsfunktion für einen gegebenen *Bereich* unter der Normalverteilungskurve, die über μ und σ definiert ist.

invt()

Katalog >

invt(*Fläche,FreiGrad*)

Berechnet die inverse kumulative Wahrscheinlichkeitsfunktion student-t, die über den Freiheitsgrad, *df*, definiert ist, für eine bestimmte *Fläche* unter der Kurve.

iPart()**iPart(Zahl) \Rightarrow Ganzzahl****iPart(Liste l) \Rightarrow Liste****iPart(Matrix l) \Rightarrow Matrix**

Gibt den ganzzahligen Teil des Arguments zurück.

Für eine Liste oder Matrix wird der ganzzahlige Teil jedes Elements zurückgegeben.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

iPart(-1.234)

-1.

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

{1, -2, 7.}

irr()**irr(CF0,CFListe [,CFFreq]) \Rightarrow Wert**

Finanzfunktion, die den internen Zinsfluss einer Investition berechnet.

CF0 ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

CFListe ist eine Liste von Cash-Flow-Beträgen nach dem Anfangs-Cash-Flow *CF0*.

CFFreq ist eine optionale Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von *CFList* ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzahlen < 10.000 sein.

Hinweis: Siehe auch **mirr()**, Seite 104.

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()**isPrime(Zahl) \Rightarrow Boolescher konstanter Ausdruck**

Gibt "wahr" oder "falsch" zurück, um anzugeben, ob es sich bei *Zahl* um eine ganze Zahl ≥ 2 handelt, die nur durch sich selbst oder 1 ganzzahlig teilbar ist.

isPrime(5)

true

isPrime(6)

false

Funktion zum Auffinden der nächsten Primzahl nach einer angegebenen Zahl:

isPrime()

Katalog >

Übersteigt *Zahl* ca. 306 Stellen und hat sie keine Faktoren ≤ 1021 , dann zeigt **isPrime** (*Zahl*) eine Fehlermeldung an.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define *nextprim(n)*=Func

Done

Loop

$n+1 \rightarrow n$

If *isPrime(n)*

Return *n*

EndLoop

EndFunc

nextprim(7)

11

isVoid()

Katalog >

isVoid(Var) \Rightarrow Boolescher konstanter Ausdruck

isVoid(Ausdruck) \Rightarrow Boolescher konstanter Ausdruck

isVoid(Liste) \Rightarrow Liste Boolescher konstanter Ausdrücke

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

Gibt wahr oder falsch zurück, um anzuzeigen, ob das Argument ein ungültiger Datentyp ist.

Weitere Informationen zu ungültigen Elementen finden Sie auf Seite 235.

L

Lbl (Marke)

Katalog >

Lbl MarkeName

Definiert in einer Funktion eine Marke mit dem Namen *MarkeName*.

Mit der Anweisung **Goto MarkeName** können Sie die Ausführung an der Anweisung fortsetzen, die unmittelbar auf die Marke folgt.

Für *MarkeName* gelten die gleichen Benennungsregeln wie für einen Variablenamen.

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

g()

55

Hinweis zum Eingeben des Beispiels:
 Anweisungen für die Eingabe von
 mehrzeiligen Programm- und
 Funktionsdefinitionen finden Sie im
 Abschnitt „Calculator“ des
 Produkthandbuchs.

lcm() (Kleinste gemeinsame Vielfache)

lcm(Zahl1, Zahl2)⇒Ausdruck

$\text{lcm}(6,9)$ 18

lcm(Liste1, Liste2)⇒Liste

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

lcm(Matrix1, Matrix2)⇒Matrix

Gibt das kleinste gemeinsame Vielfache der beiden Argumente zurück. Das **lcm** zweier Brüche ist das **lcm** ihrer Zähler dividiert durch den größten gemeinsamen Teiler (**gcd**) ihrer Nenner. Das **lcm** von Dezimalbruchzahlen ist ihr Produkt.

Für zwei Listen oder Matrizen wird das kleinste gemeinsame Vielfache der entsprechenden Elemente zurückgegeben.

left() (Links)

left(Quellstring[, Anz])⇒String

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

Gibt *Anz* Zeichen zurück, die links in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

left(Liste1[, Anz])⇒Liste

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

Gibt *Anz* Elemente zurück, die links in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

left(Vergleich)⇒Ausdruck

Gibt die linke Seite einer Gleichung oder Ungleichung zurück.

libShortcut()

**libShortcut(BiblioNameString,
VerknNameString)**

[, BiblioPrivMerker]) \Rightarrow Liste von
Variablen

Erstellt eine Variablengruppe im aktuellen Problem, die Verweise auf alle Objekte im angegebenen Bibliotheksdokument *BiblioNameString* enthält. Fügt außerdem die Gruppenmitglieder dem Variablenmenü hinzu. Sie können dann auf jedes Objekt mit *VerknNameString* verweisen.

Setzen Sie *BiblioPrivMerker=0*, um private Bibliotheksobjekte auszuschließen (Standard)

Setzen Sie *BiblioPrivMerker=1*, um private Bibliotheksobjekte einzubeziehen

Informationen zum Kopieren einer Variablengruppe finden Sie unter **CopyVar** (Seite 27).

Informationen zum Löschen einer Variablengruppe finden Sie unter **DelVar** (Seite 42).

Dieses Beispiel setzt ein richtig gespeichertes und aktualisiertes Bibliotheksdokument namens **linalg2** voraus, das als *clearmat*, *gauss1* und *gauss2* definierte Objekte enthält.

```
getVarInfo("linalg2")
```

<i>clearmat</i>	"FUNC"	"LibPub "
<i>gauss1</i>	"PRGM"	"LibPriv "
<i>gauss2</i>	"FUNC"	"LibPub "

```
libShortcut("linalg2","la")
```

```
{la.clearmat,la.gauss2}
```

```
libShortcut("linalg2","la",1)
```

```
{la.clearmat,la.gauss1,la.gauss2}
```

LinRegBx

LinRegBx X,Y,[Häuf],[Kategorie,Mit]]

Berechnet die lineare Regression $y = a + b \cdot x$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt *X* und *Y* an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $a+b \cdot x$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LinRegMx *X, Y[, [Häuf][, Kategorie, Mit]]*

Berechnet die lineare Regression $y = m \cdot x + b$ auf Liste X und Y mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt X und Y an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 235).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $m \cdot x + b$
stat.m, stat.b	Regressionskoeffizienten
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde

AusgabevARIABLE	Beschreibung
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LinRegtIntervals (Lineare Regressions-t-Intervalle)

Katalog >

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

Für Steigung. Berechnet ein Konfidenzintervall des Niveaus K für die Steigung.

LinRegtIntervals *X,Y[,F[,1,XWert[,KStufe]]]*

Für Antwort. Berechnet einen vorhergesagten y-Wert, ein Niveau-K-Vorhersageintervall für eine einzelne Beobachtung und ein Niveau-K-Konfidenzintervall für die mittlere Antwort.

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

F ist eine optionale Liste von Frequenzwerten. Jedes Element in *F* gibt die Häufigkeit für jeden entsprechenden *X* und *Y* Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $a+b \cdot x$
stat.a, stat.b	Regressionskoeffizienten
stat.df	Freiheitsgrade
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient

Ausgabeverable	Beschreibung
stat.Resid	Residuen von der Regression

Nur für Steigung

Ausgabeverable	Beschreibung
[stat.CLower, stat.CUpper]	Konfidenzintervall für die Steigung
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SESlope	Standardfehler der Steigung
stat.s	Standardfehler an der Linie

Nur für Antwort

Ausgabeverable	Beschreibung
[stat.CLower, stat.CUpper]	Konfidenzintervall für die mittlere Antwort
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SE	Standardfehler der mittleren Antwort
[stat.LowerPred, stat.UpperPred]	Vorhersageintervall für eine einzelne Beobachtung
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat. \hat{y}	$a + b \cdot X$ Wert

LinRegtTest (t-Test bei linearer Regression)

Katalog > 

LinRegtTest $X, Y[, Häuf[, Hypoth]]$

Berechnet eine lineare Regression auf den X - und Y -Listen und einen t -Test auf dem Wert der Steigung β und den Korrelationskoeffizienten ρ für die Gleichung $y = \alpha + \beta x$. Er berechnet die Null-Hypothese $H_0: \beta = 0$ (gleichwertig, $\rho = 0$) in Bezug auf eine von drei alternativen Hypothesen.

Alle Listen müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Hypoth ist ein optionaler Wert, der eine von drei alternativen Hypothesen angibt, in Bezug auf die Nullhypothese ($H_0: \beta = p = 0$) untersucht wird.

Für $H_a: \beta \neq 0$ und $p \neq 0$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \beta < 0$ und $p < 0$ setzen Sie *Hypoth*<0

Für $H_a: \beta > 0$ und $p > 0$ setzen Sie *Hypoth*>0

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert.
(Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot x$
stat.t	t-Statistik für Signifikanztest
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade
stat.a, stat.b	Regressionskoeffizienten
stat.s	Standardfehler an der Linie
stat.SESlope	Standardfehler der Steigung
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression

linSolve()

Katalog >

linSolve(*SystemLinearerGl*, *Var1*, *Var2*, ...) \Rightarrow Liste

linSolve(*LineareGl1 and LineareGl2 and ...*, *Var1*, *Var2*, ...) \Rightarrow Liste

linSolve(*{LineareGl1, LineareGl2, ...}*, *Var1*, *Var2*, ...) \Rightarrow Liste

linSolve(*SystemLinearerGl*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

linSolve(*LineareGl1 and LineareGl2 and ...*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

linSolve(*{LineareGl1, LineareGl2, ...}*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

Liefert eine Liste mit Lösungen für die Variablen *Var1*, *Var2*, ...

Das erste Argument muss ein System linearer Gleichungen bzw. eine einzelne lineare Gleichung ergeben. Andernfalls tritt ein Argumentfehler auf.

Die Auswertung von **linSolve**(x=1 and x=2,x) führt beispielsweise zu dem Ergebnis "Argumentfehler".

$\text{linSolve}\left(\begin{cases} 2x+4y=3 \\ 5x-3y=7 \end{cases}, \{x,y\}\right)$	$\left\{\frac{37}{26}, \frac{1}{26}\right\}$
$\text{linSolve}\left(\begin{cases} 2x=3 \\ 5x-3y=7 \end{cases}, \{x,y\}\right)$	$\left\{\frac{3}{2}, \frac{1}{6}\right\}$
$\text{linSolve}\left(\begin{cases} apple+4\cdot pear=23 \\ 5\cdot apple-pear=17 \end{cases}, \{apple,pear\}\right)$	$\left\{\frac{13}{3}, \frac{14}{3}\right\}$
$\text{linSolve}\left(\begin{cases} apple+4\cdot \frac{pear}{3}=14 \\ -apple+pear=6 \end{cases}, \{apple,pear\}\right)$	$\left\{\frac{36}{13}, \frac{114}{13}\right\}$

Δlist() (Listendifferenz)

Katalog >

Δlist(*Liste1*) \Rightarrow Liste

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

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **deltaList**(...) eintippen.

Ergibt eine Liste mit den Differenzen der aufeinander folgenden Elemente in *Liste1*. Jedes Element in *Liste1* wird vom folgenden Element in *Liste1* subtrahiert. Die Ergebnisliste enthält stets ein Element weniger als die ursprüngliche *Liste1*.

list►mat() (Liste in Matrix)

list►mat(Liste [, ElementeProZeile])⇒Matrix

Gibt eine Matrix zurück, die Zeile für Zeile mit den Elementen aus *Liste* aufgefüllt wurde.

ElementeProZeile gibt (sofern angegeben) die Anzahl der Elemente pro Zeile an.
Vorgabe ist die Anzahl der Elemente in *Liste* (eine Zeile).

Wenn *Liste* die resultierende Matrix nicht vollständig auffüllt, werden Nullen hinzugefügt.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **list@>mat(...)** eintippen.

list►mat({1,2,3})	[1 2 3]
list►mat({1,2,3,4,5},2)	[1 2 3 4 5 0]

ln() (Natürlicher Logarithmus)
Tasten

ln(WertI)⇒Wert

ln(2.)	0.693147
--------	----------

ln(ListeI)⇒Liste

Gibt den natürlichen Logarithmus des Arguments zurück.

Gibt für eine Liste die natürlichen Logarithmen der einzelnen Elemente zurück.

Bei Komplex-Formatmodus reell:

ln({-3,1,2,5})	"Error: Non-real calculation"
----------------	-------------------------------

ln(QuadratmatrixI)⇒Quadratmatrix

Ergibt den natürlichen Matrix-Logarithmus von *QuadratmatrixI*. Dies ist nicht gleichbedeutend mit der Berechnung des natürlichen Logarithmus jedes einzelnen Elements. Näheres zum Berechnungsverfahren finden Sie im Abschnitt **cos()**.

Bei Komplex-Formatmodus kartesisch:

ln({-3,1,2,5})	{1.09861+3.14159·i,0.182322,1.60944}
----------------	--------------------------------------

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

Quadratmatrix X muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

LnReg**Katalog > **

LnReg $X, Y[, Häuf] [, Kategorie, Mit]$

Berechnet die logarithmische Regression $y = a + b \cdot \ln(x)$ auf Listen X und Y mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden X - und Y -Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot \ln(x)$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten ($\ln(x)$, y)

Ausgabevariable	Beschreibung
stat.Resid	Mit dem logarithmischen Modell verknüpfte Residuen
stat.ResidTrans	Residuen für die lineare Anpassung transformierter Daten
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

Local (Lokale Variable)

Katalog > 

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

Deklariert die angegebenen Variablen *Variable* als lokale Variablen. Diese Variablen existieren nur während der Auswertung einer Funktion und werden gelöscht, wenn die Funktion beendet wird.

Hinweis: Lokale Variablen sparen Speicherplatz, da sie nur temporär existieren. Außerdem stören sie keine vorhandenen globalen Variablenwerte. Lokale Variablen müssen für **For**-Schleifen und für das temporäre Speichern von Werten in mehrzeiligen Funktionen verwendet werden, da Änderungen globaler Variablen in einer Funktion unzulässig sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

```
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

LockVar1 [, Var2] [, Var3] ...**LockVar.**

Sperrt die angegebenen Variablen bzw. die Variablengruppe. Gesperrte Variablen können nicht geändert oder gelöscht werden.

Die Systemvariable *Ans* können Sie nicht sperren oder entsperren, ebenso können Sie die Systemvariablengruppen *stat.* oder *tvm.* nicht sperren.

Hinweis: Der Befehl **Sperren (Lock)** löscht den Rückgängig/Wiederholen-Verlauf, wenn er für nicht gesperrte Variablen verwendet wird.

Siehe **unLock**, Seite 182, und **getLockInfo()**, Seite 71.

a:=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() (Logarithmus)**ctrl** **10x** **Tasten****log(Wert1[,Wert2])**⇒*Wert*

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

log(Liste1[,Wert2])⇒*Liste*

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

Gibt für den Logarithmus des Arguments zur Basis *Ausdr2* zurück.

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

Hinweis: Siehe auch **Vorlage Logarithmus**, Seite 2.

Gibt bei einer Liste den Logarithmus der Elemente zur Basis *Wert2* zurück.

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

Wenn *Wert* weggelassen wird, wird 10 als Basis verwendet.

"Error: Non-real calculation"

log(Quadratmatrix1 [,Zahl2])⇒*Quadratmatrix*

Bei Komplex-Formatmodus reell:

Bei Komplex-Formatmodus kartesisch:

$$\log_{10}(\{-3,1,2,5\}) \\ \{0.477121+1.36438 \cdot i, 0.079181, 0.69897\}$$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

log() (Logarithmus)

ctrl 10^x Tasten

Gibt den Matrix-Logarithmus von *Zahl2* zur Basis *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Logarithmus jedes Elements zur Basis *Zahl2*. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Wenn das Basisargument weggelassen wird, wird 10 als Basis verwendet.

$$\log_{10} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 0.795387 + 0.753438 \cdot i & 0.003993 - 0.6474 \cdot i \\ 0.194895 - 0.315095 \cdot i & 0.462485 + 0.2707 \cdot i \\ -0.115909 - 0.904706 \cdot i & 0.488304 + 0.7774 \cdot i \end{pmatrix}$$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Logistic

Katalog >

Logistic *X, Y[, Häuf] [, Kategorie, Mit]*

Berechnet die logistische Regression $y = \frac{c}{(1+a \cdot e^{-bx})}$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LogisticD

Katalog > 

LogisticD *X* [, [Iterationen], [*Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die logistische Regression $y = (c/(1+a \cdot e^{-bx})+d)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf* unter Verwendung einer bestimmten Anzahl von *Iterationen*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert.
(Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Iterationen ist ein optionaler Wert, der angibt, wie viele Lösungsversuche maximal stattfinden. Bei Auslassung wird 64 verwendet. Größere Werte führen in der Regel zu höherer Genauigkeit, aber auch zu längeren Ausführungszeiten, und umgekehrt.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $c/(1+a \cdot e^{-bx})+d)$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

Loop (Schleife)

Loop
Block
EndLoop

Führt die in *Block* enthaltenen Anweisungen wiederholt aus. Beachten Sie, dass dies eine Endlosschleife ist. Beenden Sie sie, indem Sie die Anweisung **Goto** oder **Exit** in *Block* ausführen.

Block ist eine Folge von Anweisungen, die durch das Zeichen ":" voneinander getrennt sind.

```
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

Hinweis zum Eingeben des Beispiels:
 Anweisungen für die Eingabe von
 mehrzeiligen Programm- und
 Funktionsdefinitionen finden Sie im
 Abschnitt „Calculator“ des
 Produkthandbuchs.

LU (Untere/obere Matrixzerlegung)

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

Berechnet die Doolittle LU-Zerlegung (LR-Zerlegung) einer reellen oder komplexen Matrix. Die untere (bzw. linke) Dreiecksmatrix ist in *lMatrix* gespeichert, die obere (bzw. rechte) Dreiecksmatrix in *uMatrix* und die Permutationsmatrix (in welcher der bei der Berechnung vorgenommene Zeilentausch dokumentiert ist) in *pMatrix*.

$$lMatrix \cdot uMatrix = pMatrix \cdot Matrix$$

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

- Wenn Sie **ctrl** **enter** verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 \cdot \max(\dim(Matrix)) \cdot \text{rowNorm}(Matrix)$

Der **LU**-Faktorisierungsalgorithmus verwendet partielle Pivotisierung mit Zeilentausch.

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

M

mat►list() (Matrix in Liste)

mat►list(Matrix)⇒Liste

Gibt eine Liste zurück, die mit den Elementen aus *Matrix* gefüllt wurde. Die Elemente werden Zeile für Zeile aus *Matrix* kopiert.

Katalog >

mat►list([1 2 3])	{1,2,3}
[1 2 3] → m1	[1 2 3]
mat►list(m1)	{1,2,3,4,5,6}

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **mat@>list(...)** eintippen.

max() (Maximum)

max(Wert1, Wert2)⇒Ausdruck

max(Liste1, Liste2)⇒Liste

max(Matrix1, Matrix2)⇒Matrix

Gibt das Maximum der beiden Argumente zurück. Wenn die Argumente zwei Listen oder Matrizen sind, wird eine Liste bzw. Matrix zurückgegeben, die den Maximalwert für jedes entsprechende Elementpaar enthält.

Katalog >

max{2.3,1.4}	2.3
max{{1,2},{-4,3}}	{1,3}

max(Liste)⇒Ausdruck

Gibt das größte Element von *Liste* zurück.

max(Matrix1)⇒Matrix

Gibt einen Zeilenvektor zurück, der das größte Element jeder Spalte von *Matrix1* enthält.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

max{{0,1,-7,1.3,0.5})	1.3
-----------------------	-----

max[[1 -3 7] [-4 0 0.3]]	[1 0 7]
-----------------------------	---------

Hinweis: Siehe auch **min()**.

mean() (Mittelwert)

Katalog >

mean(Liste[, Häufigkeitsliste])⇒Ausdruck

Gibt den Mittelwert der Elemente in *Liste* zurück.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

mean(MatrixI[, Häufigkeitsmatrix])
⇒Matrix

Ergibt einen Zeilenvektor aus den Mittelwerten aller Spalten in *MatrixI*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *MatrixI* in der gegebenen Reihenfolge entsprechend.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

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

Im Vektorformat kartesisch:

mean $\begin{Bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{Bmatrix}$	$\begin{bmatrix} -0.133333 & 0.833333 \end{bmatrix}$
mean $\begin{Bmatrix} 1 & 0 \\ 5 & \\ -1 & 3 \\ 2 & -1 \\ 5 & 2 \end{Bmatrix}$	$\begin{bmatrix} -\frac{2}{15} & \frac{5}{6} \end{bmatrix}$
mean $\begin{Bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{Bmatrix}, \begin{Bmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{Bmatrix}$	$\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

median() (Median)

Katalog >

median(Liste[, freqList])⇒Ausdruck

median({0.2,0.1,-0.3,0.4})	0.2
----------------------------	-----

Gibt den Medianwert der Elemente in *Liste* zurück.

Jedes *freqList*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

median(MatrixI[, freqMatrix])⇒Matrix

Gibt einen Zeilenvektor zurück, der die Medianwerte der einzelnen Spalten von *MatrixI* enthält.

Jedes *freqMatrix*-Element gewichtet die Elemente von *MatrixI* in der gegebenen Reihenfolge entsprechend.

median $\begin{Bmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{Bmatrix}$	$\begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$
--	--

Hinweise:

- Alle Elemente der Liste bzw. der Matrix müssen zu Zahlen vereinfachbar sein.
- Leere (ungültige) Elemente in der Liste

oder Matrix werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

MedMed

MedMed $X, Y [, Häuf] [, Kategorie, Mit]$

Berechnet die Median-Median-Liniey = $(m \cdot x + b)$ auf Listen X und Y mit der Häufigkeit $Häuf$. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden X - und Y -Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Median-Median-Linien-Gleichung: $m \cdot x + b$
stat.m, stat.b	Modellkoeffizienten
stat.Resid	Residuen von der Median-Median-Linie

Ausgabeveriable	Beschreibung
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

mid() (Teil-String)

Katalog > 

mid(Quellstring, Start[, Anzahl]) \Rightarrow String

Gibt *Anzahl* Zeichen aus der Zeichenkette *Quellstring* ab dem Zeichen mit der Nummer *Start* zurück.

Wird *Anzahl* weggelassen oder ist sie größer als die Länge von *Quellstring*, werden alle Zeichen von *Quellstring* ab dem Zeichen mit der Nummer *Start* zurückgegeben.

Anzahl muss ≥ 0 sein. Bei *Anzahl* = 0 wird eine leere Zeichenkette zurückgegeben.

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

mid(Quellliste, Start [, Anzahl]) \Rightarrow Liste

Gibt *Anzahl* Elemente aus *Quellliste* ab dem Element mit der Nummer *Start* zurück.

Wird *Anzahl* weggelassen oder ist sie größer als die Dimension von *Quellliste*, werden alle Elemente von *Quellliste* ab dem Element mit der Nummer *Start* zurückgegeben.

Anzahl muss ≥ 0 sein. Bei *Anzahl* = 0 wird eine leere Liste zurückgegeben.

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(QuellstringListe, Start[, Anzahl]) \Rightarrow Liste

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

Gibt *Anzahl* Strings aus der Stringliste *QuellstringListe* ab dem Element mit der Nummer *Start* zurück.

min() (Minimum)

Katalog >

min(Wert1, Wert2)⇒Ausdruck

min{2,3,1,4} 1.4

min(Liste1, Liste2)⇒Liste

min{{1,2},{-4,3}} {-4,2}

min(Matrix1, Matrix2)⇒Matrix

Gibt das Minimum der beiden Argumente zurück. Wenn die Argumente zwei Listen oder Matrizen sind, wird eine Liste bzw. Matrix zurückgegeben, die den Minimalwert für jedes entsprechende Elementpaar enthält.

min(Liste)⇒Ausdruck

min{{0,1,-7,1,3,0,5}} -7

Gibt das kleinste Element von *Liste* zurück.

min(Matrix1)⇒Matrix

min[[1 -3 7] [-4 0 0.3]] [-4 -3 0.3]

Gibt einen Zeilenvektor zurück, der das kleinste Element jeder Spalte von *Matrix1* enthält.

Hinweis: Siehe auch **max()**.

mirr()

Katalog >

mirr

(

Finanzierungsrate

,*Reinvestitionsrate*,*CF0*,*CFListe*
[,*CFFreq*])

list1:={6000,-8000,2000,-3000}
{6000,-8000,2000,-3000}

list2:={2,2,2,1} {2,2,2,1}

mirr(4.65,12,5000,list1,list2) 13.41608607

Finanzfunktion, die den modifizierten internen Zinsfluss einer Investition zurückgibt.

Finanzierungsrate ist der Zinssatz, den Sie für die Cash-Flow-Beträge zahlen.

Reinvestitionsrate ist der Zinssatz, zu dem die Cash-Flows reinvestiert werden.

CF0 ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

CFListe ist eine Liste von Cash-Flow-Beträgen nach dem Anfangs-Cash-Flow *CF0*.

CFFreq ist eine optionale Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von *CFListe* ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzahlen < 10.000 sein.

Hinweis: Siehe auch **irr()**, Seite 82.

mod() (Modulo)

mod(*Wert1, Wert2*)⇒Ausdruck

mod(*Liste1, Liste2*)⇒Liste

mod(*Matrix1, Matrix2*)⇒Matrix

Gibt das erste Argument modulo das zweite Argument gemäß der folgenden Identitäten zurück:

$$\text{mod}(x, 0) = x$$

$$\text{mod}(x, y) = x - y \text{ floor}(x/y)$$

Ist das zweite Argument ungleich Null, ist das Ergebnis in diesem Argument periodisch. Das Ergebnis ist entweder Null oder besitzt das gleiche Vorzeichen wie das zweite Argument.

Sind die Argumente zwei Listen bzw. zwei Matrizen, wird eine Liste bzw. Matrix zurückgegeben, die den Modulus jedes Elementpaares enthält.

Hinweis: Siehe auch **remain()**, Seite 138

mod(7,0)	7
mod(7,3)	1
mod(-7,3)	2
mod(7,-3)	-2
mod(-7,-3)	-1
mod({12,-14,16},{9,7,-5})	{3,0,-4}

mRow() (Matrixzeilenoperation)

mRow(*Zahl, Matrix1, Index*)⇒Matrix

Gibt eine Kopie von *Matrix1* zurück, in der jedes Element der Zeile *Index* von *Matrix1* mit *Zahl* multipliziert ist.

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

mRowAdd() (Matrixzeilenaddition)**Katalog > **

mRowAdd(*Zahl, Matrix1, Index1, Index2*)
 \Rightarrow Matrix

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

Gibt eine Kopie von *Matrix1* zurück, wobei jedes Element in Zeile *Index2* von *Matrix1* ersetzt wird durch:

Zahl \times Zeile *Index1* + Zeile *Index2*

MultReg**Katalog > **

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

Berechnet die lineare Mehrfachregression der Liste *Y* für die Listen *X1, X2, ..., X10*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.b0, stat.b1, ...	Regressionskoeffizienten
stat.R ²	Multiples Bestimmtheitsmaß
stat.yList	\hat{y} List = $b_0+b_1 \cdot x_1+ \dots$
stat.Resid	Residuen von der Regression

MultRegIntervals**Katalog > **

MultRegIntervals *Y, X1[,X2[,X3,...[,X10]]],XWertListe[,KNiveau]*

Berechnet einen vorhergesagten y-Wert, ein Niveau-K-Vorhersageintervall für eine einzelne Beobachtung und ein Niveau-K-Confidenzintervall für die mittlere Antwort.

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert.
(Seite 162.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b0+b1 \cdot x1+b2 \cdot x2+ \dots$
stat.ŷ	Eine Punktschätzung: $\hat{y} = b0 + b1 \cdot x1 + \dots$ für <i>XWertListe</i>
stat.dfError	Fehler-Freiheitsgrade
stat.CLower, stat.CUpper	Konfidenzintervall für eine mittlere Antwort
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SE	Standardfehler der mittleren Antwort
stat.LowerPred,	Vorhersageintervall für eine einzelne Beobachtung
stat.UpperrPred	
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat.bList	Liste der Regressionskoeffizienten, {b0,b1,b2,...}
stat.Resid	Residuen von der Regression

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

Der lineare Mehrfachregressionstest berechnet eine lineare Mehrfachregression für die gegebenen Daten sowie die globale *F*-Teststatistik und *t*-Teststatistik für die Koeffizienten.

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert.
(Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgaben

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+\dots$
stat.F	Globale F -Testgröße
stat.PVal	Mit globaler F -Statistik verknüpfter P-Wert
stat.R ²	Multiples Bestimmtheitsmaß
stat.AdjR ²	Angepasster Koeffizient des multiplen Bestimmtheitsmaßes
stat.s	Standardabweichung des Fehlers
stat.DW	Durbin-Watson-Statistik; bestimmt, ob in dem Modell eine Autokorrelation erster Ordnung vorhanden ist
stat.dfReg	Regressions-Freiheitsgrade
stat.SSReg	Summe der Regressionsquadrate
stat.MSReg	Mittlere Regressionsstreuung
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittleres Fehlerquadrat
stat.bList	{ b_0, b_1, \dots } Liste der Koeffizienten
stat.tList	Liste der t-Testgrößen, eine für jeden Koeffizienten in b-Liste
stat.PList	Liste der P-Werte für jede t-Testgröße
stat.SEList	Liste der Standardfehler für Koeffizienten in b-Liste
stat.gList	\hat{y} List = $b_0+b_1 \cdot x_1+\dots$
stat.Resid	Residuen von der Regression
stat.sResid	Standardisierte Residuen; wird durch Division eines Residiums durch die Standardabweichung ermittelt
stat.CookDist	Cookscher Abstand; Maß für den Einfluss einer Beobachtung auf der Basis von Residuum und Hebelwert
stat.Leverage	Maß für den Abstand der Werte der unabhängigen Variable von den Mittelwerten (Hebelwerte)

nand

Tasten

BoolescherAusdr1 nand BoolescherAusdr2
ergibt Boolescher Ausdruck

BoolescheListe1 nand BoolescheListe2
ergibt Boolesche Liste

BoolescheMatrix1 nand BoolescheMatrix2
ergibt Boolesche Matrix

Gibt die Negation einer logischen **and** Operation auf beiden Argumenten zurück.
Gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Ganzzahl1 nand Ganzzahl2 \Rightarrow *Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer **nand**-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 64-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 0, wenn beide Bits 1 sind; anderenfalls ist das Ergebnis 1. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 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() (Kombinationen)

Katalog >

nCr(Wert1, Wert2) \Rightarrow *Ausdruck*

nCr(z,3) z=5	10
nCr(z,3) z=6	20

nCr() (Kombinationen)

Katalog >

Für ganzzahlige Wert1 und Wert2 mit $Wert1 \geq Wert2 \geq 0$ ist nCr() die Anzahl der Möglichkeiten, Wert1 Elemente aus Wert2 Elementen auszuwählen (auch als Binomialkoeffizient bekannt).

nCr(Wert, 0)⇒1

nCr(Wert, negGanzzahl)⇒0

nCr(Wert, posGanzzahl)⇒ Wert · (Wert-1)...(Wert-posGanzzahl+1)/posGanzzahl!

nCr(Wert, keineGanzzahl)⇒Ausdruck!/((Wert-keineGanzzahl)! · keineGanzzahl!)

nCr(Liste1, Liste2)⇒Liste

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

Gibt eine Liste von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

nCr(Matrix1, Matrix2)⇒Matrix

Gibt eine Matrix von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Matrizen zurück. Die Argumente müssen Matrizen gleicher Größe sein.

nCr([6 5],[2 2]) [15 10]
[4 3][2 2]

nDerivative()

Katalog >

nDerivative(Ausdr1,Var=Wert [,Ordnung])⇒Wert

nDerivative(|x|,x=1) 1

nDerivative(Ausdr1,Var[,Ordnung]) | Var=Wert⇒Wert

nDerivative(|x|,x)|x=0 undef

nDerivative(sqrt(x-1),x)|x=1 undef

Gibt die numerische Ableitung zurück, berechnet durch automatische Ableitungsmethoden.

Wenn Wert angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

nDerivative()

Katalog >

Wenn die Variable *Var* keinen Zahlenwert enthält, müssen Sie *Wert* angeben.

Ordnung der Ableitung muss **1** oder **2** sein.

Hinweis: Der Algorithmus von nDerivative() hat eine Einschränkung: Er arbeitet den nicht-vereinfachten Ausdruck rekursiv ab und berechnet dabei den numerischen Wert der ersten (und ggf. der zweiten) Ableitung sowie die Auswertung jedes Unterausdrucks. Dies kann zu unerwarteten Ergebnissen führen.

Hierzu rechts ein Beispiel. Die erste Ableitung von $x \cdot (x^2+x)^{1/3}$ bei $x=0$ ist gleich 0. Nun ist allerdings die erste Ableitung des Unterausdrucks $(x^2+x)^{1/3}$ bei $x=0$ nicht definiert. Dieser Wert wird gleichzeitig jedoch verwendet, um die Ableitung des Gesamtausdrucks zu berechnen. Daher meldet nDerivative() das Ergebnis als nicht definiert und zeigt eine Warnmeldung an.

Wenn Sie bei der Arbeit auf diese Einschränkung stoßen, prüfen Sie die Lösung grafisch. Ggf. können Sie es auch mit centralDiff() probieren.

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

newList() (Neue Liste)

Katalog >

newList(AnzElemente)⇒Liste

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

Gibt eine Liste der Dimension AnzElemente zurück. Jedes Element ist Null.

newMat() (Neue Matrix)

Katalog >

newMat(AnzZeil, AnzSpalt)⇒Matrix

newMat(2,3) [0 0 0
0 0 0]

Gibt eine Matrix der Dimension AnzZeil mal AnzSpalt zurück, wobei die Elemente Null sind.

nfMax() (Numerisches Funktionsmaximum)

Katalog >

nfMax(Ausdr, Var)⇒Wert

nfMax(Ausdr, Var, UntereGrenze)⇒Wert

nfMax(Ausdr, Var, UntereGrenze, ObereGrenze)⇒Wert

**nfMax(Ausdr, Var) | UntereGrenze≤Var
≤ObereGrenze⇒Wert**

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Maximum von *Ausdr* auftritt.

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Intervall $[UntereGrenze, ObereGrenze]$ für das lokale Maximum.

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

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

nfMin() (Numerisches Funktionsminimum)

Katalog >

nfMin(Ausdr, Var)⇒Wert

nfMin(Ausdr, Var, UntereGrenze)⇒Wert

nfMin(Ausdr, Var, UntereGrenze, ObereGrenze)⇒Wert

**nfMin(Ausdr, Var) | UntereGrenze≤Var
≤ObereGrenze⇒Wert**

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Minimum von *Ausdr* auftritt.

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Intervall $[UntereGrenze, ObereGrenze]$ für das lokale Minimum.

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

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

nInt() (Numerisches Integral)

Katalog >

nInt(Ausdr1, Var, Untere, Obere)⇒Ausdruck

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

nInt() (Numerisches Integral)

Katalog > 

Wenn der Integrand *Ausdr1* außer *Var* keine anderen Variablen enthält und wenn *Untere* und *Obere* Konstanten oder positiv ∞ oder negativ ∞ sind, gibt **nInt()** eine Näherung für $\int(Ausdr1, Var, Untere, Obere)$ zurück. Diese Näherung ist der gewichtete Durchschnitt von Stichprobenwerten des Integranden im Intervall *Untere*<*Var*<*Obere*.

Das Berechnungsziel sind sechs signifikante Stellen. Der angewendete Algorithmus beendet die Weiterberechnung, wenn das Ziel hinreichend erreicht ist oder wenn weitere Stichproben wahrscheinlich zu keiner sinnvollen Verbesserung führen.

Wenn es scheint, dass das Berechnungsziel nicht erreicht wurde, wird die Meldung "Zweifelhafte Genauigkeit" angezeigt.

Sie können **nInt()** verschachteln, um mehrere numerische Integrationen durchzuführen. Die Integrationsgrenzen können von außerhalb liegenden Integrationsvariablen abhängen.

nInt(cos(x),x,-π,π+1.e-12) -1.04144e-12

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

nom()

Katalog > 

nom(Effektivzins,CpY)=Wert

nom(5.90398,12) 5.75

Finanzfunktion zur Umrechnung des jährlichen Effektivzinssatzes *Effektivzins* in einen Nominalzinssatz, wobei *CpY* als Anzahl der Verzinsungsperioden pro Jahr gegeben ist.

Effektivzins muss eine reelle Zahl sein und *CpY* muss eine reelle Zahl > 0 sein.

Hinweis: Siehe auch **eff()**, Seite 48.

nor

  Tasten

BoolescherAusdr1norBoolescherAusdr2
ergibt Boolescher Ausdruck

BoolescheListe1norBoolescheListe2
ergibt Boolesche Liste

BoolescheMatrix1 nor *BoolescheMatrix2*
ergibt Boolesche Matrix

Gibt die Negation einer logischen **or** Operation auf beiden Argumenten zurück.
Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Ganzzahl1 nor *Ganzzahl2* \Rightarrow *Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer **nor**-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 64-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn beide Bits 1 sind; anderenfalls ist das Ergebnis 0. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

3 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

norm()

Katalog >

norm(Matrix) \Rightarrow Ausdruck

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

norm(Vektor) \Rightarrow Ausdruck

norm([1 2]) 2.23607

Gibt die Frobeniusnorm zurück.

norm([1
2]) 2.23607

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog >

**normCdf(untereGrenze, obereGrenze[, μ ,
[σ]])** \Rightarrow Zahl, wenn untereGrenze und

normCdf()
(Normalverteilungswahrscheinlichkeit)

Katalog >

obereGrenze Zahlen sind, *Liste*, wenn
untereGrenze und *obereGrenze* Listen sind

Berechnet die Normalverteilungswahrscheinlichkeit zwischen *untereGrenze* und *obereGrenze* für die angegebenen μ (Standard = 0) und σ (Standard = 1).

Für $P(X \leq obereGrenze)$ setzen Sie
 $untereGrenze = -9E999$.

normPdf() (Wahrscheinlichkeitsdichte)

Katalog > 

normPdf($XWert$ [, μ [, σ]])⇒*Zahl*, wenn
XWert eine Zahl ist, *Liste*, wenn *XWert*
eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion für die Normalverteilung an einem bestimmten X Wert für die vorgegebenen μ und σ .

not (nicht)

Katalog > 

not

BoolescherAusdr1 \Rightarrow *BoolescherAusdruck*

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des Arguments zurück.

not Ganzzahl \Rightarrow Ganzzahl

Gibt das Einerkomplement einer reellen ganzen Zahl zurück. Intern wird *Ganzzahl1* in eine 32-Bit-Dualzahl mit Vorzeichen umgewandelt. Für das Einerkomplement werden die Werte aller Bits umgekehrt (so dass 0 zu 1 wird und umgekehrt). Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen mit jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix wird die ganze Zahl als dezimal behandelt (Basis 10).

not (2≥3)	true
not 0hB0►Base16	0hFFFFFFFFFFFFFFF4F
not not 2	2

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O.

not 0h7AC36 0hFFFFFFFFFFFF853C9

Im Bin-Modus:

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **Base2**, Seite 17.

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix **0b** wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

nPr() (Permutationen)

nPr(Wert1, Wert2)⇒Ausdruck

Für ganzzahlige *Wert1* und *Wert2* mit $Wert1 \geq Wert2 \geq 0$ ist **nPr()** die Anzahl der Möglichkeiten, *Wert1* Elemente unter Berücksichtigung der Reihenfolge aus *Wert2* Elementen auszuwählen.

nPr($z, 3 z=5$)	60
nPr($z, 3 z=6$)	120
nPr($\{5, 4, 3\}, \{2, 4, 2\}$)	{20, 24, 6}
nPr($\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$)	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$

nPr(Wert, 0)⇒1

nPr(Wert, negGanzzahl)⇒1/((Wert+1) · (Wert+2) ... (Wert-negGanzzahl))

nPr(Wert, posGanzzahl)⇒Wert · (Wert-1) ... (Wert-posGanzzahl+1)

nPr(Wert, keineGanzzahl)⇒Wert! / (Wert-keineGanzzahl)!

nPr(Liste1, Liste2)⇒Liste

Gibt eine Liste der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

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

nPr(Matrix1, Matrix2)⇒Matrix

Gibt eine Matrix der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Matrizen zurück. Die Argumente müssen Matrizen gleicher Größe sein.

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

npv()

Katalog >

npv(Zinssatz,CFO,CFListe[CFFreq])

Finanzfunktion zur Berechnung des Nettoarwerts; die Summe der Barwerte für die Bar-Zuflüsse und -Abflüsse. Ein positives Ergebnis für npv zeigt eine rentable Investition an.

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

Zinssatz ist der Satz, zu dem die Cash-Flows (der Geldpreis) für einen Zeitraum.

CF0 ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

CFListe ist eine Liste der Cash-Flow-Beträge nach dem anfänglichen Cash-Flow CF0.

CFFreq ist eine Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von CFListe ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzahlen < 10.000 sein.

nSolve() (Numerische Lösung)

Katalog >

nSolve(Gleichung,Var [=Schätzwert]) \Rightarrow Zahl oder Fehler_String

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

nSolve(Gleichung,Var [=Schätzwert],UntereGrenze) \Rightarrow Zahl oder Fehler_String

nSolve($x^2 = 4, x = -1$) -2.

nSolve(Gleichung,Var [=Schätzwert],UntereGrenze,ObereGrenze) \Rightarrow Zahl oder Fehler_String

nSolve($x^2 = 4, x = 1$) 2.

nSolve(Gleichung,Var [=Schätzwert]) | UntereGrenze \leq Var \leq ObereGrenze \Rightarrow Zahl oder Fehler_String

Hinweis: Existieren mehrere Lösungen, können Sie mit Hilfe einer Schätzung eine bestimmte Lösung suchen.

Ermittelt iterativ eine reelle numerische Näherungslösung von Gleichung für deren eine Variable. Geben Sie die Variable an als:

Variable

– oder –

Variable = reelle Zahl

Beispiel: x ist gültig und x=3 ebenfalls.

nSolve() versucht entweder einen Punkt zu ermitteln, wo der Unterschied zwischen tatsächlichem und erwartetem Wert Null ist oder zwei relativ nahe Punkte, wo der Restfehler entgegengesetzte Vorzeichen besitzt und nicht zu groß ist. Wenn nSolve() dies nicht mit einer kleinen Anzahl von Versuchen erreichen kann, wird die Zeichenkette "Keine Lösung gefunden" zurückgegeben.

nSolve($x^2+5 \cdot x - 25 = 9, x$) $x < 0$	-8.84429
nSolve($\frac{(1+r)^{24}-1}{r} = 26, r$) $r > 0$ and $r < 0.25$	0.006886
nSolve($x^2 = -1, x$)	"No solution found"

O

OneVar (Eine Variable)

**OneVar [1,]X1,[Häufigkeit]
[,Kategorie,Mit]]**

OneVar [n,]X1,X2[X3[,...,X20]]]

Berechnet die 1-Variablenstatistik für bis zu 20 Listen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

Die *X*-Argumente sind Datenlisten.

Häufigkeit ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häufigkeit* gibt die Häufigkeit für jeden entsprechenden *X*-Wert an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen *X*, *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Ein leeres (ungültiges) Element in einer der Listen *X1* bis *X20* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Ausgabevariable	Beschreibung
stat. \bar{x}	Mittelwert der x-Werte
stat. Σx	Summe der x-Werte
stat. Σx^2	Summe der x^2 -Werte
stat.sx	Stichproben-Standardabweichung von x
stat.x	Populations-Standardabweichung von x
stat.n	Anzahl der Datenpunkte
stat.MinX	Minimum der x-Werte
stat.Q ₁ X	1. Quartil von x
stat.MedianX	Median von x
stat.Q ₃ X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.SSX	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert

or (oder)

BoolescherAusdr1 **or** *BoolescherAusdr2*
ergibt Boolescher Ausdruck

BoolescheListe1 **or** *BoolescheListe2* ergibt
Boolesche Liste

BoolescheMatrix1 **or** *BoolescheMatrix2*
ergibt Boolesche Matrix

or (oder)

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des ursprünglichen Terms zurück.

Gibt „wahr“ zurück, wenn ein Ausdruck oder beide Ausdrücke zu „wahr“ ausgewertet werden. Gibt nur dann „falsch“ zurück, wenn beide Ausdrücke „falsch“ ergeben.

Hinweis: Siehe xor.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Ganzzahl1 or *Ganzzahl2* \Rightarrow *Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer or-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn eines der Bits 1 ist; das Ergebnis ist nur dann 0, wenn beide Bits 0 sind. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **►Base2**, Seite 17.

Hinweis: Siehe xor.

Define $g(x)=\text{Func}$	<i>Done</i>
If $x \leq 0$ or $x \geq 5$	
Goto end	
Return $x \cdot 3$	
Lbl end	
EndFunc	

$g(3)$	9
$g(0)$	<i>A function did not return a value</i>

Im Hex-Modus:

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

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 or 0b100	0b100101
-------------------	----------

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

ord() (Numerischer Zeichencode)

Katalog >

ord(String)⇒Ganzzahl

ord(Liste1)⇒Liste

Gibt den Zahlenwert (Code) des ersten Zeichens der Zeichenkette *String* zurück.
Handelt es sich um eine Liste, wird der Code des ersten Zeichens jedes Listenelements zurückgegeben.

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

P

►Rx() (Kartesische x-Koordinate)

Katalog >

►Rx(rAusdr, θAusdr)⇒Ausdruck

►Rx(rListe, θListe)⇒Liste

►Rx(rMatrix, θMatrix)⇒Matrix

Gibt die äquivalente x-Koordinate des Paares (r, θ) zurück.

Hinweis: Das θ -Argument wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein Ausdruck, können Sie $^\circ$, g oder $'$ benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **P@>Rx (...)** eintippen.

Im Bogenmaß-Modus:

►Rx(4,60°)	2.
►Rx({ -3,10,1.3 }, { $\frac{\pi}{3}, \frac{\pi}{4}, 0 $ })	{ -1.5,7.07107,1.3 }

►Ry() (Kartesische y-Koordinate)

Katalog >

►Ry(rWert, θWert)⇒Wert

Im Bogenmaß-Modus:

►Ry(rListe, θListe)⇒Liste

►Ry(4,60°)	3.4641
►Ry({ -3,10,1.3 }, { $\frac{\pi}{3}, \frac{\pi}{4}, 0 $ })	{ -2.59808, 7.07107,0, }

►Ry(rMatrix, θMatrix)⇒Matrix

Gibt die äquivalente y-Koordinate des Paares (r, θ) zurück.

Hinweis: Das θ -Argument wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **P@>Ry (...)** eintippen.

PassErr (Ügebefehl)

PassErr

Übergibt einen Fehler an die nächste Stufe.

Wenn die Systemvariable *Fehlercode* (*errCode*) Null ist, tut **PassErr** nichts.

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** verwenden. Wenn der Fehler verarbeitet oder ignoriert werden soll, verwenden Sie **ClrErr**. Wenn nicht bekannt ist, was mit dem Fehler zu tun ist, verwenden Sie **PassErr**, um ihn an den nächsten Error Handler zu übergeben. Wenn keine weiteren **Try...Else...EndTry** Error Handler unerledigt sind, wird das Fehlerdialogfeld als normal angezeigt.

Hinweis: Siehe auch **LöFehler**, Seite 24, und **Versuche**, Seite 175.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

piecewise() (Stückweise)

piecewise(Ausdr1 [, Bedingung1 [, Ausdr2 [, Bedingung2 [, ...]]]])

Gibt Definitionen für einestückweise definierte Funktion in Form einer Liste zurück. Sie können auch mit Hilfe einer Vorlage stückweise Definitionen erstellen.

Hinweis: Siehe auch **Vorlage Stückweise**, Seite 3.

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$	Done
$p(1)$	1
$p(-1)$	undef

poissCdf

$(\lambda, \text{untereGrenze}, \text{obereGrenze}) \Rightarrow \text{Zahl}$,
 wenn *untereGrenze* und *obereGrenze*
 Zahlen sind, *Liste*, wenn *untereGrenze* und
obereGrenze Listen sind

poissCdf($\lambda, \text{obereGrenze}$) (für $P(0 \leq X \leq \text{obereGrenze}) \Rightarrow \text{Zahl}$, wenn *obereGrenze*
 eine Zahl ist, Liste, wenn *obereGrenze* eine
 Liste ist)

Berechnet die kumulative
 Wahrscheinlichkeit für die diskrete Poisson-
 Verteilung mit dem vorgegebenen
 Mittelwert λ .

Für $P(X \leq \text{obereGrenze})$ setzen Sie
untereGrenze = 0

poissPdf($\lambda, XWert$) $\Rightarrow \text{Zahl}$, wenn *XWert* eine
 Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeit für die
 diskrete Poisson-Verteilung mit dem
 vorgegebenen Mittelwert λ .

►Polar

Vektor ►Polar

[1 3.] ►Polar [3.16228 ∠ 71.5651]

Hinweis: Sie können diesen Operator über
 die Tastatur Ihres Computers eingeben,
 indem Sie @>**Polar** eintippen.

Zeigt *Vektor* in Polarform [$r \angle \theta$] an. Der
 Vektor muss die Dimension 2 besitzen und
 kann eine Zeile oder eine Spalte sein.

Hinweis: ►Polar ist eine
 Anzeigeformatanweisung, keine
 Konvertierungsfunktion. Sie können sie nur
 am Ende einer Eingabezeile benutzen, und
 sie nimmt keine Aktualisierung von *ans* vor.

Hinweis: Siehe auch ►Rect, Seite 135.

komplexerWert ►Polar

Im Bogenmaß-Modus:

►Polar

Katalog >

Zeigt komplexerVektor in Polarform an.

- Der Grad-Modus für Winkel gibt $(r \angle \theta)$ zurück.
- Der Bogenmaß-Modus für Winkel gibt $r e^{i\theta}$ zurück.

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

komplexerWert kann jede komplexe Form haben. Eine $r e^{i\theta}$ -Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Im Neugrad-Modus:

$$(4 \cdot i) \blacktriangleright \text{Polar} \quad (4 \angle 100.)$$

Hinweis: Für eine Eingabe in Polarform müssen Klammern $(r \angle \theta)$ verwendet werden.

Im Grad-Modus:

$$(3+4 \cdot i) \blacktriangleright \text{Polar} \quad (5 \angle 53.1301)$$

polyEval() (Polynom auswerten)

Katalog >

polyEval(Liste1, Ausdr1) \Rightarrow Ausdruck

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

polyEval(Liste1, Liste2) \Rightarrow Ausdruck

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

Interpretiert das erste Argument als Koeffizienten eines nach fallenden Potenzen geordneten Polynoms und gibt das Polynom bezüglich des zweiten Arguments zurück.

polyRoots()

Katalog >

polyRoots(Poly, Var) \Rightarrow Liste

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

polyRoots(KoeffListe) \Rightarrow Liste

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

Die erste Syntax **polyRoots(Poly, Var)** gibt eine Liste mit reellen Wurzeln des Polynoms *Poly* bezüglich der Variablen *Var* zurück. Wenn keine reellen Wurzeln existieren, wird eine leere Liste zurückgegeben: {}.

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

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

Poly muss dabei ein Polynom in entwickelter Form in einer Variablen sein.
Verwenden Sie keine nicht-entwickelten Formen wie z. B. $y^2 \cdot y + 1$ oder $x \cdot x + 2 \cdot x + 1$

Die zweite Syntax **polyRoots(KoeffListe)** liefert eine Liste mit reellen Wurzeln für die Koeffizienten in *KoeffListe*.

Hinweis: Siehe auch **cPolyRoots()**, Seite 33.

PowerReg

PowerReg *X, Y [, Häuf] [, Kategorie, Mit]*

Berechnet die Potenzregression $y = (a \cdot x^b)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot (x)^b$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten ($\ln(x)$, $\ln(y)$)
stat.Resid	Mit dem Potenzmodell verknüpfte Residuen
stat.ResidTrans	Residuen für die lineare Anpassung transformierter Daten
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

Prgm

Katalog > 

Prgm
Block
EndPrgm

Vorlage zum Erstellen eines benutzerdefinierten Programms. Muss mit dem Befehl **Definiere (Define)**, **Definiere LibPub (Define LibPub)** oder **Definiere LibPriv (Define LibPriv)** verwendet werden.

Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen ":" voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

GCD berechnen und Zwischenergebnisse anzeigen.

```
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

Product (Π) (Produkt)Siehe $\Pi()$, Seite 207.**product() (Produkt)****Katalog >** **product(Liste[, Start[, Ende]])** \Rightarrow Ausdruck

Gibt das Produkt der Elemente von *Liste* zurück. *Start* und *Ende* sind optional. Sie geben einen Elementebereich an.

product(Matrix1[, Start[, Ende]]) \Rightarrow Matrix

Gibt einen Zeilenvektor zurück, der die Produkte der Elemente aus den Spalten von *Matrix1* enthält. *Start* und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

product({1,2,3,4})

24

product({4,5,8,9},2,3)

40

$\text{product} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	$[28 \quad 80 \quad 162]$
--	---------------------------

$\text{product} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, 1,2$	$[4 \quad 10 \quad 18]$
---	-------------------------

propFrac() (Echter Bruch)**Katalog >** **propFrac(Wert1[, Var])** \Rightarrow Wert

propFrac(rationale_Wert) gibt *rationale_Wert* als Summe einer ganzen Zahl und eines Bruchs zurück, der das gleiche Vorzeichen besitzt und dessen Nenner größer ist als der Zähler.

propFrac(rationaler_Ausdruck,Var) gibt die Summe der echten Brüche und ein Polynom bezüglich *Var* zurück. Der Grad von *Var* im Nenner übersteigt in jedem echten Bruch den Grad von *Var* im Zähler. Gleichartige Potenzen von *Var* werden zusammengefasst. Die Terme und Faktoren werden mit *Var* als der Hauptvariablen sortiert.

propFrac($\frac{4}{3}$) $1\frac{1}{3}$ propFrac($\frac{-4}{3}$) $-1\frac{1}{3}$

Wird *Var* weggelassen, wird eine Entwicklung des echten Bruchs bezüglich der wichtigsten Hauptvariablen vorgenommen. Die Koeffizienten des Polynomteils werden dann zuerst bezüglich der wichtigsten Hauptvariablen entwickelt usw.

Mit der Funktion **propFrac()** können Sie gemischte Brüche darstellen und die Addition und Subtraktion bei gemischten Brüchen demonstrieren.

$\text{propFrac}\left(\frac{11}{7}\right)$	$1\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2\frac{29}{44}$

Q**QR**

QR Matrix, qMatrix, rMatrix[, Tol]

Berechnet die Householdersche QR-Faktorisierung einer reellen oder komplexen Matrix. Die sich ergebenden Q- und R-Matrizen werden in den angegebenen *Matrix* gespeichert. Die Q-Matrix ist unitär. Bei der R-Matrix handelt es sich um eine obere Dreiecksmatrix.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

Die Fließkommazahl (9,) in m1 bewirkt, dass das Ergebnis in Fließkommaform berechnet wird.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
QR m1,qm,rm	Done
qm	$\begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$
rm	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

- Wenn Sie **ctrl enter** verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:

`5E-14 ·max(dim(Matrix)) ·rowNorm
(Matrix)`

Die QR-Faktorisierung wird anhand von Householderschen Transformationen numerisch berechnet. Die symbolische Lösung wird mit dem Gram-Schmidt-Verfahren berechnet. Die Spalten in *qMatName* sind die orthonormalen Basisvektoren, die den durch *Matrix* definierten Raum aufspannen.

QuadReg

`QuadReg X, Y [, Häuf] [, Kategorie, Mit]`

Berechnet die quadratische polynomiale Regression $y = a \cdot x^2 + b \cdot x + c$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert.
(Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^2 + b \cdot x + c$
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

QuartReg

Katalog > 

QuartReg X, Y [, Häuf] [, Kategorie, Mit]

Berechnet die polynomiale Regression vierter Ordnung $= a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoeffizienten
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

R**R►Pθ()**Katalog > 

R►Pθ (xWert, yWert) ⇒ Wert
R►Pθ (xListe, yListe) ⇒ Liste
R►Pθ (xMatrix, yMatrix) ⇒ Matrix

Gibt die äquivalente θ-Koordinate des Paares (x,y) zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Im Grad-Modus:

R►Pθ(2,2) 45.

Im Neugrad-Modus:

R►Pθ(2,2) 50.

Im Bogenmaß-Modus:

R►Pθ()

Katalog >

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie R@Ptheta (...) eintippen.

R►Pθ(3,2)	0.588003
R►Pθ([3 -4 2],[0 π/4 1.5])	[0. 2.94771 0.643501]

R►Pr()

Katalog >

R►Pr(xWert,yWert) ⇒ Wert
R►Pr(xListe,yListe) ⇒ Liste
R►Pr(xMatrix,yMatrix) ⇒ Matrix

Gibt die äquivalente r-Koordinate des Paares (x,y) zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie R@Pr (...) eintippen.

Im Bogenmaß-Modus:

R►Pr(3,2)	3.60555
R►Pr([3 -4 2],[0 π/4 1.5])	[3 4.07638 5/2]

► Rad

Katalog >

Wert1►Rad ⇒ Wert

Im Grad-Modus:

(1.5)►Rad	(0.02618) ^r
-----------	------------------------

Wandelt das Argument ins Bogenmaß um.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @Rad eintippen.

Im Neugrad-Modus:

(1.5)►Rad	(0.023562) ^r
-----------	-------------------------

rand() (Zufallszahl)

Katalog >

rand() ⇒ Ausdruck
rand(#Trials) ⇒ List

Setzt Ausgangsbasis für Zufallszahlengenerierung.

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

rand() gibt einen Zufallswert zwischen 0 und 1 zurück.

rand(#Trials) gibt eine Liste zurück, die #Trials Zufallswerte zwischen 0 und 1 enthält.

randBin() (Zufallszahl aus Binomialverteilung)

Katalog >

randBin(*n, p*) \Rightarrow Ausdruck

randBin(*n, p, #Trials*) \Rightarrow Liste

randBin(*n, p*) gibt eine reelle Zufallszahl aus einer angegebenen Binomialverteilung zurück.

randBin(*n, p, #Trials*) gibt eine Liste mit $\#Trials$ reellen Zufallszahlen aus einer angegebenen Binomialverteilung zurück.

randBin(80,0.5)

46.

randBin(80,0.5,3)

{ 43.,39.,41. }

randInt() (Ganzzahlige Zufallszahl)

Katalog >

**randInt
(*lowBound,upBound*)**

\Rightarrow Ausdruck

randInt

$(\text{lowBound}, \text{upBound}$,
 $\#\text{Trials})$ \Rightarrow Liste

**randInt
(*lowBound,upBound*)**

gibt eine ganzzahlige Zufallszahl innerhalb der durch

UntereGrenze

(lowBound) und

ObereGrenze

(upBound)

festgelegten

Grenzen zurück.

randInt

(

lowBound

,*upBound*,*#Trials*

gibt eine Liste mit $\#Trials$ ganzzahligen Zufallszahlen

innerhalb des

festgelegten

Bereichs zurück.

randMat() (Zufallsmatrix)**Katalog >** **randMat(*AnzZeil*, *AnzSpalt*)** \Rightarrow Matrix

Gibt eine Matrix der angegebenen Dimension mit ganzzahligen Werten zwischen -9 und 9 zurück.

Beide Argumente müssen zu ganzen Zahlen vereinfachbar sein.

RandSeed 1147

Done

randMat(3,3)

8	-3	6
-2	3	-6
0	4	-6

Hinweis: Die Werte in dieser Matrix ändern sich mit jedem Drücken von **enter**.**randNorm() (Zufallsnorm)****Katalog >** **randNorm(μ , σ)** \Rightarrow Ausdruck**randNorm(μ , σ , #Trials)** \Rightarrow List**randNorm(μ , σ)** gibt eine Dezimalzahl aus der Gaußschen Normalverteilung zurück. Dies könnte eine beliebige reelle Zahl sein, die Werte konzentrieren sich jedoch stark in dem Intervall [$\mu - 3 \cdot \sigma$, $\mu + 3 \cdot \sigma$].**randNorm(μ , σ , #Trials)** gibt eine Liste mit #Trials Dezimalzahlen aus der angegebenen Normalverteilung zurück.

RandSeed 1147

Done

randNorm(0,1)

0.492541

randNorm(3,4.5)

-3.54356

randPoly() (Zufallspolynom)**Katalog >** **randPoly(*Var*, *Ordnung*)** \Rightarrow AusdruckGibt ein Polynom in *Var* der angegebenen *Ordnung* zurück. Die Koeffizienten sind ganze Zufallszahlen im Bereich -9 bis 9. Der führende Koeffizient ist nicht null.*Ordnung* muss zwischen 0 und 99 betragen.

RandSeed 1147

Done

randPoly(x,5)

 $-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$ **randSamp() (Zufallsstichprobe)****Katalog >** **randSamp(*List*,#Trials[,noRepl])** \Rightarrow ListeGibt eine Liste mit einer Zufallsstichprobe von #Trials Versuchen aus Liste (*List*) zurück mit der Möglichkeiten, Stichproben zu ersetzen (*noRepl*=0) oder nicht zu ersetzen (*noRepl*=1). Die Vorgabe ist mit Stichprobenersatz.

Define list3={1,2,3,4,5}

Done

Define list4=randSamp(list3,6)

Done

list4

{1,3.,3.,1.,3.,1.}

RandSeed (Zufallszahl)

Katalog >

RandSeed Zahl

Zahl = 0 setzt die Ausgangsbasis ("seed") für den Zufallszahlengenerator auf die Werkseinstellung zurück. Bei **Zahl** ≠ 0 werden zwei Basen erzeugt, die in den Systemvariablen seed1 und seed2 gespeichert werden.

RandSeed 1147	Done
rand()	0.158206

real() (Reell)

Katalog >

real(Value1) ⇒ Wert

Gibt den Realteil des Arguments zurück.

real(2+3·i)	2
-------------	---

real(List1) ⇒ Liste

Gibt für jedes Element den Realteil zurück.

real({1+3·i,3,i})	{1,3,0}
-------------------	---------

real(Matrix1) ⇒ Matrix

Gibt für jedes Element den Realteil zurück.

real([1+3·i 3 2 i])	[1 3 2 0]
------------------------	--------------

► Rect

Katalog >

Vektor ►Rect

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @Rect eintippen.

$\begin{bmatrix} 3 & \angle \frac{\pi}{4} & \angle \frac{\pi}{6} \end{bmatrix}$	►Rect
[1.06066 1.06066 2.59808]	

Zeigt **Vektor** in der kartesischen Form [x, y, z] an. Der Vektor muss die Dimension 2 oder 3 besitzen und kann eine Zeile oder eine Spalte sein.

Hinweis: ►Rect ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen, und sie nimmt keine Aktualisierung von ans vor.

Hinweis: Siehe auch ►Polar Seite 123.

komplexer Wert ►Rect

Zeigt **komplexerWert** in der kartesischen Form a+bi an. **komplexerWert** kann jede komplexe Form haben. Eine reiθ-Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Im Bogenmaß-Modus:

$\begin{pmatrix} \frac{\pi}{3} \\ 4 \cdot e^3 \end{pmatrix}$	►Rect	11.3986
$\begin{pmatrix} 4 \angle \frac{\pi}{3} \end{pmatrix}$	►Rect	2.+3.4641·i

Hinweis: Für eine Eingabe in Polarform müssen Klammern ($r \angle \theta$) verwendet werden.

Im Neugrad-Modus:

$\{(1 \angle 100)\} \blacktriangleright \text{Rect}$

i

Im Grad-Modus:

$\{(4 \angle 60)\} \blacktriangleright \text{Rect}$

$2+3.4641 \cdot i$

Hinweis: Wählen Sie zur Eingabe von \angle das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ref() (Diagonalform)

ref(*MatrixI[, Tol]*) \Rightarrow *Matrix*

Gibt die Diagonalform von *MatrixI* zurück.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

$$\text{ref} \begin{pmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{pmatrix} = \begin{pmatrix} 1 & -2 & -4 & 4 \\ 0 & 5 & 5 & 5 \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{pmatrix}$$

- Wenn Sie **ctrl enter** verwenden oder den Modus **Autom. oder Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 * \max(\dim(\text{MatrixI})) * \text{rowNorm}(\text{MatrixI})$

Vermeiden Sie nicht definierte Elemente in *MatrixI*. Sie können zu unerwarteten Ergebnissen führen.

Wenn z. B. im folgenden Ausdruck *a* nicht definiert ist, erscheint eine Warnmeldung und das Ergebnis wird wie folgt angezeigt:

$$\text{ref} \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Die Warnung erscheint, weil das verallgemeinerte Element $1/a$ für $a=0$ nicht zulässig wäre.

Sie können dieses Problem umgehen, indem Sie zuvor einen Wert in a speichern oder wie im folgenden Beispiel gezeigt eine Substitution mit dem womit-Operator „|“ vornehmen.

$$\text{ref} \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} | a=0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Hinweis: Siehe auch rref() page 146.

RefreshProbeVars

RefreshProbeVars

Ermöglicht den Zugriff auf Sensordaten von allen verbundenen Sensorsonden in Ihrem TI-Basic-Programm.

StatusVar	Value	Status
<i>statusVar</i>	=0	Normal (Programmausführung fortsetzen) Die Applikation Vernier DataQuest™ befindet sich im Data Collection-Modus.
<i>statusVar</i>	=1	Hinweis: Die Applikation Vernier DataQuest™ muss sich im Messgerätmodus befinden, damit dieser Befehl funktioniert.
<i>statusVar</i>	=2	Die Applikation Vernier DataQuest™ wurde nicht gestartet.

Beispiel

```
Define temp ()=
Prgm
④ Prüfen, ob System bereit ist
RefreshProbeVars status
If status=0 Then
Disp "ready"
For n,1,50
RefreshProbeVars status
temperature:=meter.temperature
Disp "Temperature:
",temperature
If temperature>30 Then
Disp "Too hot"
```

StatusVar	Status	
Value		EndIf
Die Applikation Vernier		© 1 Sekunde zwischen den
<i>statusVar</i>	DataQuest™ wurde gestartet, ist	Messungen warten
=3	jedoch noch nicht mit Sonden	Wait 1
	verbunden.	EndFor
		Else
		Disp "Not ready. Try again
		later"
		EndIf
		EndPrgm

Hinweis: Dies kann auch mit TI-Innovator™ Hub verwendet werden.

remain() (Rest)

remain(Wert1, Wert2) \Rightarrow Wert
remain(Liste1, Liste2) \Rightarrow Liste
remain(Matrix1, Matrix2) \Rightarrow Matrix

Gibt den Rest des ersten Arguments bezüglich des zweiten Arguments gemäß folgender Definitionen zurück:

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

Als Folge daraus ist zu beachten, dass **remain(-x,y) = remain(x,y)**. Das Ergebnis ist entweder Null oder besitzt das gleiche Vorzeichen wie das erste Argument.

Hinweis: Siehe auch **mod()** Seite 105.

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

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

Request

Request promptString, var[, FlagAnz [, statusVar]]

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

Definieren Sie ein Programm:

```
Define request_demo()=Prgm
  Request "Radius: ",r
  Disp "Fläche = ",pi*r^2
EndPrgm
```

Programmierbefehl: Pausiert das Programm und zeigt ein Dialogfeld mit der Meldung *promptString* sowie einem Eingabefeld für die Antwort des Benutzers an.

Wenn der Benutzer eine Antwort eingibt und auf **OK** klickt, wird der Inhalt des Eingabefelds in die Variable *var* geschrieben.

Falls der Benutzer auf **Abbrechen** klickt, wird das Programm fortgesetzt, ohne Eingaben zu übernehmen. Das Programm verwendet den vorherigen *var*-Wert, soweit *var* bereits definiert wurde.

Bei dem optionalen Argument *FlagAnz* kann es sich um einen beliebigen Ausdruck handeln.

- Wenn *FlagAnz* fehlt oder den Wert **1** ergibt, werden die Eingabeaufforderung und die Benutzerantwort im Calculator-Protokoll angezeigt.
- Wenn *FlagAnz* den Wert **0** ergibt, werden die Aufforderung und die Antwort nicht im Protokoll angezeigt.

Das optionale Argument *statusVar* ermöglicht es dem Programm, zu bestimmen, wie der Benutzer das Dialogfeld verlassen hat. Beachten Sie, dass *statusVar* das Argument *FlagAnz* erfordert.

- Wenn der Benutzer auf **OK** geklickt oder die **Eingabetaste** bzw. **Strg+Eingabetaste** gedrückt hat, wird die Variable *statusVar* auf den Wert **1** gesetzt.
- Andernfalls wird die Variable *statusVar* auf den Wert **0** gesetzt.

Mit dem Argument *func()* kann ein Programm die Benutzerantwort als Funktionsdefinition speichern. Diese Syntax verhält sich so, als hätte der Benutzer den folgenden Befehl ausgeführt:

Starten Sie das Programm und geben Sie eine Antwort ein:

`request_demo()`



Ergebnis nach Auswahl von **OK**:

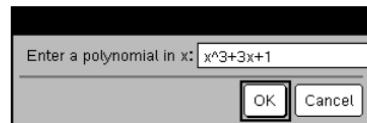
`Radius: 6/2
Fläche = 28.2743`

Definieren Sie ein Programm:

```
Define polynomial()=Prgm
  Request "Polynom in x
  eingeben:",p(x)
  Disp "Reelle Wurzeln:",polyRoots
  (p(x),x)
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

`polynomial()`



Ergebnis nach Eingabe von x^3+3x+1 und Auswahl von **OK**:

Define *Fkt(Arg1, ...Argn)* =
Benutzerantwort

Reelle Wurzeln: {-0,322185}

Anschließend kann das Programm die so definierte Funktion *Fkt()* nutzen. Die Meldung *EingabeString* sollte dem Benutzer die nötigen Informationen geben, damit dieser eine passende *Benutzerantwort* zur Vervollständigung der Funktionsdefinition eingeben kann.

Hinweis: Mit der Option Request Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen.

So halten Sie ein Programm an, das einen Befehl **Request** in einer Endlosschleife enthält:

- **Handheld:** Halten Sie die Taste  **on** gedrückt und drücken Sie mehrmals **enter**.
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Siehe auch **RequestStr**, page 140.

RequestStr

RequestStr *promptString, var[, FlagAnz]*

Programmierbefehl: Verhält sich genauso wie die erste Syntax des Befehls **Request**, die Benutzerantwort wird jedoch immer als String interpretiert. Der Befehl **Request** interpretiert die Antwort hingegen als Ausdruck, es sei denn, der Benutzer setzt sie in Anführungszeichen ("").

Hinweis: Sie können den Befehl **RequestStr** in benutzerdefinierten Programmen verwenden, jedoch nicht in Funktionen.

Definieren Sie ein Programm:

```
Define requestStr_demo()=Prgm
  RequestStr "Ihr Name:",name,0
  Disp "Die Antwort hat ",dim
  (name)," Zeichen."
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

```
requestStr_demo()
```

Zum Anhalten eines Programms mit dem Befehl **RequestStr** in einer Endlosschleife:

- **Handheld**: Halten Sie die Taste gedrückt und drücken Sie mehrmals **enter**.
- **Windows®**: Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®**: Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®**: Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Siehe auch **Request**, page 138.



Ergebnis nach Auswahl von **OK** (Hinweis:
Wegen *DispFlag = 0* werden
Eingabeaufforderung und Antwort nicht im
Protokoll angezeigt):

`requestStr_demo()`

Die Antwort hat 5 Zeichen.

Return

Return [Ausdr]

Gibt *Ausdr* als Ergebnis der Funktion zurück. Verwendbar in einem Block **Func...EndFunc**.

Hinweis: Verwenden Sie Zurück (**Return**) ohne Argument innerhalb eines Blocks **Prgm...EndPrgm**, um ein Programm zu beenden.

Hinweis zum Eingeben des Beispieles:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

```
Define factorial (nn)=
Func
Local answer,counter
1->answer
For counter,1,nn
answer· counter-> answer
EndFor
Return answer|
EndFunc
```

`factorial (3)`

6

right() (Rechts)

right(Liste1[, Anz]) ⇒ *Liste*

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

Gibt *Anz* Elemente zurück, die rechts in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

right(Quellstring[, Anz]) \Rightarrow string

right("Hello",2)

"lo"

Gibt *Anz* Zeichen zurück, die rechts in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

right(Vergleich) \Rightarrow Ausdruck

Gibt die rechte Seite einer Gleichung oder Ungleichung zurück.

rk23 ()**rk23(Ausdr, Var, abhVar, {Var0, VarMax}, abhVar0, VarSchritt [, diftol])** \Rightarrow Matrix**rk23(AusdrSystem, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt[, diftol])** \Rightarrow Matrix**rk23(AusdrListe, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt[, diftol])** \Rightarrow Matrix

Verwendet die Runge-Kutta-Methode zum Lösen des Systems

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

mit *abhVar*(*Var0*)=*abhVar0* auf dem Intervall [*Var0*,*VarMax*]. Gibt eine Matrix zurück, deren erste Zeile die Ausgabewerte von *Var* definiert, wie durch *VarSchritt* definiert. Die zweite Zeile definiert den Wert der ersten Lösungskomponente an den entsprechenden *Var* Werten usw.

Ausdr ist die rechte Seite, die die gewöhnliche Differentialgleichung (ODE) definiert.

AusdrSystem ist ein System rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

AusdrListe ist eine Liste rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

Differentialgleichung:

$$y' = 0.001 * y * (100 - y) \text{ und } y(0) = 10$$

$$\begin{aligned} \text{rk23}\left(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix} \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

Dieselbe Gleichung mit *diftol* auf $1.E-6$

$$\begin{aligned} \text{rk23}\left(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1, 1.E-6\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix} \end{aligned}$$

Gleichungssystem:

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

$$\text{mit } y1(0)=2 \text{ und } y2(0)=5$$

$$\begin{aligned} \text{rk23}\left(\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix} \end{aligned}$$

Var ist die unabhängige Variable.

ListeAbhVar ist eine Liste abhängiger Variablen.

{*Var0*, *VarMax*} ist eine Liste mit zwei Elementen, die die Funktion anweist, von *Var0* zu *VarMax* zu integrieren.

ListeAbhVar0 ist eine Liste von Anfangswerten für abhängige Variablen.

Wenn *VarSchritt* eine Zahl ungleich Null ergibt: Zeichen(*VarSchritt*) = Zeichen(*VarMax-Var0*) und Lösungen werden an *Var0+i*VarSchritt* für alle i=0,1,2,... zurückgegeben, sodass *Var0+i*VarSchritt* in [*var0*,*VarMax*] ist (möglicherweise gibt es keinen Lösungswert an *VarMax*).

Wenn *VarSchritt* Null ergibt, werden Lösungen an den „Runge-Kutta“ *Var*-Werten zurückgegeben.

diftol ist die Fehlertoleranz (standardmäßig 0.001).

root() (Wurzel)

root(Wert) ⇒ Wurzel

root(Wert1, Wert2) ⇒ Wurzel

root(*Wert*) gibt die Quadratwurzel von *Wert* zurück.

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

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

root(Wert1, Wert2) gibt die *Wert2* Wurzel von *Wert1* zurück. *Wert1* kann eine reelle oder komplexe Fließkommakonstante, eine ganze Zahl oder eine komplexe rationale Konstante sein.

Hinweis: Siehe auch **Vorlage n-te Wurzel**, Seite Seite 2.

rotate() (Rotieren)

rotate(Ganzzahl1[,#Rotationen])⇒ Ganzzahl

Im Bin-Modus>

rotate() (Rotieren)

Katalog >

Rotiert die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ► **Base2**, Seite 17.

Ist $\#Rotationen$ positiv, erfolgt eine Rotation nach links. Ist $\#Rotationen$ negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Bit nach rechts rotieren).

Beispielsweise in einer Rechtsrotation:

Jedes Bit rotiert nach rechts.

0b000000000000001111010110000110101

Bit ganz rechts rotiert nach ganz links.

ergibt sich:

0b1000000000000000111101011000011010

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

rotate(Liste1[, #Rotationen]) \Rightarrow Liste

Gibt eine um #Rotationen Elemente nach rechts oder links rotierte Kopie von *Liste1* zurück. Verändert *Liste1* nicht.

Ist $\#Rotationen$ positiv, erfolgt eine Rotation nach links. Ist $\#Rotationen$ negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Element nach rechts rotieren).

rotate(String l[,#Rotationen]) ⇒ String

Gibt eine um #Rotationen Zeichen nach rechts oder links rotierte Kopie von String1 zurück. Verändert String1 nicht.

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ◀ und ►, um den Cursor zu bewegen.

Im Hex-Modus:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h8000000000000001E3
rotate(0h78E,2)	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

Im Dec-Modus:

<code>rotate({1,2,3,4})</code>	{4,1,2,3}
<code>rotate({1,2,3,4},-2)</code>	{3,4,1,2}
<code>rotate({1,2,3,4},1)</code>	{2,3,4,1}

<code>rotate("abcd")</code>	"dabc"
<code>rotate("abcd",-2)</code>	"cdab"
<code>rotate("abcd",1)</code>	"bcda"

rotate() (Rotieren)

Katalog >

Ist $\#Rotationen$ positiv, erfolgt eine Rotation nach links. Ist $\#Rotationen$ negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Zeichen nach rechts rotieren)

round() (Runden)

Katalog >

round(Wert1[, Stellen]) \Rightarrow Wert

round(1.234567,3)

1.235

Gibt das Argument gerundet auf die angegebene Anzahl von Stellen nach dem Dezimaltrennzeichen zurück.

Stellen muss eine Ganzzahl zwischen 0 und 12 sein. Wenn *Stellen* nicht eingeschlossen wird, wird das Argument auf 12 Stellen gerundet zurückgegeben.

Hinweis: Die Anzeige des Ergebnisses kann von der Einstellung "Angezeigte Ziffern" beeinflusst werden.

round(Liste1[, Stellen]) \Rightarrow Liste

round({ $\pi, \sqrt{2}, \ln(2)$ },4)

{3.1416,1.4142,0.6931}

Gibt eine Liste von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

round(Matrix1[, Stellen]) \Rightarrow Matrix

round([$\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}$],1)

Gibt eine Matrix von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

rowAdd() (Zeilenaddition)

Katalog >

rowAdd(Matrix1, rIndex1, rIndex2) \Rightarrow Matrix

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

[$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$]

Gibt eine Kopie von *Matrix1* zurück, in der die Zeile *rIndex2* durch die Summe der Zeilen *rIndex1* und *rIndex2* ersetzt ist.

rowDim() (Zeilendimension)**Katalog >** **rowDim(Matrix) \Rightarrow Ausdruck**Gibt die Anzahl der Zeilen von *Matrix* zurück.

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

Hinweis: Siehe auch **colDim()** Seite 25.**rowNorm() (Zeilennorm)****Katalog >** **rowNorm(Matrix) \Rightarrow Ausdruck**Gibt das Maximum der Summen der Absolutwerte der Elemente der Zeilen von *Matrix* zurück.

$\text{rowNorm} \begin{pmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{pmatrix}$	25
--	----

Hinweis: Alle Matrixelemente müssen zu Zahlen vereinfachbar sein. Siehe auch **colNorm()** Seite 25.**rowSwap() (Zeilentausch)****Katalog >** **rowSwap(Matrix1, rIndex1, rIndex2) \Rightarrow Matrix**Gibt *Matrix1* zurück, in der die Zeilen *rIndex1* und *rIndex2* vertauscht sind.

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

rref() (Reduzierte Diagonalform)**Katalog >** **rref(Matrix1[, Tol]) \Rightarrow Matrix**Gibt die reduzierte Diagonalform von *Matrix1* zurück.

$\text{rref} \begin{pmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{pmatrix}$
--	---

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

- Wenn Sie **ctrl** **enter** verwenden oder den Modus **Autom.** oder **Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 * \max(\dim(Matrix\ I)) * \text{rowNorm}(Matrix\ I)$

Hinweis: Siehe auch **ref()** page 136.

S

sec() (Sekans)

 Taste

sec(WertI) \Rightarrow Wert

Im Grad-Modus:

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

sec(ListeI) \Rightarrow Liste

Gibt den Sekans von *WertI* oder eine Liste der Sekans aller Elemente in *ListeI* zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, g oder r benutzen, um den Winkelmodus vorübergend aufzuheben.

sec⁻¹() (Arkussekans)

 Taste

sec⁻¹(WertI) \Rightarrow Wert

Im Grad-Modus:

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

sec⁻¹(ListeI) \Rightarrow Liste

Gibt entweder den Winkel, dessen Sekans *WertI* entspricht, oder eine Liste der inversen Sekans aller Elemente in *ListeI* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

sec⁻¹() (Arkussekans)

trig Taste

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsec (...)** eintippen.

sec⁻¹({1,2,5}) {0,1.0472,1.36944}

sech() (Sekans hyperbolicus)

Katalog >

sech(Wert1) ⇒ Wert

sech(3) 0.099328

sech(Liste1) ⇒ Liste

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

Gibt den hyperbolischen Sekans von *Wert1* oder eine Liste der hyperbolischen Sekans der Elemente in *Liste1* zurück.

sech⁻¹() (Arkussekans hyperbolicus)

Katalog >

sech⁻¹(Wert1) ⇒ Wert

sech⁻¹(1) 0

sech⁻¹(Liste1) ⇒ Liste

sech⁻¹({1,-2,2.1}) {0,2.0944·i,8.e-15+1.07448·i}

Gibt den inversen hyperbolischen Sekans von *Wert1* oder eine Liste der inversen hyperbolischen Sekans aller Elemente in *Liste1* zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsech (...)** eintippen.

Send

Hub-Menü

Send exprOrString1[, exprOrString2] ...

Beispiel: Schalten Sie das blaue Element der integrierten RGB LED 0,5 Sekunden lang ein.

Send "SET COLOR.BLUE ON TIME .5"

Done

Programmierbefehl: Sendet einen oder mehrere TI-Innovator™ Hub Befehle an den verbundenen Hub.

exprOrString muss ein gültiger TI-Innovator™ Hub Befehl sein.

Send "SET COLOR.BLUE ON TIME .5"

Normalerweise enthält *exprOrString* einen Befehl "SET ..." zum Steuern eines Geräts oder einen Befehl "READ ..." zum Anfordern von Daten.

Done

Die Argumente werden hintereinander an den Hub gesendet.

Beispiel: Fordern Sie den aktuellen Wert des integrierten Lichtpegelsensors des Hub an. Ein Befehl **Get** ruft den Wert ab und weist ihn der Variablen *lightval* zu.

Send "READ BRIGHTNESS" Done

Get *lightval* Done

lightval 0.347922

Hinweis: Sie können den Befehl **Send** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

Hinweis: Siehe auch **Get** (Seite 65), **GetStr** (Seite 73) und **eval()** (Seite 52).

Beispiel: Senden Sie eine berechnete Frequenz an den integrierten Lautsprecher des Hub. Verwenden Sie die spezielle Variable *iostr.SendAns*, um den Hub-Befehl mit dem ausgewerteten Ausdruck anzuzeigen.

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

seq() (Folge)

Katalog >

seq(Ausdr, Var, Von, Bis[, Schritt]) \Rightarrow Liste

Erhöht Var in durch Schritt festgelegten Stufen von Von bis Bis, wertet Ausdr aus und gibt die Ergebnisse als Liste zurück. Der ursprüngliche Inhalt von Var ist nach Beendigung von **seq()** weiterhin vorhanden.

Der Vorgabewert für *Schritt* ist 1.

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

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl** **enter**.

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie **≈** aus.

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

seqGen()

Katalog >

seqGen(Ausdr, Var, abhVar, {Var0, VarMax}{, ListeAnfTerme [, VarSchritt [, ObergrWert]}]) \Rightarrow Liste

Generieren Sie die ersten 5 Terme der Folge $u(n) = u(n-1)^2/2$ mit $u(1)=2$ und $VarSchritt=1$.

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

Generiert eine Term-Liste für die Folge $abhVar(Var)=Ausdr$ wie folgt: Erhöht die unabhängige Variable Var von $Var0$ bis $VarMax$ um $VarSchritt$, wertet $abhVar(Var)$ für die entsprechenden Werte von Var mithilfe der Formel $Ausdr$ und der *ListeAnfTerme* aus und gibt die Ergebnisse als Liste zurück.

seqGen(*SystemListeOderAusdr*, *Var*,
ListeAbhVar, {*Var0*, *VarMax*} [,
MatrixAnfTerme [, *VarSchritt* [,
ObergrWert]]]) \Rightarrow *Matrix*

Generiert eine Term-Matrix für ein System (oder eine Liste) von Folgen *ListeAbhVar(Var)=SystemListeOderAusdr* wie folgt: Erhöht die unabhängige Variable *Var* von *Var0* bis *VarMax* um *VarSchritt*, wertet *ListeAbhVar(Var)* für die entsprechenden Werte von *Var* mithilfe der Formel *SystemListeOderAusdr* und der *MatrixAnfTerme* aus und gibt die Ergebnisse als Matrix zurück.

Der ursprüngliche Inhalt von *Var* ist nach Beendigung von **seqGen()** weiterhin vorhanden.

Der Standardwert für *VarSchritt* ist **1**.

seqn()

seqn(*Ausdr{u, n [, ListeAnfTerme[, nMax [, ObergrWert]]}]*) \Rightarrow *Liste*

Generiert eine Term-Liste für eine Folge $u(n)=Ausdr(u, n)$ wie folgt: Erhöht n von 1 bis $nMax$ um 1, wertet $u(n)$ für die entsprechenden Werte von n mithilfe der Formel $Ausdr(u, n)$ und *ListeAnfTerme* aus und gibt die Ergebnisse als Liste zurück.

seqn(*Ausdr{n [, nMax [, ObergrWert]}]*) \Rightarrow *Liste*

Beispiel mit *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 zweiter Folgen:

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u2(n-1)}{2} + uI(n-1)\right\}, n, \{u1, u2\}, \{1, 5\}, \begin{bmatrix} \square \\ 2 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Ein Hinweis: Die Lücke (\square) in der oben aufgeführten Anfangsterm-Matrix zeigt an, dass der Anfangsterm für $u1(n)$ mit der expliziten Folge-Formel $u1(n)=1/n$ berechnet wird.

Generieren Sie die ersten 6 Terme der Folge $u(n)=u(n-1)/2$ mit $u(1)=2$.

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

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

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

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

Generiert eine Term-Liste für eine nichtrekursive Folge $u(n)=\text{Ausdr}(n)$ wie folgt: Erhöht n von 1 bis $nMax$ um 1, wertet $u(n)$ für die entsprechenden Werte von n mithilfe der Formel $\text{Ausdr}(n)$ aus und gibt die Ergebnisse als Liste zurück.

Wenn $nMax$ fehlt, wird $nMax$ auf 2500 gesetzt

Wenn $nMax=0$, wird $nMax$ auf 2500 gesetzt

Hinweis: seqn() gibt seqGen() mit $n0=1$ und $nSchritt =1$ an

setMode

setMode(*ModusNameGanzzahl*,
GanzzahlFestlegen) \Rightarrow *Ganzzahl*

setMode(Liste) \Rightarrow *Liste mit ganzen Zahlen*

Nur gültig innerhalb einer Funktion oder eines Programms.

setMode(*ModusNameGanzzahl*,
GanzzahlFestlegen) schaltet den Modus *ModusNameGanzzahl* vorübergehend in *GanzzahlFestlegen* und gibt eine ganze Zahl entsprechend der ursprünglichen Einstellung dieses Modus zurück. Die Änderung ist auf die Dauer der Ausführung des Programms / der Funktion begrenzt.

ModusNameGanzzahl gibt an, welchen Modus Sie einstellen möchten. Hierbei muss es sich um eine der Modus-Ganzzahlen aus der nachstehenden Tabelle handeln.

GanzzahlFestlegen gibt die neue Einstellung für den Modus an. Für den Modus, den Sie festlegen, müssen Sie eine der in der nachstehenden Tabelle aufgeführten Einstellungs-Ganzzahlen verwenden.

Zeigen Sie den Näherungswert von π an, indem Sie die Standardeinstellung für Zahlen anzeigen (Display Digits) verwenden, und zeigen Sie dann π mit einer Einstellung von Fix 2 an. Kontrollieren Sie, dass der Standardwert nach Beendigung des Programms wieder hergestellt wird.

Define <i>prog1()</i> =Prgm	<i>Done</i>
Disp π	
setMode(1,16)	
Disp π	
EndPrgm	

<i>prog1()</i>	3.14159
	3.14
	<i>Done</i>

setMode(Liste) dient zum Ändern mehrerer Einstellungen. *Liste* enthält Paare von Modus- und Einstellungs-Ganzzahlen. **setMode(Liste)** gibt eine ähnliche Liste zurück, deren Ganzzahlen-Paare die ursprünglichen Modi und Einstellungen angeben.

Wenn Sie alle Moduseinstellungen mit **getMode(0) → var** gespeichert haben, können Sie **setMode(var)** verwenden, um diese Einstellungen wiederherzustellen, bis die Funktion oder das Programm beendet wird. Siehe **getMode()**, Seite 71.

Hinweis: Die aktuellen Moduseinstellungen werden an aufgerufene Subroutinen weitergegeben. Wenn eine der Subroutinen eine Moduseinstellung ändert, geht diese Modusänderung verloren, wenn die Steuerung zur aufrufenden Routine zurückkehrt.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Modus Name	Modus Ganzzahl	Einstellen von Ganzzahlen
Angezeigte Ziffern	1	1=Fließ, 2=Fließ 1, 3=Fließ 2, 4=Fließ 3, 5=Fließ 4, 6=Fließ 5, 7=Fließ 6, 8=Fließ 7, 9=Fließ 8, 10=Fließ 9, 11=Fließ 10, 12=Fließ 11, 13=Fließ 12, 14=Fix 0, 15=Fix 1, 16=Fix 2, 17=Fix 3, 18=Fix 4, 19=Fix 5, 20=Fix 6, 21=Fix 7, 22=Fix 8, 23=Fix 9, 24=Fix 10, 25=Fix 11, 26=Fix 12
Winkel	2	1=Bogenmaß, 2=Grad, 3=Neugrad
Exponentialformat	3	1=Normal, 2=Wissenschaftlich, 3=Technisch
Reell oder komplex	4	1=Reell, 2=Kartesisch, 3=Polar
Auto oder Approx.	5	1=Auto, 2=Approximiert
Vektorformat	6	1=Kartesisch, 2=Zylindrisch, 3=Sphärisch
Basis	7	1=Dezimal, 2=Hex, 3=Binär

shift(Ganzzahl1 [,#Verschiebungen])⇒Ganzzahl

Verschiebt die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ▶Base2, Seite 17.

Ist #*Verschiebungen* positiv, erfolgt die Verschiebung nach links. Ist #*Verschiebungen* negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1 (ein Bit nach rechts verschieben).

In einer Rechtsverschiebung wird das ganz rechts stehende Bit abgeschnitten und als ganz links stehendes Bit eine 0 oder 1 eingesetzt. Bei einer Linksverschiebung wird das Bit ganz links abgeschnitten und 0 als letztes Bit rechts eingesetzt.

Beispielsweise in einer Rechtsverschiebung:

Alle Bits werden nach rechts verschoben.

0b000000000000000111101011000011010

Setzt 0 ein, wenn Bit ganz links 0 ist, und 1, wenn Bit ganz links 1 ist.

Es ergibt sich:

0b000000000000000111101011000011010

Das Ergebnis wird gemäß dem jeweiligen Basis-Modus angezeigt. Führende Nullen werden nicht angezeigt.

shift(Liste1 [,#Verschiebungen])⇒Liste

Gibt eine um #*Verschiebungen* Elemente nach rechts oder links verschobene Kopie von *Liste1* zurück. Verändert *Liste1* nicht.

Im Bin-Modus:

shift(0b1111010110000110101)	0b111101011000011010
shift(256,1)	0b1000000000

Im Hex-Modus:

shift(0h78E)	0h3C7
shift(0h78E,-2)	0h1E3
shift(0h78E,2)	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

Im Dec-Modus:

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() (Verschieben)

Katalog > 

Ist #Verschiebungen positiv, erfolgt die Verschiebung nach links. ist #Verschiebungen negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1 (ein Element nach rechts verschieben).

Dadurch eingeführte neue Elemente am Anfang bzw. am Ende von *Liste* werden auf "undef" gesetzt.

shift(String1 [,#Verschiebungen])⇒String

Gibt eine um #Verschiebungen Zeichen nach rechts oder links verschobene Kopie von *Liste1* zurück. Verändert *String1* nicht.

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

Ist #Verschiebungen positiv, erfolgt die Verschiebung nach links. ist #Verschiebungen negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1 (ein Zeichen nach rechts verschieben).

Dadurch eingeführte neue Zeichen am Anfang bzw. am Ende von *String* werden auf ein Leerzeichen gesetzt.

sign() (Zeichen)

Katalog > 

sign(Wert1)⇒Wert

sign(-3.2)	-1
------------	----

sign(Liste1)⇒Liste

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

sign(Matrix1)⇒Matrix

Gibt für reelle und komplexe Wert1 Wert1 / abs(Wert1) zurück, wenn Wert1 ≠ 0.

Bei Komplex-Formatmodus Reell:	
sign([-3 0 3])	[-1 undef 1]

Gibt 1 zurück, wenn Wert1 positiv ist.

Gibt -1 zurück, wenn Wert1 negativ ist.

sign(0) gibt ±1 zurück, wenn als Komplex-Formatmodus Reell eingestellt ist; anderenfalls gibt es sich selbst zurück.

sign(0) stellt im komplexen Bereich den Einheitskreis dar.

Gibt für jedes Element einer Liste bzw. Matrix das Vorzeichen zurück.

simult() (Gleichungssystem)

Katalog > 

simult(KoeffMatrix, KonstVektor[, Tol]) \Rightarrow Matrix

Ergibt einen Spaltenvektor, der die Lösungen für ein lineares Gleichungssystem enthält.

Hinweis: Siehe auch **linSolve()**, Seite 91.

KoeffMatrix muss eine quadratische Matrix sein, die die Koeffizienten der Gleichung enthält.

KonstVektor muss die gleiche Zeilenanzahl (gleiche Dimension) besitzen wie *KoeffMatrix* und die Konstanten enthalten.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

- Wenn Sie den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E^{-14} \cdot \max(\dim(KoeffMatrix)) \cdot \text{rowNorm}(KoeffMatrix)$

simult(KoeffMatrix, KonstMatrix[, Tol]) \Rightarrow Matrix

Löst mehrere lineare Gleichungssysteme, die alle dieselben Gleichungskoeffizienten, aber unterschiedliche Konstanten haben.

Jede Spalte in *KonstMatrix* muss die Konstanten für ein Gleichungssystem enthalten. Jede Spalte in der sich ergebenden Matrix enthält die Lösung für das entsprechende System.

Auflösen nach x und y:

$$x + 2y = 1$$

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

$$\begin{array}{l} \text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \\ \hline \begin{bmatrix} -3 \\ 2 \end{bmatrix} \end{array}$$

Die Lösung ist $x = -3$ und $y = 2$.

Auflösen:

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{array}{l} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow matx1 \\ \hline \text{simult}\left(matx1, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \\ \hline \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \end{array}$$

Auflösen:

$$x + 2y = 1$$

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

$$x + 2y = 2$$

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

simult() (Gleichungssystem)

Katalog >

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

Für das erste System ist $x=-3$ und $y=2$. Für das zweite System ist $x=7$ und $y=9/2$.

sin() (Sinus)

Taste

$\sin(Wert\,l) \Rightarrow Wert$

$\sin(Liste\,l) \Rightarrow Liste$

$\sin(Wert\,l)$ gibt den Sinus des Arguments zurück.

$\sin(Liste\,l)$ gibt eine Liste zurück, die für jedes Element von $Liste\,l$ den Sinus enthält.

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können $^{\circ}$, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

$\sin(Quadratmatrix\,l) \Rightarrow Quadratmatrix$

Gibt den Matrix-Sinus von $Quadratmatrix\,l$ zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Sinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$Quadratmatrix\,l$ muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Grad-Modus:

$\sin\left(\frac{\pi}{4}\right)$	0.707107
$\sin(45)$	0.707107
$\sin(\{0,60,90\})$	{0.,0.866025,1.}

Im Neugrad-Modus:

$\sin(50)$	0.707107
------------	----------

Im Bogenmaß-Modus:

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

Im Bogenmaß-Modus:

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

sin⁻¹() (Arkussinus)

Taste

$\sin^{-1}(Wert\,l) \Rightarrow Wert$

Im Grad-Modus:

sin⁻¹() (Arkussinus)

trig Taste

sin⁻¹(Liste1)⇒Liste

sin⁻¹(1)

90.

sin⁻¹(Wert1) gibt den Winkel, dessen Sinus Wert1 ist, zurück.

sin⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus Liste1 den inversen Sinus zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsin (...) eintippen.**

sin⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Sinus von Quadratmatrix1 zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Sinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Neugrad-Modus:

sin⁻¹(1)

100.

Im Bogenmaß-Modus:

sin⁻¹({0,0,2,0,5}) {0.,0.201358,0.523599}

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

sin⁻¹ $\begin{pmatrix} 1 & 5 \\ 4 & 2 \end{pmatrix}$
[-0.174533-0.12198·i 1.74533-2.35591·i]
[1.39626-1.88473·i 0.174533-0.593162·i]

sinh() (Sinus hyperbolicus)

Katalog >

sinh(Wert1)⇒Wert

sinh(1,2) 1.50946

sinh(Liste1)⇒Liste

sinh({0,1,2,3.}) {0,1.50946,10.0179}

sinh (Wert1) gibt den Sinus hyperbolicus des Arguments zurück.

sinh (Liste1) gibt in Form einer Liste für jedes Element aus Liste1 den Sinus hyperbolicus zurück.

sinh(Quadratmatrix1)⇒Quadratmatrix

Im Bogenmaß-Modus:

sinh() (Sinus hyperbolicus)

Katalog >

Gibt den Matrix-Sinus hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Sinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$$\sinh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 360.954 & 305.708 & 239.604 \\ 352.912 & 233.495 & 193.564 \\ 298.632 & 154.599 & 140.251 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

sinh⁻¹⁽⁾ (Arkussinus hyperbolicus)

Katalog >

sinh^{-1(Wert1)}⇒Wert

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

sinh^{-1(Liste1)}⇒Liste

$$\sinh^{-1}(\{0,2,1,3\}) = \{0,1.48748,1.81845\}$$

sinh^{-1(Wert1)} gibt den inversen Sinus hyperbolicus des Arguments zurück.

sinh^{-1(Liste1)} gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Sinus hyperbolicus zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsinh(...)** eintippen.

sinh^{-1(Quadratmatrix1)}⇒Quadratmatrix

Im Bogenmaß-Modus:

Gibt den inversen Matrix-Sinus hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Sinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$$\sinh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

SinReg

Katalog >

SinReg X, Y [, [Iterationen],[Periode] [, Kategorie, Mit]]

Berechnet die sinusförmige Regression auf Listen X und Y . Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Iterationen ist ein Wert, der angibt, wie viele Lösungsversuche (1 bis 16) maximal unternommen werden. Bei Auslassung wird 8 verwendet. Größere Werte führen in der Regel zu höherer Genauigkeit, aber auch zu längeren Ausführungszeiten, und umgekehrt.

Periode gibt eine geschätzte Periode an. Bei Auslassung sollten die Werte in X sequentiell angeordnet und die Differenzen zwischen ihnen gleich sein. Wenn Sie *Periode* jedoch angeben, können die Differenzen zwischen den einzelnen x-Werten ungleich sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Die Ausgabe von **SinReg** erfolgt unabhängig von der Winkelmoduseinstellung immer im Bogenmaß (rad).

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten
stat.Resid	Residuen von der Regression

Ausgabeveriable	Beschreibung
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

SortA (In aufsteigender Reihenfolge sortieren)

Katalog > 

SortA *Liste1*[, *Liste2*] [, *Liste3*] ...

SortA *Vektor1*[, *Vektor2*] [, *Vektor3*] ...

Sortiert die Elemente des ersten Arguments in aufsteigender Reihenfolge.

Bei Angabe von mehr als einem Argument werden die Elemente der zusätzlichen Argumente so sortiert, dass ihre neue Position mit der neuen Position der Elemente des ersten Arguments übereinstimmt.

Alle Argumente müssen Listen- oder Vektornamen sein. Alle Argumente müssen die gleiche Dimension besitzen.

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA <i>list1</i>	<i>Done</i>
<i>list1</i>	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA <i>list2</i> , <i>list1</i>	<i>Done</i>
<i>list2</i>	$\{1,2,3,4\}$
<i>list1</i>	$\{4,3,2,1\}$

SortD (In absteigender Reihenfolge sortieren)

Katalog > 

SortD *Liste1*[, *Liste2*] [, *Liste3*] ...

SortD *Vektor1*[, *Vektor2*] [, *Vektor3*] ...

Identisch mit **SortA** mit dem Unterschied, dass **SortD** die Elemente in absteigender Reihenfolge sortiert.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD <i>list1</i> , <i>list2</i>	<i>Done</i>
<i>list1</i>	$\{4,3,2,1\}$
<i>list2</i>	$\{3,4,1,2\}$

SortD (In absteigender Reihenfolge sortieren)

Katalog >

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben.
Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

►Sphere (Kugelkoordinaten)

Katalog >

Vektor ►Sphere

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Sphere** eintippen.

Zeigt den Zeilen- oder Spaltenvektor in Kugelkoordinaten [$\rho \angle\theta \angle\phi$] an.

Vektor muss die Dimension 3 besitzen und kann ein Zeilen- oder ein Spaltenvektor sein.

Hinweis: ►Sphere ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen.

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl enter**.

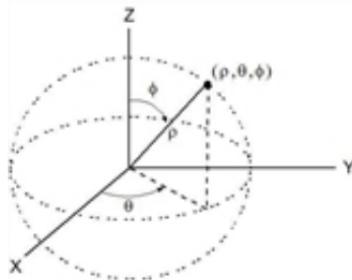
Windows®: Drücken Sie Strg+Eingabetaste.

Macintosh®: Drücken ⌘+Eingabetaste.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie aus.

```
[1 2 3]►Sphere  
[3.74166 ∠1.10715 ∠0.640522]
```

```
[2 ∠π/4 3]►Sphere  
[3.60555 ∠0.785398 ∠0.588003]
```



sqrt() (Quadratwurzel)

Katalog >

sqrt(Wert1)⇒Wert

$\sqrt{4}$ 2

sqrt(Liste1)⇒Liste

$\sqrt{\{9,2,4\}}$ {3,1.41421,2}

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 1.

stat.results

stat.results

Zeigt Ergebnisse einer statistischen Berechnung an.

Die Ergebnisse werden als Satz von Namen-Wert-Paaren angezeigt. Die angezeigten Namen hängen von der zuletzt ausgewerteten Statistikfunktion oder dem letzten Befehl ab.

Sie können einen Namen oder einen Wert kopieren und ihn an anderen Positionen einfügen.

Hinweis: Definieren Sie nach Möglichkeit keine Variablen, die dieselben Namen haben wie die für die statistische Analyse verwendeten Variablen. In einigen Fällen könnte ein Fehler auftreten. Namen von Variablen, die für die statistische Analyse verwendet werden, sind in der Tabelle unten aufgelistet.

xlist:=	{1,2,3,4,5}	{1,2,3,4,5}
ylist:=	{4,8,11,14,17}	{4,8,11,14,17}
LinRegMx xlist,ylist,1: stat.results		
"Title"	"Linear Regression (mx+b)"	
"RegEqn"	"m*x+b"	
"m"	3.2	
"b"	1.2	
"r ² "	0.996109	
"r"	0.998053	
"Resid"	"{...}"	
stat.values		
	"Linear Regression (mx+b)"	
	"m*x+b"	
	3.2	
	1.2	
	0.996109	
	0.998053	
	"{-0.4,0.4,0.2,0.,-0.2}"	

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dflnteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSIInteract
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.ȐX
stat.b9	stat.FBlock	Stat.ŷ	stat.Σx ²	stat.ȐX1

stat.b10	stat.Fcol	stat. \hat{p}_1	stat. Σ_{xy}	stat. \bar{X}_2
stat.bList	stat.FInteract	stat. \hat{p}_2	stat. Σ_y	stat. \bar{X}_{Diff}
stat. χ^2	stat.FreqReg	stat. \hat{p}_{Diff}	stat. Σ_{y^2}	stat. \bar{X}_{List}
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. \bar{y}
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. \hat{y}
stat.CookDist	stat.MaxX	stat.PValRow	stat.ESlope	stat. \hat{y}_{List}
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

Hinweis: Immer, wenn die Applikation 'Lists & Spreadsheet' statistische Ergebnisse berechnet, kopiert sie die Gruppenvariablen "stat." in eine "stat#"-Gruppe, wobei # eine automatisch inkrementierte Zahl ist. Damit können Sie vorherige Ergebnisse beibehalten, während mehrere Berechnungen ausgeführt werden.

stat.values

Katalog >

stat.values

Siehe stat.results.

Zeigt eine Matrix der Werte an, die für die zuletzt ausgewertete Statistikfunktion oder den letzten Befehl berechnet wurden.

Im Gegensatz zu stat.results lässt stat.values die den Werten zugeordneten Namen aus.

Sie können einen Wert kopieren und ihn an anderen Positionen einfügen.

stDevPop() (Populations-Standardabweichung)

Katalog >

stDevPop(Liste[, Häufigkeitsliste]) \Rightarrow Ausdruck

Im Bogenmaß- und automatischen Modus:

stDevPop({1,2,5,-6,3,-2}) 3.59398

stDevPop({1.3,2.5,-6.4},{3,2,5}) 4.11107

Ergibt die Populations-Standardabweichung der Elemente in Liste.

Jedes Häufigkeitsliste-Element gewichtet die Elemente von Liste in der gegebenen Reihenfolge entsprechend.

stDevPop() (Populations-Standardabweichung)

Katalog > 

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

stDevPop(*Matrix I[, Häufigkeitsmatrix]*) \Rightarrow *Matrix*

Ergibt einen Zeilenvektor der Populations-Standardabweichungen der Spalten in *Matrix I*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix I* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix I* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

$$\text{stDevPop} \begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix} \quad [3.26599 \quad 2.94392 \quad 1.63299]$$

$$\text{stDevPop} \begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix} \quad [2.52608 \quad 5.21506]$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog > 

stDevSamp(*Liste[, Häufigkeitsliste]*) \Rightarrow *Ausdruck*

Ergibt die Stichproben-Standardabweichung der Elemente in *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

$$\text{stDevSamp}\{\{1,2,5,-6,3,-2\}\} \quad 3.937$$

$$\text{stDevSamp}\{\{1,3,2,5,-6,4\}, \{3,2,5\}\} \quad 4.33345$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog >

stDevSamp(*Matrix1[, Häufigkeitsmatrix]*) \Rightarrow Matrix

Ergebnis einen Zeilenvektor der Stichproben-Standardabweichungen der Spalten in *Matrix1*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix1* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

stDevSamp $\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix}$
[4. 3.60555 2.]

stDevSamp $\begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix}$
[2.7005 5.44695]

Stop (Stopp)

Katalog >

Stop

Programmierbefehl: Beendet das Programm.

Stop ist in Funktionen nicht zulässig.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

i:=0	0
Define prog1() \Rightarrow Prgm	Done
For i,1,10,1	
If i=5	
Stop	
EndFor	
EndPrgm	
prog1()	Done
i	5

Store (Speichern)

Siehe → (speichern), Seite 215.

string() (String)

Katalog >

string(*Ausdr*) \Rightarrow String

Vereinfacht *Ausdr* und gibt das Ergebnis als Zeichenkette zurück.

string(1.2345)	"1.2345"
string(1+2)	"3"

subMat() (Untermatrix)**Katalog >**

subMat(*Matrix1*[, *vonZei*] [, *vonSpl*] [, *bisZei*] [, *bisSpl*]) \Rightarrow *Matrix*

Gibt die angegebene Untermatrix von *Matrix1* zurück.

Vorgaben: *vonZei*=1, *vonSpl*=1, *bisZei*=letzte Zeile, *bisSpl*=letzte Spalte.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
subMat(<i>m1</i> ,2,1,3,2)	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
subMat(<i>m1</i> ,2,2)	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Summe (Sigma)**Siehe $\Sigma()$, Seite 207.****sum() (Summe)****Katalog >**

sum(*Liste*[, *Start*[, *Ende*]]) \Rightarrow *Ausdruck*

Gibt die Summe der Elemente in *Liste* zurück.

Start und *Ende* sind optional. Sie geben einen Elementebereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *Liste* werden ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

sum(*Matrix1*[, *Start*[, *Ende*]]) \Rightarrow *Matrix*

Gibt einen Zeilenvektor zurück, der die Summen der Elemente aus den Spalten von *Matrix1* enthält.

Start und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *Matrix1* werden ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

sum({1,2,3,4,5})	15
sum({a,2·a,3·a})	
	"Error: Variable is not defined"
sum(seq(n,n,1,10))	55
sum({1,3,5,7,9},3)	21

sum({1 2 3 4 5 6})	[5 7 9]
sum({1 2 3 4 5 6 7 8 9})	[12 15 18]
sum({1 2 3 4 5 6 7 8 9},2,3)	[11 13 15]

sumIf()

Katalog > 

sumIf(*Liste*,*Kriterien*[,
SummeListe]) \Rightarrow *Wert*

Gibt die kumulierte Summe aller Elemente in *Liste* zurück, die die angegebenen *Kriterien* erfüllen. Optional können Sie eine Alternativliste, *SummeListe*, angeben, an die die Elemente zum Kumulieren weitergegeben werden sollen.

Liste kann ein Ausdruck, eine Liste oder eine Matrix sein. *SummeListe* muss, sofern sie verwendet wird, dieselben Dimension (en) haben wie *Liste*.

Kriterien können sein:

- Ein Wert, ein Ausdruck oder eine Zeichenfolge. So kumuliert beispielsweise **34** nur solche Elemente in *Liste*, die vereinfacht den Wert 34 ergeben.
- Ein Boolescher Ausdruck, der das Sonderzeichen **?** als Platzhalter für jedes Element verwendet. Beispielsweise zählt **?<10** nur solche Elemente in *Liste* zusammen, die kleiner als 10 sind.

Wenn ein Element in *Liste* die *Kriterien* erfüllt, wird das Element zur Kumulationssumme hinzugerechnet. Wenn Sie *SummeListe* hinzufügen, wird stattdessen das entsprechende Element aus *SummeListe* zur Summe hinzugerechnet.

In der Lists & Spreadsheet Applikation können Sie anstelle von *Liste* und *SummeListe* auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Hinweis: Siehe auch **countIf()**, Seite 33.

sumIf({1,2,e,3,π,4,5,6},2.5<?<4.5)

12.859874482

sumIf({1,2,3,4},2<?<5,{10,20,30,40})

70

sumSeq()

Siehe **Σ()**, Seite 207.

system(*Wert* [, *Wert2* [, *Wert3* [, ...]]])

Gibt ein Gleichungssystem zurück, das als Liste formatiert ist. Sie können ein Gleichungssystem auch mit Hilfe einer Vorlage erstellen.

T**T (Transponierte)**

Matrix1 **T** \Rightarrow *matrix*

Gibt die komplex konjugierte, transponierte Matrix von *Matrix1* zurück.

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

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

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @t eintippen.

tan() (Tangens)

tan(*Wert1*) \Rightarrow *Wert*

Im Grad-Modus:

$$\tan\left(\left(\frac{\pi}{4}\right)_r\right)$$

1.

$$\tan(45)$$

1.

$$\tan(\{0,60,90\})$$

{0.,1.73205,undef}

tan(*Liste1*) \Rightarrow *Liste*

tan(*Wert1*) gibt den Tangens des Arguments zurück.

tan(*Liste1*) gibt in Form einer Liste für jedes Element in *Liste1* den Tangens zurück.

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Im Neugrad-Modus:

$$\tan\left(\left(\frac{\pi}{4}\right)_r\right)$$

1.

$$\tan(50)$$

1.

$$\tan(\{0,50,100\})$$

{0.,1.,undef}

Im Bogenmaß-Modus:

tan() (Tangens)

trig Taste

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45^\circ)$	1.
$\tan\left\{\pi, \frac{\pi}{3}, \pi, \frac{\pi}{4}\right\}$	{0., 1.73205, 0., 1.}

tan(Quadratmatrix 1)⇒Quadratmatrix

Gibt den Matrix-Tangens von *Quadratmatrix 1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Tangens jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix 1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$\tan\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$
--	--

tan⁻¹() (Arkustangens)

trig Taste

tan⁻¹(Wert 1)⇒Wert

Im Grad-Modus:

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

tan⁻¹(Liste 1)⇒Liste

Im Neugrad-Modus:

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

tan⁻¹(Wert 1) gibt den Winkel zurück, dessen Tangens *Wert 1* ist.

tan⁻¹(Liste 1) gibt in Form einer Liste für jedes Element aus *Liste 1* den inversen Tangens zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arctan (...) eintippen**.

tan⁻¹(Quadratmatrix 1)⇒Quadratmatrix

Im Bogenmaß-Modus:

$\tan^{-1}\{0,0,2,0,5\}$	{0,0.197396,0.463648}
--------------------------	-----------------------

Gibt den inversen Matrix-Tangens von *Quadratmatrix 1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Tangens jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Im Bogenmaß-Modus:

tan⁻¹() (Arkustangens)

trig Taste

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{pmatrix}$$

tanh() (Tangens hyperbolicus)

Katalog >

tanh(Wert I)⇒Wert

$$\tanh(1.2) = 0.833655$$

tanh(Liste I)⇒Liste

$$\tanh(\{0,1\}) = \{0,0.761594\}$$

tanh(Wert I) gibt den Tangens hyperbolicus des Arguments zurück.

tanh(Liste I) gibt in Form einer Liste für jedes Element aus *Liste I* den Tangens hyperbolicus zurück.

tanh(Quadratmatrix I)⇒Quadratmatrix

Gibt den Matrix-Tangens hyperbolicus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Tangens hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$$\tanh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{pmatrix}$$

tanh⁻¹() (Arkustangens hyperbolicus)

Katalog >

tanh⁻¹(Wert I)⇒Wert

Im Komplex-Formatmodus "kartesisch":

$$\tanh^{-1}(0) = 0.$$

$$\tanh^{-1}(\{1,2,1,3\}) = \{\text{undef}, 0.518046 - 1.5708 \cdot i, 0.346574 - 1.5708 \cdot i\}$$

tanh⁻¹(Liste I)⇒Liste

tanh⁻¹(Wert I) gibt den inversen Tangens hyperbolicus des Arguments zurück.

tanh⁻¹(Liste I) gibt in Form einer Liste für jedes Element aus *Liste I* den inversen Tangens hyperbolicus zurück.

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arctanh (...)** eintippen.

tanh⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Tangens hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Tangens hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\tanh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} -0.099353+0.164058 \cdot i & 0.267834-1.4908 \\ -0.087596-0.725533 \cdot i & 0.479679-0.9473 \\ 0.511463-2.08316 \cdot i & -0.878563+1.7901 \end{bmatrix}$$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

tCdf()

tCdf

(UntGrenze,ObGrenze,FreiGrad)⇒Zahl, wenn *UntGrenze* und *ObGrenze* Zahlen sind, *Liste*, wenn *UntGrenze* und *ObGrenze* Listen sind

Berechnet für eine Student-*t*-Verteilung mit vorgegebenen Freiheitsgraden *FreiGrad* die Intervallwahrscheinlichkeit zwischen *UntGrenze* und *ObGrenze*.

Für $P(X \leq obereGrenze)$ setzen Sie *untereGrenze* = **-9E999**.

Text

Text EingabeString[, FlagAnz]

Programmierbefehl: Pausiert das Programm und zeigt die Zeichenkette *EingabeString* in einem Dialogfeld an.

Wenn der Benutzer **OK** auswählt, wird die Programmausführung fortgesetzt.

Definieren Sie ein Programm, das fünfmal anhält und jeweils eine Zufallszahl in einem Dialogfeld anzeigt.

Schließen Sie in der Vorlage **Prgm...EndPrgm** jede Zeile mit **[□]** ab anstatt mit **[enter]**. Auf der Computertastatur halten Sie **Alt** gedrückt und drücken die **Eingabetaste**.

Bei dem optionalen Argument *FlagAnz* kann es sich um einen beliebigen Ausdruck handeln.

- Wenn *FlagAnz* fehlt oder den Wert **1** ergibt, wird die Textmeldung im Calculator-Protokoll angezeigt.
- Wenn *FlagAnz* den Wert **0** ergibt, wird die Meldung nicht im Protokoll angezeigt.

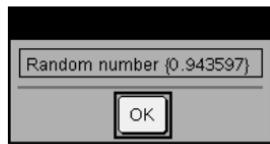
Wenn das Programm eine Eingabe vom Benutzer benötigt, verwenden Sie stattdessen **Request**, Seite 138, oder **RequestStr**, Seite 140.

Hinweis: Sie können diesen Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number " &
    string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

Starten Sie das Programm:
text_demo()

Muster eines Dialogfelds:



Then

Siehe If, Seite 75.

tInterval

Katalog >

tInterval *Liste[,Häuf[,KNiv]]*

(Datenlisteneingabe)

tInterval $\bar{x}, s_x, n[, KNiv]$

(Zusammenfassende statistische Eingabe)

Berechnet das Konfidenzintervall *t*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
stat. \bar{x}	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat. σ_x	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert

tInterval_2Samp (Zwei-Stichproben-t-Konfidenzintervall)

Katalog > 

tInterval_2Samp Liste1, Liste2[, Häufigkeit1
[, Häufigkeit2[, KStufe[, Verteilt]]]]

(Datenlisteneingabe)

tInterval_2Samp $\bar{x}_1, sx_1, n_1, \bar{x}_2, sx_2, n_2$
[, KStufe[, Verteilt]]]

(Zusammenfassende statistische Eingabe)

Berechnet ein t -Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Verteilt=1 verteilt Varianzen; *Verteilt=0* verteilt keine Varianzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. $\bar{x}_1 - \bar{x}_2$	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade

AusgabevARIABLE	Beschreibung
stat. \bar{x}_1 , stat. \bar{x}_2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat. σ_x_1 , stat. σ_x_2	Stichproben-Standardabweichungen für <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn <i>Verteilt</i> = JA.

tPdF()

Katalog >

tPdF(*XWert*,*FreiGrad*)⇒Zahl, wenn *XWert* eine Zahl ist, Liste, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion (PdF) einer Student-*t*-Verteilung an einem bestimmten *x*-Wert für die vorgegebenen Freiheitsgrade *FreiGrad*.

trace()

Katalog >

trace(*Quadratmatrix*)⇒Wert

Gibt die Spur (Summe aller Elemente der Hauptdiagonalen) von *Quadratmatrix* zurück.

trace	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	15
<i>a:=12</i>		12
trace	$\begin{pmatrix} a & 0 \\ 1 & a \end{pmatrix}$	24

```
Try
block1
Else
block2
EndTry
```

Führt *Block1* aus, bis ein Fehler auftritt.
 Wenn in *Block1* ein Fehler auftritt, wird die
 Programmausführung an *Block2*
 übertragen. Die Systemvariable *Fehlercode*
(errCode) enthält den Fehlercode, der es
 dem Programm ermöglicht, eine
 Fehlerwiederherstellung durchzuführen.
 Eine Liste der Fehlercodes finden Sie unter
 "Fehlercodes und -meldungen" (Seite 245).

Block1 und *Block2* können einzelne
 Anweisungen oder Reihen von
 Anweisungen sein, die durch das Zeichen
 ":" voneinander getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von
 mehrzeiligen Programm- und
 Funktionsdefinitionen finden Sie im
 Abschnitt „Calculator“ des
 Produkthandbuchs.

Beispiel 2

Um die Befehle **Versuche (Try)**, **LöFehler (ClrErr)** und **Ügebefehl (PassErr)** im Betrieb zu sehen, geben Sie das rechts gezeigte Programm *eigenvals()* ein. Sie starten das Programm, indem Sie jeden der folgenden Ausdrücke eingeben.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

Hinweis: Siehe auch **LöFehler**, Seite 24, und **Ügebefehl**, Seite 122.

```
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

Definiere *eigenvals(a,b)=Prgm*

© Programm *eigenvals(A,B)* zeigt die
 Eigenwerte von A·B an

Try

Disp "A= ",a

Disp "B= ",b

Disp " "

Disp "Eigenwerte von A·B sind:",eigVl(a*b)

Else

If errCode=230 Then

Disp "Fehler: Produkt von A·B muss eine
 quadratische Matrix sein"

ClrErr

```

Else
PassErr
EndIf
EndTry
EndPrgm

```

tTest**tTest** $\mu0, Liste[, Häufigkeit[, Hypoth]]$

(Datenlisteneingabe)

tTest $\mu0, \bar{x}, sx, n, [Hypoth]$

(Zusammenfassende statistische Eingabe)

Führt einen Hypothesen-Test für einen einzelnen, unbekannten Populationsmittelwert μ durch, wenn die Populations-Standardabweichung σ unbekannt ist. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 162.)

Getestet wird $H_0: \mu = \mu0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu < \mu0$ setzen Sie *Hypoth<0*

Für $H_a: \mu \neq \mu0$ (Standard) setzen Sie *Hypoth=0*

Für $H_a: \mu > \mu0$ setzen Sie *Hypoth>0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeverable	Beschreibung
stat.t	$(\bar{x} - \mu0) / (stdev / \sqrt{n})$
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade

Ausgabeveriable	Beschreibung
stat. \bar{x}	Stichprobenmittelwert der Datenfolge in <i>Liste</i>
stat.sx	Stichproben-Standardabweichung der Datenfolge
stat.n	Stichprobenumfang

tTest_2Samp (t-Test für zwei Stichproben)

Katalog > 

tTest_2Samp *Liste1*,*Liste2*[,*Häufigkeit1*
[,*Häufigkeit2*[,*Hypoth*[,*Verteilt*]]]]

(Datenlisteneingabe)

tTest_2Samp $\bar{x}_1, sx_1, n_1, \bar{x}_2, sx_2, n_2$ [,*Hypoth*
[,*Verteilt*]]]

(Zusammenfassende statistische Eingabe)

Berechnet einen *t*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Getestet wird $H_0: \mu_1 = \mu_2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu_1 < \mu_2$ setzen Sie *Hypoth*<0

Für $H_a: \mu_1 \neq \mu_2$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \mu_1 > \mu_2$ setzen Sie *Hypoth*>0

Verteilt=1 verteilt Varianzen

Verteilt=0 verteilt keine Varianzen

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeveriable	Beschreibung
stat.t	Für die Differenz der Mittelwerte berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade für die t-Statistik
stat. \bar{x}_1 , stat. \bar{x}_2	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>

Ausgabeveriable	Beschreibung
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Stichprobenumfang
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn <i>Verteilt</i> =1.

tvmFV()

Katalog >

tvmFV(*N,I,PV,Pmt,[PpY],[CpY], [PmtAt]*) \Rightarrow Wert

tvmFV(120,5,0,-500,12,12)

77641.1

Finanzfunktion, die den Geld-Endwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 179) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmI()

Katalog >

tvmI(*N,PV,Pmt,FV,[PpY],[CpY], [PmtAt]*) \Rightarrow Wert

tvmI(240,100000,-1000,0,12,12)

10.5241

Finanzfunktion, die den jährlichen Zinssatz berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 179) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmN()

Katalog >

tvmN(*I,PV,Pmt,FV,[PpY],[CpY], [PmtAt]*) \Rightarrow Wert

tvmN(5,0,-500,77641,12,12)

120.

Finanzfunktion, die die Anzahl der Zahlungsperioden berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 179) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPmt()**Katalog >** **tvmPmt**(*N,I,PV,FV,[PpY],[CpY],[PmtAt]*) \Rightarrow *Wert*

tvmPmt(60,4,30000,0,12,12)

-552.496

Finanzfunktion, die den Betrag der einzelnen Zahlungen berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 179) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPV()**Katalog >** **tvmPV**(*N,I,Pmt,FV,[PpY],[CpY],[PmtAt]*) \Rightarrow *Wert*

tvmPV(48,4,-500,30000,12,12)

-3426.7

Finanzfunktion, die den Barwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 179) beschrieben. Siehe auch **amortTbl()**, Seite 7.

TVM-Argumente*	Beschreibung	Datentyp
N	Anzahl der Zahlungsperioden	reelle Zahl
I	Jahreszinssatz	reelle Zahl
PV	Barwert	reelle Zahl
Pmt	Zahlungsbetrag	reelle Zahl
FV	Endwert	reelle Zahl
PpY	Zahlungen pro Jahr, Standard=1	Ganzzahl > 0
CpY	Verzinsungsperioden pro Jahr, Standard=1	Ganzzahl > 0
PmtAt	Zahlung fällig am Ende oder am Anfang der jeweiligen Zahlungsperiode, Standard=Ende	Ganzzahl (0=Ende, 1=Anfang)

* Die Namen dieser TVM-Argumente ähneln denen der TVM-Variablen (z.B. **tvm.pv** und **tvm.pmt**), die vom Finanzlöser der *Calculator* Applikation verwendet werden. Die Werte oder Ergebnisse der Argumente werden jedoch von den Finanzfunktionen nicht unter den TVM-Variablen gespeichert.

TwoVar *X, Y[, Häuf] [, Kategorie, Mit]*

Berechnet die 2-Variablen-Statistik. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 162.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes in numerischer Form oder als Zeichenfolge für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen *X*, *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Ein leeres (ungültiges) Element in einer der Listen *X1* bis *X20* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

AusgabevARIABLE	Beschreibung
stat. \bar{x}	Mittelwert der x-Werte
stat. x	Summe der x-Werte
stat. x2	Summe der x2-Werte
stat.sx	Stichproben-Standardabweichung von x
stat. x	Populations-Standardabweichung von x
stat.n	Anzahl der Datenpunkte

Ausgabeveriable	Beschreibung
stat. \bar{y}	Mittelwert der y-Werte
stat. y	Summe der y-Werte
stat. y ²	Summe der y ² -Werte
stat.sy	Stichproben-Standardabweichung von y
stat. y	Populations-Standardabweichung von y
Stat. xy	Summe der x · y-Werte
stat.r	Korrelationskoeffizient
stat.MinX	Minimum der x-Werte
stat.Q ₁ X	1. Quartil von x
stat.MedianX	Median von x
stat.Q ₃ X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.MinY	Minimum der y-Werte
stat.Q ₁ Y	1. Quartil von y
stat.MedY	Median von y
stat.Q ₃ Y	3. Quartil von y
stat.MaxY	Maximum der y-Werte
stat. (x-) ²	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert
stat. (y-) ²	Summe der Quadrate der Abweichungen der y-Werte vom Mittelwert

U

unitV() (Einheitsvektor)

Katalog >

unitV(Vektor1)⇒Vektor

Gibt je nach der Form von *Vektor1* entweder einen Zeilen- oder einen Spalteneinheitsvektor zurück.

Vektor1 muss eine einzeilige oder eine einspaltige Matrix sein.

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

$\text{unitV}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.408248 & 0.816497 & 0.408248 \end{bmatrix}$
$\text{unitV} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$	$\begin{bmatrix} 0.267261 \\ 0.534522 \\ 0.801784 \end{bmatrix}$

unLock**Katalog > ****unLock***Var1 [, Var2] [, Var3] ...***unLock***Var.*

Entsperrt die angegebenen Variablen bzw. die Variablengruppe. Gesperzte Variablen können nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 95, und **getLockInfo()**, Seite 71.

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

V**varPop() (Populationsvarianz)****Katalog > ****varPop**(*Liste*,
[*Häufigkeitsliste*]) \Rightarrow Ausdruck

varPop({5,10,15,20,25,30})

72.9167

Ergibt die Populationsvarianz von *Liste* zurück.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente enthalten.

Wenn ein Element in einer der Listen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Liste wird ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

varSamp() (Stichproben-Varianz)**Katalog > ****varSamp**(*Liste*[,
Häufigkeitsliste]) \Rightarrow Ausdruck

varSamp({1,2,5,-6,3,-2})

31

2

varSamp({1,3,5},{4,6,2})

68

33

Ergibt die Stichproben-Varianz von *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente enthalten.

Wenn ein Element in einer der Listen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Liste wird ebenfalls ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

varSamp(*Matrix1[, Häufigkeitsmatrix]*) \Rightarrow *Matrix*

Gibt einen Zeilenvektor zurück, der die Stichproben-Varianz jeder Spalte von *Matrix1* enthält.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Wenn ein Element in einer der Matrizen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Matrix wird ebenfalls ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 235).

Hinweis: *Matrix1* muss mindestens zwei Zeilen enthalten.

varSamp	$\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{pmatrix}$	$\begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$
varSamp	$\begin{pmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{pmatrix}$	$\begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}$ $\begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$

W

Wait

Wait *ZeitInSekunden*

Setzt die Ausführung für einen Zeitraum von *ZeitInSekunden* aus.

Wait ist besonders nützlich bei einem Programm, das eine kurze Verzögerung benötigt, damit die angeforderten Daten verfügbar werden.

Das Argument *ZeitInSekunden* muss ein Ausdruck sein, der zu einem Dezimalwert im Bereich von 0 bis 100 vereinfacht wird. Der Befehl runden diesen Wert auf die nächsten 0,1 Sekunden auf.

Zum Abbrechen eines **Wait** das gerade durchgeführt wird,

Um 4 Sekunden zu warten:

Wait 4

Um 1/2 Sekunde zu warten:

Wait 0.5

Um 1,3 Sekunden mithilfe der Variablen *seccount* zu warten:

seccount:=1.3

Wait seccount

Dieses Beispiel schaltet eine grüne LED 0,5 Sekunden lang ein und anschließend aus.

Send “SET GREEN 1 ON”

Wait 0.5

Send “SET GREEN 1 OFF”

- **Handheld:** Halten Sie die Taste  gedrückt und drücken Sie mehrmals .
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Sie können den Befehl **Wait** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

warnCodes ()

warnCodes(Ausdr1, StatusVar)⇒Ausdruck

Wertet den Ausdruck *Ausdr1* aus, gibt das Ergebnis zurück und speichert die Codes aller erzeugten Warnungen in der Listenvariablen *StatusVar*. Wenn keine Warnungen erzeugt werden, weist diese Funktion *StatusVar* eine leere Liste zu.

Ausdr1 kann jeder in TI-Nspire™ oder TI-Nspire™ CAS gültige mathematische Ausdruck sein. *Ausdr1* kann kein Befehl und keine Zuweisung sein.

StatusVar muss ein gültiger Variablenname sein.

Eine Liste der Warncodes und der zugehörigen Meldungen finden Sie (Seite 254).

 warnCodes($\det([1.23456 \cdot 10^{-999}])$, warn)	1.23456 $\cdot 10^{-999}$
warn	{10029}

when() (Wenn)

when(Bedingung, wahresErgebnis [, falschesErgebnis][, unbekanntesErgebnis])⇒Ausdruck

Gibt *wahresErgebnis*,
falschesErgebnis oder
unbekanntesErgebnis zurück, je nachdem,
ob die *Bedingung* wahr, falsch oder
unbekannt ist. Gibt die Eingabe zurück,
wenn zu wenige Argumente angegeben
werden.

Lassen Sie sowohl *falschesErgebnis* als
auch *unbekanntesErgebnis* weg, um einen
Ausdruck nur für den Bereich zu
bestimmen, in dem *Bedingung* wahr ist.

Geben Sie **undef** für *falschesErgebnis* an,
um einen Ausdruck zu bestimmen, der nur
in einem Intervall graphisch dargestellt
werden soll.

when() ist hilfreich für die Definition
rekursiver Funktionen.

when($x < 0, x + 3$) $x = 5$	undef
---------------------------------	-------

when($n > 0, n \cdot factorial(n - 1), 1$) → <i>factorial</i> (n)	Done
---	------

<i>factorial</i> (3)	6
----------------------	---

3!	6
----	---

While

While *Bedingung*

Block

EndWhile

Führt die in *Block* enthaltenen
Anweisungen so lange aus, wie *Bedingung*
wahr ist.

Block kann eine einzelne Anweisung oder
eine Serie von Anweisungen sein, die durch
“.” getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von
mehrzeiligen Programm- und
Funktionsdefinitionen finden Sie im
Abschnitt „Calculator“ des
Produkt handbuchs.

Define <i>sum_of_recip</i> (n)=Func	Done
Local $i, tempsum$	
$1 \rightarrow i$	
$0 \rightarrow tempsum$	
While $i \leq n$	
$tempsum + \frac{1}{i} \rightarrow tempsum$	
$i + 1 \rightarrow i$	
EndWhile	
Return <i>tempsum</i>	
EndFunc	

<i>sum_of_recip</i> (3)	11
-------------------------	----

xor (Boolesches exklusives oder)**Katalog > **

*BoolescherAusdr1***xor***BoolescherAusdr2*
ergibt Boolescher Ausdruck

true xor true	false
5>3 xor 3>5	true

*BoolescheListe1***xor***BoolescheListe2*
ergibt Boolesche Liste

*BoolescheMatrix1***xor***BoolescheMatrix2*
ergibt Boolesche Matrix

Gibt wahr zurück, wenn *BoolescherAusdr1* wahr und *BoolescherAusdr2* falsch ist und umgekehrt.

Gibt falsch zurück, wenn beide Argumente wahr oder falsch sind. Gibt einen vereinfachten Booleschen Ausdruck zurück, wenn eines der beiden Argumente nicht zu wahr oder falsch ausgewertet werden kann.

Hinweis: Siehe **or**, Seite 119.

Ganzzahl1 **xor** *Ganzzahl2* \Rightarrow *Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer xor-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis 1, wenn eines der Bits (nicht aber beide) 1 ist; das Ergebnis ist 0, wenn entweder beide Bits 0 oder beide Bits 1 sind. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O

0h7AC36 xor 0h3D5F	0h79169
--------------------	---------

Im Bin-Modus:

0b100101 xor 0b100	0b100001
--------------------	----------

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **►Base2**, Seite 17.

Hinweis: Siehe **or**, Seite 119.

Z**zInterval (z-Konfidenzintervall)**

zInterval σ ,*Liste*[,Häufigkeit[,KStufe]]

(Datenlisteneingabe)

zInterval σ, \bar{x}, n [,KStufe]

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabeverable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
stat. \bar{x}	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.sx	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert
stat. σ	Bekannte Populations-Standardabweichung für Datenfolge <i>Liste</i>

zInterval_1Prop (z-Konfidenzintervall für eine Proportion)

zInterval_1Prop x, n [,KStufe]

zInterval_1Prop (z-Konfidenzintervall für eine Proportion)

Katalog > 

Berechnet ein z-Konfidenzintervall für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

x ist eine nicht negative Ganzzahl.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. \hat{p}	Die berechnete Erfolgsproportion
stat.ME	Fehlertoleranz
stat.n	Anzahl der Stichproben in Datenfolge

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen)

Katalog > 

zInterval_2Prop $x1,n1,x2,n2,[KStufe]$

Berechnet das z-Konfidenzintervall für zwei Proportionen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

$x1$ und $x2$ sind nicht negative Ganzzahlen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. \hat{p} Diff	Die geschätzte Differenz zwischen den Proportionen
stat.ME	Fehlertoleranz
stat. $\hat{p}1$	Geschätzte erste Stichprobenproportion
stat. $\hat{p}2$	Geschätzte zweite Stichprobenproportion

Ausgabevariable	Beschreibung
stat.n1	Stichprobenumfang in Datenfolge eins
stat.n2	Stichprobenumfang in Datenfolge zwei

zInterval_2Samp (z-Konfidenzintervall für zwei Stichproben)

Katalog >

zInterval_2Samp $\sigma_1, \sigma_2, Liste1, Liste2$
 $[, Häufigkeit1, Häufigkeit2, [KStufe]]$

(Datenlisteneingabe)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}_1, n1, \bar{x}_2, n2$
 $[, KStufe]$

(Zusammenfassende statistische Eingabe)

Berechnet ein z-Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. $\bar{x}_1 - \bar{x}_2$	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat. \bar{x}_1 , stat. \bar{x}_2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat. σ_x1 , stat. σ_x2	Stichproben-Standardabweichungen für <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.r1, stat.r2	Bekannte Populations-Standardabweichungen für Datenfolge <i>Liste 1</i> und <i>Liste 2</i>

zTest

Katalog >

zTest $\mu_0, \sigma, Liste, [Häufigkeit, [Hypoth]]$

(Datenlisteneingabe)

zTest $\mu0,\sigma,\bar{x},n,[Hypothesis]$

(Zusammenfassende statistische Eingabe)

Führt einen *z*-Test mit der Häufigkeit
Häufigkeitsliste durch. Eine
 Zusammenfassung der Ergebnisse wird in
 der Variable *stat.results* gespeichert. (Seite
 162.)

Getestet wird $H_0: \mu = \mu_0$ in Bezug auf eine
 der folgenden Alternativen:

Für $H_a: \mu < \mu_0$ setzen Sie *Hypothesis<0*

Für $H_a: \mu \neq \mu_0$ (Standard) setzen Sie
Hypothesis=0

Für $H_a: \mu > \mu_0$ setzen Sie *Hypothesis>0*

Informationen zu den Auswirkungen leerer
 Elemente in einer Liste finden Sie unter
 "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.z	$(\bar{x} - \mu_0) / (\sigma / \sqrt{n})$
stat.P Value	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat. \bar{x}	Stichprobenmittelwert der Datenfolge in <i>Liste</i>
stat.sx	Stichproben-Standardabweichung der Datenfolge. Wird nur für <i>Dateneingabe</i> zurückgegeben.
stat.n	Stichprobenumfang

zTest_1Prop (z-Test für eine Proportion)

zTest_1Prop $p0,x,n,[Hypothesis]$

Berechnet einen *z*-Test für eine Proportion.
 Eine Zusammenfassung der Ergebnisse wird
 in der Variable *stat.results* gespeichert.
 (Siehe Seite 162.)

x ist eine nicht negative Ganzzahl.

Getestet wird $H_0: p = p_0$ in Bezug auf eine
 der folgenden Alternativen:

zTest_1Prop (z-Test für eine Proportion)

Katalog > 

Für $H_a: p > p0$ setzen Sie *Hypoth>0*

Für $H_a: p \neq p0$ (*Standard*) setzen Sie
Hypoth=0

Für $H_a: p < p0$ setzen Sie *Hypoth<0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

AusgabevARIABLE	Beschreibung
stat.p0	Hypothetische Populations-Standardabweichung
stat.z	Für die Proportion berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \hat{p}	Geschätzte Stichprobenproportion
stat.n	Stichprobenumfang

zTest_2Prop (z-Test für zwei Proportionen)

Katalog > 

zTest_2Prop x1,n1,x2,n2[,Hypoth]

Berechnet einen *z*-Test für zwei Proportionen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

x1 und *x2* sind nicht negative Ganzahlen.

Getestet wird $H_0: p1 = p2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: p1 > p2$ setzen Sie *Hypoth>0*

Für $H_a: p1 \neq p2$ (*Standard*) setzen Sie
Hypoth=0

Für $H_a: p1 < p2$ setzen Sie *Hypoth<0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.z	Für die Differenz der Proportionen berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \hat{p} 1	Geschätzte erste Stichprobenproportion
stat. \hat{p} 2	Geschätzte zweite Stichprobenproportion
stat. \hat{p}	Geschätzte verteilte Stichprobenproportion
stat.n1, stat.n2	Stichprobenanzahl in Versuchen 1 und 2

zTest_2Samp (z-Test für zwei Stichproben)

Katalog > 

zTest_2Samp $\sigma_1, \sigma_2, Liste1, Liste2$
 $[, Häufigkeit1, Häufigkeit2[, Hypoth]]]$

(Datenlisteneingabe)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2[, Hypoth]$

(Zusammenfassende statistische Eingabe)

Berechnet einen *z*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 162.)

Getestet wird $H_0: \mu_1 = \mu_2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu_1 < \mu_2$ setzen Sie *Hypoth<0*

Für $H_a: \mu_1 \neq \mu_2$ (Standard) setzen Sie *Hypoth=0*

Für $H_a: \mu_1 > \mu_2$ setzen Sie *Hypoth>0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 235).

Ausgabevariable	Beschreibung
stat.z	Für die Differenz der Mittelwerte berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \bar{x} 1, stat. \bar{x} 2	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>

Ausgabeveriable	Beschreibung
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Stichprobenumfang

Sonderzeichen

+ (addieren)

Taste

Wert1 + Wert2 \Rightarrow *Wert*

Gibt die Summe der beiden Argumente zurück.

56	56
56+4	60
60+4	64
64+4	68
68+4	72

Liste1 + Liste2 \Rightarrow *Liste*

Matrix1 + Matrix2 \Rightarrow *Matrix*

Gibt eine Liste (bzw. eine Matrix) zurück, die die Summen der entsprechenden Elemente von *Liste1* und *Liste2* (oder *Matrix1* und *Matrix2*) enthält.

Die Argumente müssen die gleiche Dimension besitzen.

$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow I1$	$\{ 22, 3.14159, 1.5708 \}$
$\left\{ 10, 5, \frac{\pi}{2} \right\} \rightarrow I2$	$\{ 10, 5, 1.5708 \}$
$I1 + I2$	$\{ 32, 8.14159, 3.14159 \}$

Wert + Liste1 \Rightarrow *Liste*

Liste1 + Wert \Rightarrow *Liste*

Gibt eine Liste zurück, die die Summen von *Wert* plus jedem Element der *Liste1* enthält.

$15 + \{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
$\{ 10, 15, 20 \} + 15$	$\{ 25, 30, 35 \}$

Wert + Matrix1 \Rightarrow *Matrix*

Matrix1 + Wert \Rightarrow *Matrix*

Gibt eine Matrix zurück, in der *Wert* zu jedem Element der Diagonalen von *Matrix1* addiert ist. *Matrix1* muss eine quadratische Matrix sein.

$20 + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
---	--

Hinweis: Verwenden Sie **.+** (Punkt Plus) zum Addieren eines Ausdrucks zu jedem Element.

-(subtrahieren) **Taste** $Wert1 - Wert2 \Rightarrow Wert$

$$6 - 2 \quad 4$$

Gibt *Wert1* minus *Wert2* zurück.

$$\pi - \frac{\pi}{6} \quad 2.61799$$

 $Liste1 - Liste2 \Rightarrow Liste$

$$\left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10, 5, \frac{\pi}{2} \right\} \quad \{ 12, -1.85841, 0. \}$$

 $Matrix1 - Matrix2 \Rightarrow Matrix$

$$\begin{bmatrix} 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 2 \end{bmatrix}$$

Subtrahiert die einzelnen Elemente aus *Liste2* (oder *Matrix2*) von denen in *Liste1* (oder *Matrix1*) und gibt die Ergebnisse zurück.

Die Argumente müssen die gleiche Dimension besitzen.

 $Wert - Liste1 \Rightarrow Liste$

$$15 - \{ 10, 15, 20 \} \quad \{ 5, 0, -5 \}$$

 $Liste1 - Wert \Rightarrow Liste$

$$\{ 10, 15, 20 \} - 15 \quad \{ -5, 0, 5 \}$$

Subtrahiert jedes Element der *Liste1* von *Wert* oder subtrahiert *Wert* von jedem Element der *Liste1* und gibt eine Liste der Ergebnisse zurück.

 $Wert - Matrix1 \Rightarrow Matrix$

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

 $Matrix1 - Wert \Rightarrow Matrix$

Wert - Matrix1 gibt eine Matrix zurück, die *Wert* multipliziert mit der Einheitsmatrix minus *Matrix1* ist. *Matrix1* muss eine quadratische Matrix sein.

Matrix1 - Wert gibt eine Matrix zurück, die *Wert* multipliziert mit der Einheitsmatrix subtrahiert von *Matrix1* ist. *Matrix1* muss eine quadratische Matrix sein.

Hinweis: Verwenden Sie $-$ (Punkt Minus) zum Subtrahieren eines Ausdrucks von jedem Element.

·(multiplizieren) **Taste** $Wert1 \cdot Wert2 \Rightarrow Wert$

$$2 \cdot 3.45 \quad 6.9$$

Gibt das Produkt der beiden Argumente zurück.

·(multiplizieren)

Taste

Liste1·*Liste2*⇒*Liste*

Gibt eine Liste zurück, die die Produkte der entsprechenden Elemente aus *Liste1* und *Liste2* enthält.

Die Listen müssen die gleiche Dimension besitzen.

Matrix1·*Matrix2*⇒*Matrix*

Gibt das Matrizenprodukt von *Matrix1* und *Matrix2* zurück.

$$\{1,2,3\} \cdot \{4,5,6\}$$

$$\{4,10,18\}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 7 & 8 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 42 & 48 \\ 105 & 120 \\ 105 & 120 \end{bmatrix}$$

Die Spaltenanzahl von *Matrix1* muss gleich die Zeilenanzahl von *Matrix2* sein.

Wert·*Liste1*⇒*Liste*

$$\pi \cdot \{4,5,6\} = \{12.5664, 15.708, 18.8496\}$$

Liste1·*Wert*⇒*Liste*

Gibt eine Liste zurück, die die Produkte von *Wert* und jedem Element der *Liste1* enthält.

Wert·*Matrix1*⇒*Matrix*

Matrix1·*Wert*⇒*Matrix*

Gibt eine Matrix zurück, die die Produkte von *Wert* und jedem Element der *Matrix1* enthält.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

$$6 \cdot \text{identity}(3) = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Hinweis: Verwenden Sie . ·(Punkt-Multiplikation) zum Multiplizieren eines Ausdrucks mit jedem Element.

/ (dividieren)

Taste

Wert1/*Wert2*⇒*Wert*

$$\frac{2}{3.45} = 0.57971$$

Gibt *Wert1* dividiert durch *Wert2* zurück.

Hinweis: Siehe auch **Vorlage Bruch**, Seite 1.

Liste1/*Liste2*⇒*Liste*

$$\begin{bmatrix} 1,2,3 \\ 4,5,6 \end{bmatrix} \rightarrow \left\{ 0.25, \frac{2}{5}, \frac{1}{2} \right\}$$

Gibt eine Liste der Elemente von *Liste1* dividiert durch *Liste2* zurück.

Die Listen müssen die gleiche Dimension besitzen.

/ (dividieren)

 Taste

Wert / Liste1 \Rightarrow *Liste*

$$\frac{6}{\{3,6,\sqrt{6}\}} \quad \{2,1,2.44949\}$$

Liste1 / Wert \Rightarrow *Liste*

$$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2} \quad \left\{ \frac{1}{18}, \frac{1}{14}, \frac{1}{63} \right\}$$

Gibt eine Liste der Elemente von *Wert* dividiert durch *Liste1* oder *Liste1* dividiert durch *Wert* zurück.

Wert / Matrix1 \Rightarrow *Matrix*

$$\frac{[7 \ 9 \ 2]}{7 \cdot 9 \cdot 2} \quad \left[\begin{array}{ccc} \frac{1}{18} & \frac{1}{14} & \frac{1}{63} \end{array} \right]$$

Matrix1 / Wert \Rightarrow *Matrix*

Gibt eine Matrix zurück, die die Quotienten *Matrix1/Wert* enthält.

Hinweis: Verwenden Sie . / (Punkt-Division) zum Dividieren eines Ausdrucks durch jedes Element.

\wedge (Potenz)

 Taste

Wert1 \wedge Wert2 \Rightarrow *Wert*

$$4^2 \quad 16$$

Liste1 \wedge Liste2 \Rightarrow *Liste*

$$\frac{\{2,4,6\}}{\{1,2,3\}} \quad \{2,16,216\}$$

Gibt das erste Argument hoch dem zweiten Argument zurück.

Hinweis: Siehe auch **Vorlage Exponent**, Seite 1.

Bei einer Liste wird jedes Element aus *Liste1* hoch dem entsprechenden Element aus *Liste2* zurückgegeben.

Im reellen Bereich benutzen Bruchpotenzen mit gekürztem ungeradem Nenner den reellen statt den Hauptzeig im komplexen Modus.

Wert \wedge Liste1 \Rightarrow *Liste*

$$\frac{\pi}{\{1,2,-3\}} \quad \{3.14159, 9.8696, 0.032252\}$$

Gibt *Wert* hoch den Elementen von *Liste1* zurück.

Liste1 \wedge Wert \Rightarrow *Liste*

$$\frac{\{1,2,3,4\}^{-2}}{\{-1,0,1\}} \quad \left\{ \frac{1}{4}, \frac{1}{9}, \frac{1}{16} \right\}$$

Gibt die Elemente von *Liste1* hoch *Wert* zurück.

\wedge (Potenz)

Taste

Quadratmatrix1 \wedge *Ganzzahl* \Rightarrow Matrix

Gibt *Quadratmatrix1* hoch *Ganzzahl* zurück.

Quadratmatrix1 muss eine quadratische Matrix sein.

Ist *Ganzzahl* = -1, wird die inverse Matrix berechnet.

Ist *Ganzzahl* < -1, wird die inverse Matrix hoch der entsprechenden positiven Zahl berechnet.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2$	$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1}$	$\begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2}$	$\begin{bmatrix} \frac{11}{4} & -\frac{5}{4} \\ 2 & 2 \\ -\frac{15}{4} & \frac{7}{4} \end{bmatrix}$

x^2 (Quadrat)

Taste

*Wert1*² \Rightarrow *Wert*

Gibt das Quadrat des Arguments zurück.

*Liste1*² \Rightarrow *Liste*

Gibt eine Liste zurück, die die Produkte der Elemente in *Liste1* enthält.

*Quadratmatrix1*² \Rightarrow Matrix

Gibt das Matriz-Quadrat von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Quadrats jedes einzelnen Elements. Verwenden Sie \cdot^2 , um das Quadrat jedes einzelnen Elements zu berechnen.

4^2	16
$\{2,4,6\}^2$	$\{4,16,36\}$
$\begin{bmatrix} 2 & 4 & 6 \end{bmatrix}^2$	$\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$
$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} \cdot^2$	$\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$

.+ (Punkt-Addition)

Tasten

Matrix1 .+ *Matrix2* \Rightarrow Matrix

Wert .+ *Matrix1* \Rightarrow Matrix

Matrix1 .+ *Matrix2* gibt eine Matrix zurück, die Summe jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Wert .+ *Matrix1* gibt eine Matrix zurück, die die Summe von Zahl und jedem Element von *Matrix1* ist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$
$5 .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$

. - (Punkt-Subt.)

Tasten

Matrix1 . - Matrix2 \Rightarrow Matrix

Wert . - Matrix1 \Rightarrow Matrix

Matrix1 . - Matrix2 gibt eine Matrix zurück,
die die Differenz jedes Elementpaars von
Matrix1 und *Matrix2* ist.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} . - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} -9 & -18 \\ -27 & -36 \end{bmatrix}$$

$$5 . - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} -5 & -15 \\ -25 & -35 \end{bmatrix}$$

Wert . - Matrix1 gibt eine Matrix zurück, die
die Differenz von *Wert* und jedem Element
von *Matrix1* ist.

. · (Punkt-Mult.)

Tasten

Matrix1 . · Matrix2 \Rightarrow Matrix

Wert . · Matrix1 \Rightarrow Matrix

Matrix1 . · Matrix2 gibt eine Matrix zurück,
die das Produkt jedes Elementpaars von
Matrix1 und *Matrix2* ist.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} . \cdot \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$$

$$5 . \cdot \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 50 & 100 \\ 150 & 200 \end{bmatrix}$$

Wert . · Matrix1 Gibt eine Matrix zurück, die
die Produkte von *Wert* und jedem Element
der *Matrix1* enthält.

. / (Punkt-Division)

Tasten

Matrix1 . / Matrix2 \Rightarrow Matrix

Wert . / Matrix1 \Rightarrow Matrix

Matrix1 . / Matrix2 gibt eine Matrix
zurück, die der Quotient jedes
Elementpaars von *Matrix1* und *Matrix2* ist.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} . / \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{10} \end{bmatrix}$$

$$5 . / \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{8} \end{bmatrix}$$

Wert . / Matrix1 gibt eine Matrix zurück,
die der Quotient von *Wert* und jedem
Element von *Matrix1* ist.

= (gleich)

= Taste

Ausdr1 = Ausdr2 \Rightarrow Boolescher Ausdruck

Liste1 = Liste2 \Rightarrow Boolesche Liste

Matrix1 = Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn Ausdr1 bei Auswertung gleich Ausdr2 ist.

Gibt falsch zurück, wenn Ausdr1 bei Auswertung ungleich Ausdr2 ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Done

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

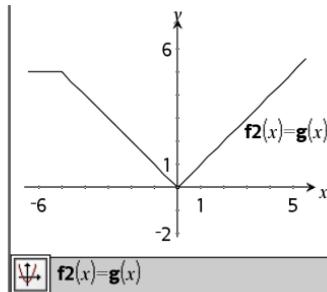
Beispelfunktion mit den mathematischen

Vergleichssymbolen: =, ≠, <, ≤, >, ≥

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
```

Ergebnis der graphischen Darstellung g(x)



≠ (ungleich)

ctrl Tasten

Ausdr1 ≠ Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 ≠ Liste2 \Rightarrow Boolesche Liste

Matrix1 ≠ Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn Ausdr1 bei Auswertung ungleich Ausdr2 ist.

\neq (ungleich)

ctrl **=** Tasten

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie **/= eintippen**

$<$ (kleiner als)

ctrl **=** Tasten

Ausdr1 < Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 < Liste2 \Rightarrow Boolesche Liste

Matrix1 < Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung kleiner als *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung größer oder gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

\leq (kleiner oder gleich)

ctrl **=** Tasten

Ausdr1 \leq Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 \leq Liste2 \Rightarrow Boolesche Liste

Matrix1 \leq Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung kleiner oder gleich *Ausdr2* ist.

\leq (kleiner oder gleich)

ctrl **=** Tasten

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung größer als *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel <=

$>$ (größer als)

ctrl **=** Tasten

Ausdr1 > *Ausdr2* \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 > *Liste2* \Rightarrow Boolesche Liste

Matrix1 > *Matrix2* \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung größer als *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung kleiner oder gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

\geq (größer oder gleich)

ctrl **=** Tasten

Ausdr1 \geq *Ausdr2* \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 \geq *Liste2* \Rightarrow Boolesche Liste

Matrix1 \geq *Matrix2* \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung größer oder gleich *Ausdr2* ist.

\geq (größer oder gleich)

ctrl = Tasten

Gibt falsch zurück, wenn Ausdr1 bei Auswertung kleiner oder gleich Ausdr2 ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel $>=$

\Rightarrow (logische Implikation)

ctrl = Tasten

BoolescherAusdr1 \Rightarrow BoolescherAusdr2 ergibt Boolescher Ausdruck

BoolescheListe1 \Rightarrow BoolescheListe2 ergibt Boolesche Liste

BoolescheMatrix1 \Rightarrow BoolescheMatrix2 ergibt Boolesche Matrix

Ganzzahl1 \Rightarrow Ganzzahl2 ergibt Ganzzahl

5>3 or 3>5	true
5>3 \Rightarrow 3>5	false
3 or 4	7
3 \Rightarrow 4	-4
$\{1,2,3\}$ or $\{3,2,1\}$	$\{3,2,3\}$
$\{1,2,3\} \Rightarrow \{3,2,1\}$	$\{-1,-1,-3\}$

Wertet den Ausdruck **not** <Argument1> **or** <Argument2> aus und gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel $=>$

\Leftrightarrow (logische doppelte Implikation,
XNOR)

ctrl = Tasten

BoolescherAusdr1 \Leftrightarrow BoolescherAusdr2
ergibt Boolescher Ausdruck

BoolescheList1 \Leftrightarrow BoolescheList2
ergibt Boolesche Liste

BoolescheMatrix1 \Leftrightarrow BoolescheMatrix2
ergibt Boolesche Matrix

Ganzzahl1 \Leftrightarrow Ganzzahl2 ergibt Ganzzahl

5>3 xor 3>5	true
5>3 \Leftrightarrow 3>5	false
3 xor 4	7
3 \Leftrightarrow 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} \Leftrightarrow {3,2,1}	{-3,-1,-3}

Gibt die Negation einer **XOR** booleschen Operation auf beiden Argumenten zurück.
Gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie $<=>$ drücken

! (Fakultät)

?! Taste

Wert1! \Rightarrow Wert

5!
120

Liste1! \Rightarrow Liste

{5,4,3}!
{120,24,6}

Matrix1! \Rightarrow Matrix

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}!$
 $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Gibt die Fakultät des Arguments zurück.

Bei Listen und Matrizen wird eine Liste/Matrix mit der Fakultät der einzelnen Elemente zurückgegeben.

&

/k Tasten

String1 & String2 \Rightarrow String

"Hello "&"Nick"
"Hello Nick"

Gibt einen String zurück, der durch Anfügen von *String2* an *String1* gebildet wurde.

d() (Ableitung)

d(Ausdr1, Var[, Ordnung]) |
 $Var = \text{Wert} \Rightarrow \text{Wert}$

d(Ausdr1, Var[, Ordnung]) \Rightarrow Wert

d(Liste1, Var[, Ordnung]) \Rightarrow Liste

d(Matrix1, Var[, Ordnung]) \Rightarrow Matrix

Außer bei der ersten Syntax müssen Sie einen Zahlenwert in der Variablen *Var* speichern, bevor Sie **d()** auswerten. Siehe hierzu die Beispiele.

d() lässt sich verwenden, um die erste und zweite Ableitung an einem Punkt numerisch durch automatische Ableitungsmethoden zu berechnen.

Ordnung (falls angegeben) muss **1** oder **2** sein. Die Vorgabe ist **1**.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **derivative (...)** eintippen.

Hinweis: Siehe auch **Erste Ableitung**, Seite 5, und **Zweite Ableitung**, Seite 6.

Hinweis: Der Algorithmus von **d()** hat eine Einschränkung: Er arbeitet den nicht vereinfachten Ausdruck rekursiv ab und berechnet dabei den numerischen Wert der ersten (und ggf. der zweiten) Ableitung sowie die Auswertung jedes Unterausdrucks. Dies kann zu unerwarteten Ergebnissen führen.

Hierzu rechts ein Beispiel. Die erste Ableitung von $x \cdot (x^2+x)^{(1/3)}$ bei $x=0$ ist gleich 0. Nun ist allerdings die erste Ableitung des Unterausdrucks $(x^2+x)^{(1/3)}$ bei $x=0$ nicht definiert. Dieser Wert wird gleichzeitig jedoch verwendet, um die Ableitung des Gesamtausdrucks zu berechnen. Daher meldet **d()** das Ergebnis als nicht definiert und zeigt eine Warnmeldung an.

$\frac{d}{dx}(x) _{x=0}$	undef
$x:=0: \frac{d}{dx}(x)$	undef
$x:=3: \frac{d}{dx}\left(\{x^2, x^3, x^4\}\right)$	{6, 27, 108}

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

Wenn Sie bei der Arbeit auf diese Beschränkung stoßen, prüfen Sie die Lösung grafisch. Ggf. können Sie es auch mit **centralDiff()** probieren.

J() (Integral)

$J(Ausdr1, Var, Untere, Obere) \Rightarrow Wert$

Gibt das Integral von *Ausdr1* bezüglich der Variablen *Var* von *Untere* bis *Obere* zurück. Hiermit können Sie das bestimmte Integral numerisch berechnen. Hierzu wird dieselbe Methode wie bei **nInt()** verwendet.

$$\int_0^1 x^2 \, dx \quad 0.333333$$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **Integral (...) eintippen**.

Hinweis: Siehe auch **nInt()**, Seite 112, und **Vorlage Bestimmtes Integral**, Seite 6.

 $\sqrt()$ (Quadratwurzel)

$\sqrt(Wert1) \Rightarrow Wert$

$$\sqrt{4} \quad 2$$

$\sqrt(Liste1) \Rightarrow Liste$

$$\sqrt{\{9,2,4\}} \quad \{3,1.41421,2\}$$

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **sqrt (...) eintippen**.

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 1.

$\Pi()$ (ProdSeq) $\Pi(Ausdr1, Var, Von, Bis) \Rightarrow Ausdruck$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **prodSeq**(...) eintippen.

Wertet *Ausdr1* für jeden Wert von *Var* zwischen *Von* und *Bis* aus und gibt das Produkt der Ergebnisse zurück.

Hinweis: Siehe auch **Vorlage Produkt** (Π), Seite 5.

 $\Pi(Ausdr1, Var, Von, Von-1) \Rightarrow 1$ $\Pi(Ausdr1, Var, Von, Bis) \Rightarrow 1 / \Pi(Ausdr1, Var, Bis+1, Von-1)$ if *Bis* < *Von-1*

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

$$\overline{\prod_{n=1}^5 \left(\left\{ \frac{1}{n}, n, 2 \right\} \right)} \quad \left\{ \frac{1}{120}, 120, 32 \right\}$$

Die verwendeten Produktformeln wurden ausgehend von der folgenden Quelle entwickelt:

Ronald L. Graham, Donald E. Knuth, Oren Patashnik: *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley 1994.

$$\overline{\prod_{k=4}^3 (k)} \quad 1$$

$$\overline{\prod_{k=4}^1 \left(\frac{1}{k} \right)} \quad 6$$

$$\overline{\prod_{k=4}^1 \left(\frac{1}{k} \right) \cdot \prod_{k=2}^4 \left(\frac{1}{k} \right)} \quad \frac{1}{4}$$

 $\Sigma()$ (SumSeq) $\Sigma(Ausdr1, Var, Von, Bis) \Rightarrow Ausdruck$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **sumSeq**(...) eintippen.

Wertet *Ausdr1* für jeden Wert von *Var* zwischen *Von* und *Bis* aus und gibt die Summe der Ergebnisse zurück.

Hinweis: Siehe auch **Vorlage Summe**, Seite 5.

 $\Sigma(Ausdr1, Var, Von, Von-1) \Rightarrow 0$ $\Sigma(Ausdr1, Var, Von, Bis) \Rightarrow -\Sigma(Ausdr1, Var, Bis+1, Von-1)$ if *Bis* < *Von-1*

$$\overline{\sum_{n=1}^5 \left(\frac{1}{n} \right)} \quad \frac{137}{60}$$

$$\overline{\sum_{k=4}^3 (k)} \quad 0$$

$\Sigma()$ (SumSeq)

Katalog >

Die verwendeten Summenformeln wurden ausgehend von der folgenden Quelle entwickelt:

Ronald L. Graham, Donald E. Knuth, Oren Patashnik: *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley 1994.

$$\begin{array}{c} \sum_{k=4}^1 (k) \\ \hline \sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) \end{array}$$

-5

4

$\SigmaInt()$

Katalog >

$\SigmaInt(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) \Rightarrow Wert$

$\SigmaInt(1,3,12,4.75,20000,,12,12)$ -213.48

$\SigmaInt(NPmt1, NPmt2, AmortTabelle) \Rightarrow Wert$

Amortisationsfunktion, die die Summe der Zinsen innerhalb eines angegebenen Zahlungsbereichs berechnet.

$NPmt1$ und $NPmt2$ definieren Anfang und Ende des Zahlungsbereichs.

$N, I, PV, Pmt, FV, PpY, CpY$ und $PmtAt$ werden in der TVM-Argumentetabelle (Seite 179) beschrieben.

- Wenn Sie Pmt nicht angeben, wird standardmäßig $Pmt=tvmPmt$ ($N, I, PV, FV, PpY, CpY, PmtAt$) eingesetzt.
- Wenn Sie FV nicht angeben, wird standardmäßig $FV=0$ eingesetzt.
- Die Standardwerte für PpY , CpY und $PmtAt$ sind dieselben wie bei den TVM-Funktionen.

$WertRunden$ legt die Anzahl der Dezimalstellen für das Runden fest. Standard=2.

$tbl:=amortTbl([12,12,4.75,20000,,12,12])$

0	0.	0.	20000.
1	-77.49	-1632.43	18367.6
2	-71.17	-1638.75	16728.8
3	-64.82	-1645.1	15083.7
4	-58.44	-1651.48	13432.2
5	-52.05	-1657.87	11774.4
6	-45.62	-1664.3	10110.1
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$\SigmaInt(1,3,tbl)$ -213.48

$\SigmaInt(NPmt1, NPmt2, AmortTable)$ berechnet die Summe der Zinsen auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle* muss eine Matrix in der unter **amortTbl()**, Seite 7, beschriebenen Form sein.

$\Sigma\text{Int}()$

Katalog >

Hinweis: Siehe auch $\Sigma\text{Prn}()$ auf dieser und $\text{Bal}()$, Seite 16.

$\Sigma\text{Prn}()$

Katalog >

$\Sigma\text{Prn}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) \Rightarrow Wert$

$\Sigma\text{Prn}(NPmt1, NPmt2, AmortTabelle) \Rightarrow Wert$

Amortisationsfunktion, die die Summe der Tilgungszahlungen innerhalb eines angegebenen Zahlungsbereichs berechnet.

$NPmt1$ und $NPmt2$ definieren Anfang und Ende des Zahlungsbereichs.

$N, I, PV, Pmt, FV, PpY, CpY$ und $PmtAt$ werden in der TVM-Argumentetabelle (Seite 179) beschrieben.

- Wenn Sie Pmt nicht angeben, wird standardmäßig $Pmt=\text{tvmPmt}$ ($N, I, PV, FV, PpY, CpY, PmtAt$) eingesetzt.
- Wenn Sie FV nicht angeben, wird standardmäßig $FV=0$ eingesetzt.
- Die Standardwerte für PpY , CpY und $PmtAt$ sind dieselben wie bei den TVM-Funktionen.

$WertRunden$ legt die Anzahl der Dezimalstellen für das Runden fest. Standard=2.

$\Sigma\text{Prn}(NPmt1, NPmt2, AmortTabelle)$ berechnet die Summe der gezahlten Tilgungsbeträge auf der Grundlage der Amortisationstabelle $AmortTabelle$. Das Argument $AmortTabelle$ muss eine Matrix in der unter $\text{amortTbl}()$, Seite 7, beschriebenen Form sein.

Hinweis: Siehe auch $\Sigma\text{Int}()$ auf dieser und $\text{Bal}()$, Seite 16.

$\Sigma\text{Prn}(1, 3, 12, 4.75, 20000, , 12, 12)$ -4916.28

$tbl:=\text{amortTbl}(12, 12, 4.75, 20000, , 12, 12)$

0	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	-58.44	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	-45.62	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$\Sigma\text{Prn}(1, 3, tbl)$ -4916.28

(Umleitung)

ctrl Tasten

varNameString

Greift auf die Variable namens *VarNameString* zu. So können Sie innerhalb einer Funktion Variablen unter Verwendung von Strings erzeugen.

xyz:=12	12
#("x"&"y"&"z")	12

Erzeugt oder greift auf die Variable xyz zu.

10→r	10
"r"→sI	"r"
#sI	10

Gibt den Wert der Variable (r) zurück, dessen Name in Variable s1 gespeichert ist.

E (Wissenschaftliche Schreibweise)

EE Taste

MantisseEExponent

Gibt eine Zahl in wissenschaftlicher Schreibweise ein. Die Zahl wird als Mantisse × 10Exponent interpretiert.

23000.	23000.
2300000000.+4.1e15	4.1e15
3·10 ⁴	30000

Tipp: Wenn Sie eine Potenz von 10 eingeben möchten, ohne ein Dezimalwertergebnis zu verursachen, verwenden Sie 10^Ganzzahl.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @E eintippen. Tippen Sie zum Beispiel 2.3@E4 ein, um 2.3E4 einzugeben.

g (Neugrad)

1 Taste

AusdrIg⇒Ausdruck

Im Grad-, Neugrad- oder Bogenmaß-Modus:

AusdrIg⇒Ausdruck

cos(50°)	0.707107
cos({0,100°,200°})	{1,0.,-1.}

Listelg⇒Liste

MatrixIg⇒Matrix

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Bogenmaß-Modus einen Winkel in Neugrad anzugeben.

Im Winkelmodus Bogenmaß wird AusdrI mit $\pi/200$ multipliziert.

g (Neugrad)

1 Taste

Im Winkelmodus Grad wird *Ausdr1* mit $g/100$ multipliziert.

Im Neugrad-Modus wird *Ausdr1* unverändert zurückgegeben.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @g eintippen.

' (Bogenmaß)

1 Taste

Wert1'⇒*Wert*

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

Liste1'⇒*Liste*

Matrix1'⇒*Matrix*

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

$$\cos\left(\left\{0^r, \left(\frac{\pi}{12}\right)^r, -(\pi)^r\right\}\right) \quad \{1, 0.965926, -1.\}$$

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Neugrad-Modus einen Winkel im Bogenmaß anzugeben.

Im Winkelmodus Grad wird das Argument mit $180/\pi$ multipliziert.

Im Winkelmodus Bogenmaß wird das Argument unverändert zurückgegeben.

Im Neugrad-Modus wird das Argument mit $200/\pi$ multipliziert.

Tipp: Verwenden Sie ' in einer Funktionsdefinition, wenn Sie bei Ausführung der Funktion das Bogenmaß frei von der Winkelmoduseinstellung erzwingen möchten.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @r eintippen.

° (Grad)

1 Taste

Wert1°⇒*Wert*

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

Liste1°⇒*Liste*

Matrix1°⇒*Matrix*

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

Im Winkelmodus Bogenmaß:

° (Grad)

1 Taste

Diese Funktion gibt Ihnen die Möglichkeit, im Neugrad- oder Bogenmaß-Modus einen Winkel in Grad anzugeben.

Im Winkelmodus Bogenmaß wird das Argument mit $\pi/180$ multipliziert.

Im Winkelmodus Grad wird das Argument unverändert zurückgegeben.

Im Winkelmodus Neugrad wird das Argument mit $10/9$ multipliziert.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @d eintippen.

°, ', " (Grad/Minute/Sekunde)

ctrl Tasten

$dd^{\circ}mm'ss.ss \Rightarrow$ Ausdruck

Im Grad-Modus:

ddEine positive oder negative Zahl

25°13'17.5"

25.2215

mmEine nicht negative Zahl

25°30'

51

ss.ssEine nicht negative Zahl

2

Gibt $dd + (mm/60) + (ss.ss/3600)$ zurück.

Mit einer solchen Eingabe auf der 60er-Basis können Sie:

- Einen Winkel unabhängig vom aktuellen Winkelmodus in Grad/Minuten/Sekunden eingeben.
- Uhrzeitangaben in Stunden/Minuten/Sekunden vornehmen.

Hinweis: Nach ss.ss werden zwei Apostrophe (') gesetzt, kein Anführungszeichen (").

∠ (Winkel)

ctrl Tasten

[Radius, $\angle\theta$ _Winkel] \Rightarrow Vektor

Im Bogenmaß-Modus mit Vektorformat eingestellt auf:

(Eingabe polar)

kartesisch

[Radius, $\angle\theta$ _Winkel, Z_Koordinate] \Rightarrow Vektor

\angle (Winkel)

ctrl



Tasten

(Eingabe zylindrisch)

 $[Radius, \angle\theta_Winkel, \angle\theta_Winkel] \Rightarrow Vektor$

(Eingabe sphärisch)

Gibt Koordinaten als Vektor zurück, wobei die aktuelle Einstellung für Vektorformat gilt: kartesisch, zylindrisch oder sphärisch.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @ eintippen.

 $[5 \angle 60^\circ \angle 45^\circ]$ $[1.76777 \ 3.06186 \ 3.53553]$

zylindrisch

 $[5 \angle 60^\circ \angle 45^\circ]$ $[3.53553 \ \angle 1.0472 \ 3.53553]$

sphärisch

 $[5 \angle 60^\circ \angle 45^\circ]$ $[5. \angle 1.0472 \angle 0.785398]$ $(Größe \angle Winkel) \Rightarrow komplexer Wert$

(Eingabe polar)

Dient zur Eingabe eines komplexen Werts in polarer ($r\angle\theta$) Form. Der *Winkel* wird gemäß der aktuellen Winkelmoduseinstellung interpretiert.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

 $5+3 \cdot i \cdot \left(10 \angle \frac{\pi}{4}\right)$ $-2.07107 - 4.07107 \cdot i$ $5+3 \cdot i \cdot \left(10 \angle \frac{\pi}{4}\right)$ $-2.07107 - 4.07107 \cdot i$

_ (Unterstrich als leeres Element)

Siehe "Leere (ungültige) Elemente", Seite 235.

$10^()$

Katalog >

 $10^(\text{Wert}1) \Rightarrow \text{Wert}$ $10^{1.5}$

31.6228

 $10^(\text{Liste}1) \Rightarrow \text{Liste}$

Gibt 10 hoch Argument zurück.

Bei einer Liste wird 10 hoch jedem Element von *Liste1* zurückgegeben.

10^(*Quadratmatrix1*)⇒*Quadratmatrix*

Ergibt 10 hoch *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von 10 hoch jedem Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

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

$$\begin{bmatrix} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

^-1(Kehrwert)

Wert1 ^-1⇒*Wert*

$$(3.1)^{-1}$$

0.322581

Liste1 ^-1⇒*Liste*

Gibt den Kehrwert des Arguments zurück.

Bei einer Liste wird für jedes Element von *Liste1* der Kehrwert zurückgegeben.

Quadratmatrix1 ^-1⇒*Quadratmatrix*

Gibt die Inverse von *Quadratmatrix1* zurück.

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

$$\begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

Quadratmatrix1 muss eine nicht-singuläre quadratische Matrix sein.

| (womit-Operator)

Ausdr1 | *BoolescherAusdr1*

$$x+1|x=3$$

4

[**and***BoolescherAusdr2*]...

$$x+55|x=\sin(55)$$

54.0002

Ausdr1 | *BoolescherAusdr1*

[**or***BoolescherAusdr2*]...

Das womit-Symbol („|“) dient als binärer Operator. Der Operand links von | ist ein Ausdruck. Der Operand rechts von | gibt eine oder mehrere Relationen an, die auf die Vereinfachung des Ausdrucks einwirken sollen. Bei Angabe mehrerer Relationen nach dem | sind diese jeweils mit logischen „and“ oder „or“ Operatoren miteinander zu verketten.

Der womit-Operator erfüllt drei

Grundaufgaben:

| (womit-Operator)

ctrl Tasten

- Ersetzung
- Intervallbeschränkung
- Ausschließung

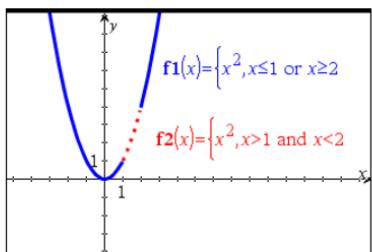
Ersetzungen werden in Form einer Gleichung angegeben, wie etwa $x=3$ oder $y=\sin(x)$. Am wirksamsten ist eine Ersetzung, wenn die linke eine einfache Variable ist. Ausdr | Variable = Wert bewirkt, dass jedes Mal, wenn Variable in Ausdr vorkommt, Wert ersetzt wird.

Intervallbeschränkungen werden in Form einer oder mehrerer mit logischen „and“ oder „or“ Operatoren verknüpfte Ungleichungen angegeben.

Intervallbeschränkungen ermöglichen auch Vereinfachungen, die andernfalls ungültig oder nicht berechenbar wären.

$x^3 - 2 \cdot x + 7 \rightarrow f(x)$	Done
$f(x) x=\sqrt{3}$	8.73205

$\text{nSolve}(x^3 + 2 \cdot x^2 - 15 \cdot x = 0, x)$	0.
$\text{nSolve}(x^3 + 2 \cdot x^2 - 15 \cdot x = 0, x) x > 0 \text{ and } x < 5$	3.



Ausschließungen verwenden den relationalen Operator „ungleich“ ($/=$ oder \neq), um einen bestimmten Wert bei der Operation auszuschließen.

→ (speichern)

ctrl var Taste

Wert → Var

$\frac{\pi}{4} \rightarrow myvar$	0.785398
-----------------------------------	----------

Liste → Var

$2 \cdot \cos(x) \rightarrow y1(x)$	Done
-------------------------------------	------

Matrix → Var

$\{1, 2, 3, 4\} \rightarrow lsl5$	$\{1, 2, 3, 4\}$
-----------------------------------	------------------

Expr → Funktion(Param1,...)

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

List → Funktion(Param1,...)

"Hello" → str1	"Hello"
----------------	---------

Matrix → Funktion(Param1,...)

Wenn Variable Var noch nicht existiert, wird Var erzeugt und auf Wert, Liste oder Matrix initialisiert.

→ (speichern)

ctrl var

Taste

Wenn *Var* existiert und nicht gesperrt oder geschützt ist, wird der Variableninhalt durch *Wert*, *Liste* oder *Matrix* ersetzt.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel `=:` eintippen. Geben Sie zum Beispiel `pi/4 =: myvar` ein.

:= (zuweisen)

ctrl var Tasten

Var := *Wert*

myvar := $\frac{\pi}{4}$.785398

Var := *Liste*

y1(*x*) := 2 · cos(*x*) Done

Var := *Matrix*

lst5 := {1,2,3,4} {1,2,3,4}

Function(*Param1*,...) := *Ausdr*

matg := $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Function(*Param1*,...) := *Liste*

str1 := "Hello" "Hello"

Function(*Param1*,...) := *Matrix*

Wenn Variable *Var* noch nicht existiert, wird *Var* erzeugt und auf *Wert*, *Liste* oder *Matrix* initialisiert.

Wenn *Var* existiert und nicht gesperrt oder geschützt ist, wird der Variableninhalt durch *Wert*, *Liste* bzw. *Matrix* ersetzt.

© (Kommentar)

ctrl book Tasten

© [*Text*]

Define *g*(*n*)=Func

© Declare variables

Local *i,result*

result:=0

For *i*,1,*n*,1 ©Loop *n* times

result:=*result*+*i*²

EndFor

Return *result*

EndFunc

Done

© verarbeitet *Text* als Kommentarzeile und ermöglicht so die Eingabe von Anmerkungen zu von Ihnen erstellten Funktionen und Programmen.

© kann an den Zeilenanfang oder an eine beliebige Stelle der Zeile gesetzt werden. Alles, was rechts von © bis zum Zeilenende steht, gilt als Kommentar.

g(3)

14

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von
mehrzeiligen Programm- und
Funktionsdefinitionen finden Sie im
Abschnitt „Calculator“ des
Produkthandbuchs.

0b, 0h**0b binäre_Zahl****0h hexadezimale_Zahl**

Kennzeichnet eine Dual- bzw.
Hexadezimalzahl. Zur Eingabe einer Dual-
oder Hexadezimalzahl muss unabhängig
vom jeweiligen Basis-Modus das Präfix 0b
bzw. 0h verwendet werden. Eine Zahl ohne
Präfix wird als dezimal behandelt
(Basis 10).

Die Ergebnisse werden im jeweiligen Basis-
Modus angezeigt.

0 [B] Tasten, 0 [H] Tasten

Im Dec-Modus:

0b10+0hF+10

27

Im Bin-Modus:

0b10+0hF+10

0b11011

Im Hex-Modus:

0b10+0hF+10

0h1B

TI-Nspire™ CX II – Zeichenbefehle

Das vorliegende Dokument ergänzt das TI-Nspire™ Referenzhandbuch und das TI-Nspire™ CAS Referenzhandbuch. Alle TI-Nspire™ CX II Befehle werden in Version 5.1 des TI-Nspire™ Referenzhandbuchs und des TI-Nspire™ CAS Referenzhandbuchs ergänzt und mit ihnen veröffentlicht.

Grafikprogrammierung

In den TI-Nspire™ CX II Handhelds und TI-Nspire™ Desktop-Applikationen wurden für die Grafikprogrammierung Befehle hinzugefügt.

Die TI-Nspire™ CX II Handhelds wechseln in diesen Grafikmodus, wenn Grafikbefehle ausgeführt werden und wechseln nach Beendigung des Programms in den Kontext zurück, in dem das Programm ausgeführt wurde.

Auf dem Bildschirm wird bei Ausführung des Programms in der oberen Leiste „Wird ausgeführt...“ angezeigt. Bei Beendigung des Programms wird „Beendet“ angezeigt. Durch Drücken einer beliebigen Taste verlässt das System den Grafikmodus.

- Der Wechsel zum Grafikmodus wird automatisch ausgelöst, wenn bei Ausführung des TI-Basic-Programms einer der Zeichenbefehle (Grafikbefehle) erkannt wird.
- Dieser Wechsel findet nur dann statt, wenn ein Programm in Calculator ausgeführt wird bzw. in Scratchpad in einem Dokument oder Taschenrechner.
- Der Wechsel vom Grafikmodus weg wird bei Programmbeendigung ausgeführt.
- Der Grafikmodus ist nur in der TI-Nspire™ CX II Handheld- und Desktop-TI-Nspire™ CX II Handheld-Ansicht verfügbar. Das bedeutet, dass dieser in der PC-Dokumentenansicht oder im PublishView (.tnsp) weder auf dem Desktop noch in iOS verfügbar ist.
 - Bei Erkennen eines Grafikbefehls während der Ausführung eines TI-Basic-Programms in einem falschen Kontext wird eine Fehlermeldung angezeigt und das TI-Basic-Programm beendet.

Grafikbildschirm

Der Grafikbildschirm enthält oben eine Kopfzeile, in die durch Grafikbefehle nicht geschrieben werden kann.

Der Zeichenbereich des Grafikbildschirms wird bei Initialisierung des Grafikbildschirms entfernt (Farbe = 255,255,255).

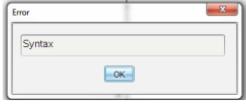
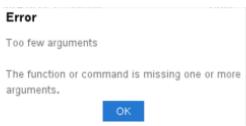
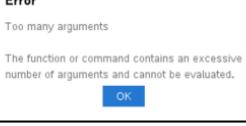
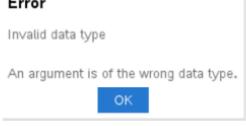
Grafikbildschirm	Standard
Höhe	212
Breite	318
Farbe	Weiß: 255,255,255

Standardansicht und Einstellungen

- Die Statussymbole in der oberen Symbolleiste (Batteriestatus, Press-to-Test-Status, Netzwerkanzeige usw.) sind bei Ausführung eines Grafikprogramms nicht sichtbar.
- Standardzeichenfarbe: Schwarz (0,0,0)
- Standard-Stiftstil – normal, geglättet
 - Dicke: 1 (dünn), 2 (normal), 3 (dick)
 - Stil 1 = (durchgängig), 2 = (gepunktet), 3 = (gestrichelt)
- Alle Zeichenbefehle verwenden die aktuellen Farb- und Stifteinstellungen; entweder Standardwerte oder solche, die über TI-Basic-Befehle eingestellt wurden.
- Die Schriftgröße ist unveränderlich.
- Jede Ausgabe in einem Grafikbildschirm wird in einem Zuschneidefenster gezeichnet, das die Größe des Grafikfenster-Zeichenbereichs hat. Jede Zeichnungsausgabe, die sich über dieses Zuschneide-Grafikfenster hinaus erstreckt, wird nicht gezeichnet. Es wird keine Fehlermeldung angezeigt.
- Alle X-Y-Koordinaten, die für Zeichenbefehle angegeben werden, sind derart definiert, dass sich (0,0) in der oberen linken Ecke des Zeichenbereichs des Grafikbildschirms befindet.
 - **Ausnahmen:**
 - **DrawText** verwendet für den Text die Koordinaten als untere linke Ecke des begrenzenden Rechtecks.
 - **SetWindow** verwendet die untere linke Ecke des Bildschirms.
- Alle Parameter für die Befehle können als Ausdrücke bereitgestellt werden, die eine Zahl ergeben, die dann auf die nächste Ganzzahl aufgerundet wird.

Fehlermeldungen des Grafikbildschirms

Schlägt die Validierung fehl, wird eine Fehlermeldung angezeigt.

Fehlermeldung	Beschreibung	Ansicht
Fehler Syntax	Wenn bei der Syntaxprüfung Syntaxfehler festgestellt werden, wird eine Fehlermeldung angezeigt und versucht, den Cursor nahe dem ersten Fehler zu platzieren, sodass Sie ihn korrigieren können.	
Fehler Zu wenig Argumente	Der Funktion oder dem Befehl fehlen ein oder mehr Argumente	Error Too few arguments The function or command is missing one or more arguments. 
Fehler Zu viele Argumente	Die Funktion oder der Befehl enthält zu viele Argumente und kann nicht ausgewertet werden.	Error Too many arguments The function or command contains an excessive number of arguments and cannot be evaluated. 
Fehler Ungültiger Datentyp	Ein Argument weist einen falschen Datentyp auf.	Error Invalid data type An argument is of the wrong data type. 

Im Grafikmodus ungültige Befehle

Einige Befehle sind unzulässig, sobald das Programm in den Grafikmodus wechselt. Stößt das System im Grafikmodus auf solche Befehle, wird ein Fehler angezeigt und das Programm beendet.

Unzulässiger Befehl	Fehlermeldung
Request	Anfrage kann nicht im Grafikmodus ausgeführt werden
RequestStr	RequestStr kann im Grafikmodus nicht ausgeführt werden
Text	Text kann im Grafikmodus nicht ausgeführt werden

Die Befehle, mit denen Text im Calculator gedruckt wird – **disp** und **dispAt** – sind im Grafikkontext unterstützte Befehle. Der Text dieser Befehle wird an den Calculator-Bildschirm (nicht an den Grafikbildschirm) gesendet und ist nach der Beendigung des Programms sichtbar. Das System wechselt anschließend zurück zur Calculator App.

Löschen (Clear)**Katalog >** 
CXII**Clear $x, y, \text{Breite}, \text{Höhe}$**

Löscht den gesamten Bildschirm, wenn keine Parameter angegeben wurden.

Werden x, y, Breite und Höhe angegeben, wird das durch die Parameter definierte Rechteck gelöscht.

Löschen

Löscht den gesamten Bildschirm

Clear 10,10,100,50

Löscht eine Rechtecksfläche mit der oberen linken Ecke in (10, 10), einer Breite 100 und einer Höhe 50

DrawArc

Katalog > CXII

DrawArc *x, y, Breite, Höhe, startAngle, arcAngle*

Zeichnet einen Bogen innerhalb eines definierten begrenzenden Rechtecks mit dem angegebenen Start- und Bogenwinkel.

x, y: obere linke Koordinate des begrenzenden Rechtecks

Breite, Höhe: Abmessungen des begrenzenden Rechtecks

Der „Bogenwinkel“ definiert die Ausbiegung des Bogens.

Diese Parameter können als Ausdrücke bereitgestellt werden, die eine Zahl ergeben, die dann auf die nächste Ganzzahl gerundet wird.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



Siehe auch: [FillArc](#)

DrawCircle

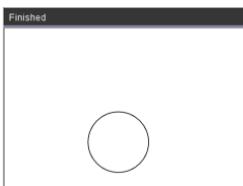
Katalog > CXII

DrawCircle *x, y, Radius*

x, y: Koordinate des Mittelpunkts

Radius: Radius des Kreises

DrawCircle 150,150,40



Siehe auch: [FillCircle](#)

DrawLine

Katalog > CXII

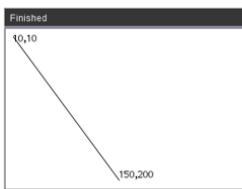
DrawLine $x1, y1, x2, y2$

Zeichnet eine Linie von $x1, y1, x2, y2$ aus.

Ausdrücke, die eine Zahl ergeben, die dann auf die nächste Ganzzahl gerundet wird.

Bildschirmgrenzen: Wenn aufgrund der angegebenen Koordinaten ein Teil der Zeile außerhalb des Grafikbildschirms gezeichnet wird, dann wird dieser Teil der Linie abgeschnitten und keine Fehlermeldung angezeigt.

DrawLine 10,10,150,200



DrawPoly

Katalog > CXII

Es gibt zwei Varianten der Befehle:

DrawPoly $xlist, ylist$

oder

DrawPoly $x1, y1, x2, y2, x3, y3...xn, yn$

Hinweis: DrawPoly $xlist, ylist$

Form (Shape) verbindet $x1, y1$ mit $x2, y2$,
 $x2, y2$ mit $x3, y3$ usw.

Hinweis: DrawPoly $x1, y1, x2, y2, x3,$

$y3...xn, yn$

xn, yn wird **NICHT** automatisch mit $x1, y1$ verbunden.

Ausdrücke, die eine Liste von realen Float-Variablen ergeben

$xlist, ylist$

Ausdrücke, die eine reale einzelne Float-Variable ergeben

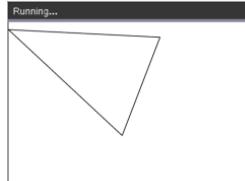
$x1, y1...xn, yn$ = Koordinaten für

Polygoneckpunkte

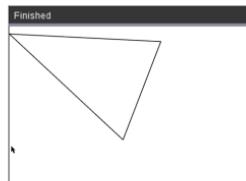
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

DrawPoly xlist,ylist



DrawPoly 0,10,200,20,150,150,0,10



Hinweis: DrawPoly:

Eingabegrößenabmessungen (Breite/Höhe) relativ zu gezeichneten Linien.
Die Zeilen werden in einem begrenzenden Rechteck um die angegebene Koordinate gezeichnet, und Abmessungen wie beispielsweise die tatsächliche Größe des gezeichneten Polygons sind größer als die Breite und Höhe.

Siehe auch: [FillPoly](#)

DrawRect

DrawRect *x, y, Breite, Höhe*

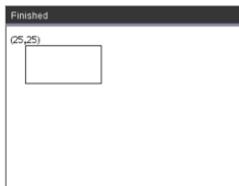
x, y: Obere linke Koordinate des Rechtecks

Breite, Höhe: Breite und Höhe des Rechtecks. (Das Rechteck wird von der Startkoordinate ausgehend nach unten und nach rechts gezeichnet.)

Hinweis: Die Zeilen werden in einem begrenzenden Rechteck um die angegebene Koordinate gezeichnet, und Abmessungen wie beispielsweise die tatsächliche Größe des gezeichneten Rechtecks sind größer als die angezeigte Breite und Höhe.

Siehe auch: [FillRect](#)

DrawRect 25,25,100,50

**DrawText**

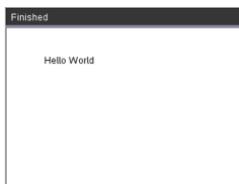
DrawText *x, y, exprOrString1
[,exprOrString2]...*

x, y: Koordinaten der Textausgabe

Zeichnet den Text in *exprOrString* an der angegebenen *x--y*-Koordinatenposition.

Die Regeln für *exprOrString* sind die gleichen wie für **Disp – DrawText** kann mehrere Argumente akzeptieren.

DrawText 50,50,"Hallo Welt"



FillArc

Katalog > CXII

FillArc *x, y, Breite, Höhe startAngle, arcAngle*

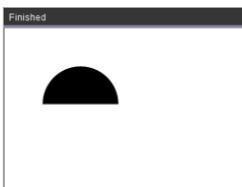
x, y: obere linke Koordinate des begrenzenden Rechtecks

Innerhalb des definierten begrenzenden Rechtecks mit den angegebenen Start- und Bogenwinkeln einen Bogen zeichnen und füllen.

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem [SetColor](#)-Befehl eingestellt werden.

Der „Bogenwinkel“ definiert die Ausbiegung des Bogens.

FillArc 50,50,100,100,0,180

**FillCircle**

Katalog > CXII

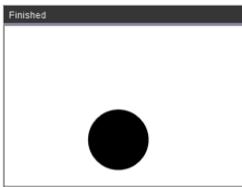
FillCircle *x, y, Radius*

x, y: Koordinate des Mittelpunkts

Einen Kreis mit angegebenen Mittelpunkt und Radius zeichnen und füllen.

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem [SetColor](#)-Befehl eingestellt werden.

FillCircle 150,150,40



Hier!

FillPoly

Katalog > CXII

FillPoly *xlist, ylist*

oder

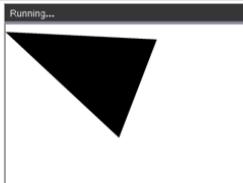
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

Hinweis: Linie und Farbe werden durch [SetColor](#) und [SetPen](#) festgelegt.

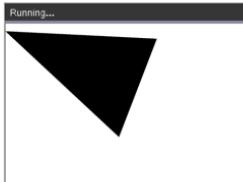
xlist:={0,200,150,0}

ylist:={10,20,150,10}

FillPoly xlist,ylist



```
FillPoly 0,10,200,20,150,150,0,10
```



FillRect

FillRect *x, y, Breite, Höhe*

x, y: Obere linke Koordinate des Rechtecks

Breite, Höhe: Breite und Höhe des Rechtecks

An der durch (*x,y*) angegebenen Koordinate mit der oberen linken Ecke ein Rechteck zeichnen und füllen

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem [SetColor](#)-Befehl eingestellt werden.

Hinweis: Linie und Farbe werden durch [SetColor](#) und [SetPen](#) festgelegt.

```
FillRect 25,25,100,50
```



getPlatform()**Katalog > CXII****getPlatform()**

getPlatform()

"dt"

Ergibt:

„dt“ auf Desktop-Softwareanwendungen

„hh“ auf TI-Nspire™ CX Handhelds

„ios“ auf TI-Nspire™ CX App für iPad®

PaintBuffer**PaintBuffer**

Farbengrafik-Puffer zum Bildschirm

Dieser Befehl wird in Verbindung mit UseBuffer verwendet, um die Geschwindigkeit der Darstellung auf dem Bildschirm zu erhöhen, wenn das Programm mehrere Grafikobjekte erzeugt.

UseBuffer

```
For n,1,10  
x:=randInt(0,300)  
y:=randInt(0,200)  
Radius:=randInt(10,50)  
Wait 0,5  
DrawCircle x,y,Radius  
EndFor  
PaintBuffer  
Dieses Programm zeigt alle  
10 Kreise gleichzeitig an.  
Wird der Befehl „UseBuffer“  
entfernt, wird jeder Kreis so  
angezeigt, wie er gezeichnet wird.
```

Siehe auch: [UseBuffer](#)

PlotXY $x, y, Form$ x, y : Koordinate zur Plot-Form*Form*: eine Zahl zwischen 1 und 13, die die Form festlegt

1 – Gefüllter Kreis

2 – Leerer Kreis

3 – Gefülltes Quadrat

4 – Leeres Quadrat

5 – Kreuz

6 – Plus

7 – Dünn

8 – Mittelgroßer Punkt, ausgefüllt

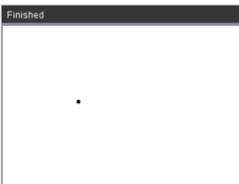
9 – Mittelgroßer Punkt, unausgefüllt

10 – Großer Punkt, ausgefüllt

11 – Großer Punkt, unausgefüllt

12 – Größter Punkt, ausgefüllt

13 – Größter Punkt, unausgefüllt

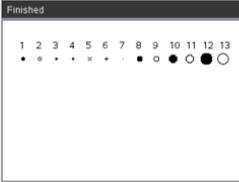
PlotXY 100,100,1

For n,1,13

DrawText 1+22*n,40,n

PlotXY 5+22*n,50,n

EndFor



SetColor

Katalog > CXII

SetColor

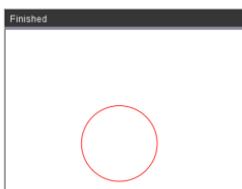
Rot-Wert, Grün-Wert, Blau-Wert

Gültige Werte für Rot, Grün und Blau liegen zwischen 0 und 255.

Legt die Farbe für nachfolgende Draw-Befehle fest

SetColor 255,0,0

DrawCircle 150,150,100

**SetPen**

Katalog > CXII

SetPen

Dicke, Stil

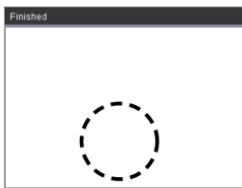
Dicke: $1 \leq \text{Dicke} \leq 3 | 1$ ist am dünnsten, 3 ist am dicksten

Stil: 1 = Durchgängig, 2 = Gepunktet, 3 = Gestrichelt

Richtet den Stiftstil für nachfolgende Zeichenbefehle ein

SetPen 3,3

DrawCircle 150,150,50

**SetWindow**

Katalog > CXII

SetWindow

xMin, xMax, yMin, yMax

Richtet ein logisches Fenster ein, das dem Grafikzeichnenbereich zugeordnet ist. Alle Parameter sind erforderlich.

Befindet sich der Teil des gezeichneten Objekts außerhalb des Fensters, wird die Ausgabe zugeschnitten (nicht angezeigt) und keine Fehlermeldung angezeigt.

SetWindow 0,160,0,120

Stellt das Ausgabefenster wie folgt ein: (0,0) in der linken unteren Ecke mit einer Breite von 160 und einer Höhe von 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

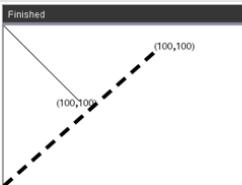
DrawLine 0,0,100,100

Ist xmin größer oder gleich xmax oder ymin größer oder gleich ymax, wird eine Fehlermeldung angezeigt.

Objekte, die vor einem SetWindow-Befehl gezeichnet wurden, werden mit der neuen Konfiguration nicht neu gezeichnet.

Verwenden Sie zum Zurücksetzen der Fensterparameter auf die Standardeinstellungen:

SetWindow 0,0,0,0



UseBuffer**UseBuffer**

Zeichnet Grafik-Buffer außerhalb des Bildschirms anstatt auf den Bildschirm (zur Leistungssteigerung)

Dieser Befehl wird in Verbindung mit PaintBuffer verwendet, um die Geschwindigkeit der Darstellung auf dem Bildschirm zu erhöhen, wenn das Programm mehrere Grafikobjekte erzeugt.

Mit UseBuffer werden alle Grafiken erst nach Ausführung des nächsten PaintBuffer-Befehls angezeigt.

UseBuffer muss lediglich einmal im Programm aufgerufen werden, d. h. nicht bei jeder Verwendung von PaintBuffer ist ein entsprechender UseBuffer erforderlich.

UseBuffer

```
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
```

```
Radius:=randInt(10,50)
```

```
Wait 0,5
```

```
DrawCircle x,y,Radius
```

```
EndFor
```

```
PaintBuffer
```

Dieses Programm zeigt alle 10 Kreise gleichzeitig an.

Wird der Befehl „UseBuffer“ entfernt, wird jeder Kreis so angezeigt, wie er gezeichnet wird.

Siehe auch: [PaintBuffer](#)

Leere (ungültige) Elemente

Bei der Analyse von Daten der realen Welt liegt möglicherweise nicht immer ein vollständiger Datensatz vor. TI-Nspire™ lässt leere bzw. ungültige Datenelemente zu, sodass Sie mit den nahezu vollständigen Daten fortfahren können anstatt von vorn anfangen oder unvollständige Fälle verwerfen zu müssen.

Ein Beispiel für Daten mit leeren Elementen finden Sie im Kapitel Lists & Spreadsheet unter „*Tabellendaten grafisch darstellen*“.

Mit der Funktion **delVoid()** können Sie leere Elemente aus einer Liste löschen. Die Funktion **isVoid()** sucht nach leeren Elementen. Einzelheiten finden Sie unter **delVoid()**, Seite 43, und **isVoid()**, Seite 83.

Hinweis: Um ein leeres Element manuell in einen mathematischen Ausdruck einzugeben, geben Sie „_“ oder das Schlüsselwort **void** ein. Das Schlüsselwort **void** wird bei der Auswertung des Ausdrucks automatisch in das Symbol „_“ konvertiert. Um „_“ auf dem Handheld einzugeben, drücken Sie **ctrl** **enter**.

Kalkulationen mit ungültigen Elementen

Bei der Mehrzahl aller Kalkulationen, die ein ungültiges Element enthalten, wird das Ergebnis ebenfalls ungültig sein.

Sonderfälle sind nachstehend aufgeführt.

$\lfloor _\rfloor$	-
$\text{gcd}\{100, _\}$	-
$3 + __$	-
$\{5, __, 10\} - \{3, 6, 9\}$	$\{2, __, 1\}$

Listenargumente, die ungültige Elemente enthalten

Die folgenden Funktionen und Befehle ignorieren (überspringen) ungültige Elemente, die in Listenargumenten gefunden werden.

count, **countIf**, **cumulativeSum**, **freqTable**►list, **frequency**, **max**, **mean**, **median**, **product**, **stDevPop**, **stDevSamp**, **sum**, **sumIf**, **varPop** und **varSamp** sowie Regressionskalkulationen, **OneVar**, **TwoVar** und **FiveNumSummary** Statistiken, Konfidenzintervalle und statistische Tests

$\text{sum}\{\{2, __, 3, 5, 6, 6\}\}$	16.6
$\text{median}\{\{1, 2, __, __, 3\}\}$	2
$\text{cumulativeSum}\{\{1, 2, __, 4, 5\}\}$	$\{1, 3, __, 7, 12\}$
$\text{cumulativeSum}\begin{bmatrix} 1 & 2 \\ 3 & _ \\ 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 4 & _ \\ 9 & 8 \end{bmatrix}$

Listenargumente, die ungültige Elemente enthalten

SortA und **SortD** verschieben alle ungültigen Elemente im ersten Argument nach unten.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
SortA $list1, list2$	<i>Done</i>
$list1$	$\{1,3,4,5,_\}$
$list2$	$\{1,3,4,5,2\}$

In Regressionen sorgt ein ungültiges Element in einer Liste X oder Y dafür, dass auch das entsprechende Element im Residuum ungültig ist.

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD $list1, list2$	<i>Done</i>
$list1$	$\{5,3,2,1,_\}$
$list2$	$\{5,3,2,1,4\}$

$l1:=\{1,2,3,4,5\}; l2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx $l1, l2$	<i>Done</i>
<i>stat.Resid</i>	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
<i>stat.XReg</i>	$\{1,_, 3, 4, 5, _\}$
<i>stat.YReg</i>	$\{2,_, 3, 5, 6, 6\}$
<i>stat.FreqReg</i>	$\{1,_, 1, 1, 1, _\}$

Eine ausgelassene Kategorie in Regressionen sorgt dafür, dass das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
<i>cat</i> := { "M", "M", "F", "F" }; <i>incl</i> := { "F" }	{ "F" }
LinRegMx $l1, l2, 1, cat, incl$	<i>Done</i>
<i>stat.Resid</i>	$\{_, _, 0, 0, _\}$
<i>stat.XReg</i>	$\{_, _, 4, 5, _\}$
<i>stat.YReg</i>	$\{_, _, 5, 6, 6\}$
<i>stat.FreqReg</i>	$\{_, _, 1, 1, _\}$

Eine Häufigkeit von 0 in Regressionen führt dazu, dass das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx $l1, l2, \{1, 0, 1, 1\}$	<i>Done</i>
<i>stat.Resid</i>	$\{0.069231,_, -0.276923, 0.207692\}$
<i>stat.XReg</i>	$\{1,_, 4, 5, _\}$
<i>stat.YReg</i>	$\{2,_, 5, 6, 6\}$
<i>stat.FreqReg</i>	$\{1,_, 1, 1, _\}$

Tastenkürzel zum Eingeben mathematischer Ausdrücke

Tastenkürzel ermöglichen es Ihnen, Elemente mathematischer Ausdrücke über die Tastatur einzugeben anstatt über den Katalog oder die Sonderzeichenpalette. Um beispielsweise den Ausdruck $\sqrt{6}$ einzugeben, können Sie `sqrt(6)` in die Eingabezeile eingeben. Wenn Sie **enter** drücken, ändert sich der Ausdruck `sqrt(6)` in $\sqrt{6}$. Einige Tastenkürzel sind sowohl für die Eingabe über das Handheld als auch über die Computertastatur nützlich. Andere sind hauptsächlich für die Computertastatur hilfreich.

Von Handheld oder Computertastatur

Sonderzeichen:	Tastenkürzel:
π	<code>pi</code>
θ	<code>theta</code>
∞	<code>infinity</code>
\leq	<code><=</code>
\geq	<code>>=</code>
\neq	<code>/=</code>
\Rightarrow (logische Implikation)	<code>=></code>
\Leftrightarrow (logische doppelte Implikation, XNOR)	<code><=></code>
\rightarrow (Operator speichern)	<code>=:</code>
$ $ (Absolutwert)	<code>abs(...)</code>
$\sqrt()$	<code>sqrt(...)</code>
$\Sigma()$ (Vorlage Summe)	<code>sumSeq(...)</code>
$\Pi()$ (Vorlage Produkt)	<code>prodSeq(...)</code>
$\sin^{-1}(), \cos^{-1}(), \dots$	<code>arcsin(...), arccos(...), ...</code>
Δ Liste()	<code>deltaList(...)</code>

Von der Computertastatur

Sonderzeichen:	Tastenkürzel:
i (imaginäre Konstante)	<code>@i</code>
e (natürlicher Logarithmus zur Basis e)	<code>@e</code>
E (wissenschaftliche Schreibweise)	<code>@E</code>
T (Transponierte)	<code>@t</code>

Sonderzeichen:	Tastenkürzel:
r (Bogenmaß)	@r
\circ (Grad)	@d
g (Neugrad)	@g
\angle (Winkel)	@<
\blacktriangleright (Umwandlung)	@>
►Decimal, ►approxFraction() usw.	@>Decimal, @>approxFraction() usw.

Auswertungsreihenfolge in EOS™ (Equation Operating System)

Dieser Abschnitt beschreibt das Equation Operating System (EOS™), das von der TI-Nspire™ Technologie genutzt wird. Zahlen, Variablen und Funktionen werden in einer einfachen Abfolge eingegeben. Die EOS™ Software wertet Ausdrücke und Gleichungen anhand der gesetzten Klammern und der im Folgenden beschriebenen Priorität der Operatoren aus.

Auswertungsreihenfolge

Ebene	Operator
1	Klammern: rund (), eckig [], geschweift { }
2	Umleitung (#)
3	Funktionsaufrufe
4	Postfix-Operatoren: Grad-Minuten-Sekunden ($^{\circ}, ", !$), Fakultät (!), Prozent (%), Bogenmaß (Γ), Tiefstellen ([]), Transponieren (T)
5	Potenzieren, Potenzoperator (^)
6	Negation (-)
7	Stringverkettung (&)
8	Multiplikation (\cdot), Division (/)
9	Addition (+), Subtraktion (-)
10	Gleichheitsbeziehungen: gleich (=), ungleich (\neq oder $/=$), kleiner als (<), kleiner oder gleich (\leq oder $<=$), größer als (>), größer oder gleich (\geq oder $>=$)
11	Logisches Nicht: not
12	Logische Konjunktion: and
13	Logisch or
14	xor, nor, nand
15	logische Implikation, (\Rightarrow)
16	Logische doppelte Implikation, XNOR (\Leftrightarrow)
17	womit-Operator („ “)
18	Speichern (\rightarrow)

Klammern (rund, eckig, geschweift)

Alle Berechnungen, die in Klammern – runde, eckige oder geschweifte – gesetzt sind, werden als erste ausgewertet. Ein Beispiel: Im Ausdruck $4(1+2)$ wertet die EOS™ Software zunächst $1+2$ aus, da dieser Teil des Ausdrucks in Klammern steht. Das Ergebnis 3 wird dann mit 4 multipliziert.

Die Anzahl der öffnenden und schließenden Klammern eines jeden Typs muss innerhalb eines Ausdrucks oder einer Gleichung jeweils übereinstimmen. Andernfalls wird eine Fehlermeldung mit dem fehlenden Element angezeigt. Beim Ausdruck $(1+2)/(3+4$ erscheint beispielsweise die Fehlermeldung „) fehlt“.

Hinweis: In der TI-Nspire™ Software können Sie Ihre eigenen Funktionen definieren. Daher wird eine Variable, auf die ein Ausdruck in Klammern folgt, als Funktionsaufruf und nicht wie sonst implizit als Multiplikation interpretiert. Der Ausdruck $a(b+c)$ steht beispielsweise für den Wert der Funktion a mit dem Argument $b+c$. Um den Ausdruck $b+c$ mit der Variablen a zu multiplizieren, verwenden Sie die explizite Multiplikation: $a*(b+c)$.

Umleitung

Der Umleitungsoperator # wandelt eine Zeichenfolge (String) in einen Variablen- oder Funktionsnamen um. Mit #("x"&"y"&"z") wird beispielsweise der Variablenname xyz erstellt. Mithilfe der Umleitung können Sie auch Variablen aus einem Programm heraus erstellen und modifizieren. Beispiel: Wenn $10 \rightarrow r$ und $"r" \rightarrow s1$, dann $#s1=10$.

Postfix-Operatoren

Postfix-Operatoren sind Operatoren, die direkt nach einem Argument stehen, zum Beispiel 5!, 25% oder $60^{\circ}15'45''$. Argumente, auf die ein Postfix-Operator folgt, werden auf der vierten Prioritätsebene ausgewertet. Beispiel: Im Ausdruck $4^3!$ wird zuerst $3!$ ausgewertet. Das Ergebnis 6 wird dann als Exponent für 4 verwendet, und das Endergebnis ist 4096.

Potenz

Potenzen (^) und elementweise Potenzen (.^) werden von rechts nach links ausgewertet. Der Ausdruck 2^3^2 wird zum Beispiel wie $2^(3^2)$ ausgewertet, hat also das Ergebnis 512. Er unterscheidet sich damit vom Ausdruck $(2^3)^2$ mit dem Ergebnis 64.

Negation

Zum Eingeben einer negativen Zahl drücken Sie [(-)] und geben dann die Zahl ein. Postfix-Operatoren und Potenzen werden vor der Negation ausgewertet. Das Ergebnis von $-x^2$ ist zum Beispiel eine negative Zahl; $-9^2 = -81$. Um eine negative Zahl zu quadrieren, verwenden Sie Klammern: $(-9)^2$, Ergebnis 81.

Einschränkung („|“)

Das Argument nach dem womit-Operator „|“ stellt eine Reihe von Einschränkungen dar, die beeinflussen, wie das Argument vor dem Operator ausgewertet wird.

TI-Nspire CX II – TI-Basic Programmierfunktionen

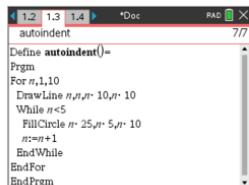
Automatisches Einrücken im Programmierungseditor

Der TI-Nspire™ Programmeditor rückt Anweisungen nun automatisch in einem Blockbefehl ein.

Blockbefehle sind If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry.

Der Editor stellt in einem Blockbefehl Programmbefehl automatisch Leerstellen voran. Der Schließbefehl des Blocks wird am Öffnungsbefehl ausgerichtet.

Das unten stehende Beispiel zeigt das automatische Einrücken in Befehlen mit verschachtelten Blöcken.



The screenshot shows the TI-Nspire CX II TI-Basic Programming Editor interface. The code window contains the following TI-Basic program:

```
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
```

Bei Codefragmenten, die kopiert und eingefügt werden, wird deren ursprüngliche Einrückung beibehalten.

Wird ein in einer früheren Version der Software erstelltes Programm geöffnet, wird die ursprüngliche Einrückung beibehalten.

Verbesserte Fehlermeldungen für TI-Basic

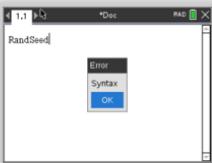
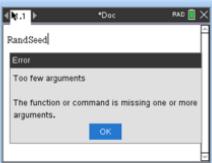
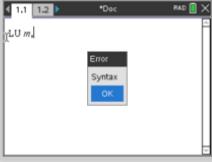
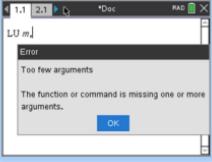
Fehler

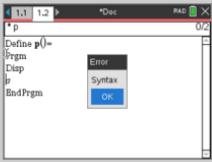
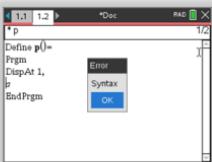
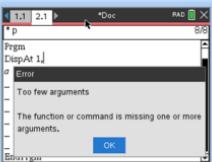
Fehlermeldungen	Neue Meldung
Fehler in der Bedingungsanweisung (If/While)	Eine bedingte Anweisung hat RICHTIG oder FALSCH nicht aufgeklärt. HINWEIS: Durch die Platzierung des Cursors auf die Linie mit dem Fehler muss nicht mehr angegeben werden, ob der Fehler ein „If“-Ausdruck oder ein „While“-Ausdruck ist.
EndIf fehlt	Erwartete EndIf , fand aber eine andere End-Anweisung
EndFor fehlt	Erwartete EndFor , fand aber eine andere End-Anweisung
EndWhile fehlt	Erwartete EndWhile , fand aber eine andere End-Anweisung

Fehlermeldungen	Neue Meldung
EndLoop fehlt	Erwartete EndLoop , fand aber eine andere End-Anweisung
EndTry fehlt	Erwartete EndTry , fand aber eine andere End-Anweisung
„ Then “ nach If <condition> nicht angegeben	If..Then fehlt
„ Then “ nach ElseIf <condition> nicht angegeben	Then fehlt in Block: Elseif .
Wenn „ Then “, „ Else “ und „ Elseif “ außerhalb der Steuerblöcke gefunden wurden	Else ungültig außerhalb der Blöcke: If..Then..Endif oder Try..Endtry
„ Elseif “ erscheint außerhalb des „ If..Then..Endif -Blocks	Elseif ungültig außerhalb des Blocks: If..Then..Endif
„ Then “ erscheint außerhalb des „ If....Endif -Blocks	Then ungültig außerhalb des Blocks: If..Endif

Syntaxfehler

Wenn Befehle, die ein oder mehrere Argumente erfordern, mit einer unvollständigen Argumentenliste aufgerufen werden, wird anstelle eines „**Syntax**“-Fehlers ein „**Zu wenige Argumente**“-Fehler ausgegeben.

Aktuelles Verhalten	Neues CX II-Verhalten
	
	

Aktuelles Verhalten	Neues CX II-Verhalten
	
	

Hinweis: Wenn auf eine unvollständige Argumentenliste kein Komma folgt, lautet die Fehlermeldung: „zu wenig Argumente“. Bei früheren Versionen war das genauso.



Konstanten und Werte

Die folgende Tabelle führt die Konstanten und ihre Werte auf, die verfügbar sind, wenn eine Einheitenumrechnung durchgeführt wird. Diese können manuell eingegeben werden oder aus der Liste der **Konstanten in Hilfsfunktionen > Einheitenumrechnungen** ausgewählt werden (Beim Handheld: Drücken Sie  3).

Konstante	Name	Wert
c	Lichtgeschwindigkeit	299792458 m/s
C_C	Coulombsche Konstante	8987551787,3682 m/F
F_C	Faraday-Konstante	96485,33289 coul/mol
g	Erdbeschleunigung	9,80665 m/s^2
G_C	Gravitationskonstante	6,67408E-11 $\text{m}^3/\text{kg}/\text{s}^2$
h	Plancksche Konstante	6,626070040E-34 $\text{J}\cdot\text{s}$
k	Boltzmann-Konstante	1,38064852E-23 J/K
μ_0	Permeabilität des Vakuums	1,2566370614359E-6 N/A^2
μ_B	Bohr-Magneton	9,274009994E-24 $\text{J}\cdot\text{m}^2/\text{Wb}$
M_e	Ruhemasse des Elektrons	9,10938356E-31 kg
M_μ	Myonmasse	1,883531594E-28 kg
M_n	Ruhemasse des Neutrons	1,674927471E-27 kg
M_p	Ruhemasse des Protons	1,672621898E-27 kg
N_A	Avogadro-Zahl	6,022140857E23 $/\text{mol}$
q	Elektronenladung	1,6021766208E-19 coul
R_b	Bohr-Radius	5,2917721067E-11 m
R_c	Molare Gaskonstante	8,3144598 $\text{J}/\text{mol}/\text{K}$
R_dB	Rydberg-Konstante	10973731,568508/ m
R_e	Elektronenradius	2,8179403227E-15 m
u	Atommasse	1,660539040E-27 kg
V_m	Molvolumen	2,2413962E-2 m^3/mol
ϵ_0	Permittivität des Vakuums	8,8541878176204E-12 F/m
σ	Stefan-Boltzmann-Konstante	5,670367E-8 $\text{W}/\text{m}^2/\text{K}^4$
Φ_0	Magnetisches Flussquantum	2,067833831E-15 Wb

Fehlercodes und -meldungen

Wenn ein Fehler auftritt, wird sein Code der Variablen *errCode* zugewiesen. Benutzerdefinierte Programme und Funktionen können *errCode* auswerten, um die Ursache eines Fehlers zu bestimmen. Ein Beispiel für die Benutzung von *errCode* finden Sie als Beispiel 2 unter dem Befehl **Versuche (Try)** (Seite 175).

Hinweis: Einigen Fehlerbedingungen gelten nur für TI-Nspire™ CAS Produkte, andere gelten nur für TI-Nspire™ Produkte.

Fehlercode	Beschreibung
10	Funktion ergab keinen Wert
20	Test ergab nicht WAHR oder FALSCH. Generell können nicht definierte Variablen nicht verglichen werden. Beispielsweise würde der Test 'if a<b' diesen Fehler auslösen, wenn entweder a oder b zum Zeitpunkt der Ausführung der If-Anweisung nicht definiert ist.
30	Argument darf kein Verzeichnisname sein.
40	Argumentfehler
50	Argumente passen nicht Zwei oder mehr Argumente müssen vom gleichen Typ sein.
60	Argument muss Boolescher Ausdruck oder ganze Zahl sein
70	Argument muss Dezimalzahl sein
90	Argument muss Liste sein
100	Argument muss Matrix sein
130	Argument muss String sein
140	Argument muss Variablenname sein. Vergewissern Sie sich, dass der Name: <ul style="list-style-type: none">• nicht mit einer Ziffer beginnt• keine Leerzeichen oder Sonderzeichen enthält• keine unzulässigen Unterstriche oder Punkte enthält• die maximale Zeichenlänge nicht überschreitet Weitere Einzelheiten finden Sie im Abschnitt Calculator in der Dokumentation.
160	Argument muss Ausdruck sein
165	Batteriespannung zu niedrig zum Senden/Empfangen Setzten Sie vor dem Senden oder Empfangen neue Batterien ein.
170	Grenze

Fehlercode	Beschreibung
	Um das Suchintervall zu definieren, muss die untere Grenze kleiner sein als die obere Grenze.
180	Abbruch Die Taste esc oder on wurde gedrückt, während eine lange Berechnung oder ein Programm ausgeführt wurde.
190	Zirkuläre Definition Diese Meldung wird angezeigt, um zu verhindern, dass durch unendliches Ersetzen von Variablenwerten bei der Vereinfachung der Platz im Hauptspeicher nicht ausreicht. Dieser Fehler wird beispielsweise durch 'a+1->a' ausgelöst, wenn a eine nicht definierte Variable ist.
200	Zusammengesetzter Ausdruck ungültig Diese Fehlermeldung würde zum Beispiel durch 'solve(3x^2-4=0,x) x<0 or x>5' ausgelöst werden, weil die Einschränkung durch "oder (or)" anstatt "und (and)" getrennt wird.
210	Ungültiger Datentyp Ein Argument weist einen falschen Datentyp auf.
220	Abhängiger Grenzwert
230	Dimension Ein Listen- oder Matrixindex ist ungültig. Wenn beispielsweise die Liste {1,2,3,4} in L1 gespeichert wird, ist L1[5] ein Dimensionsfehler, weil L1 nur vier Elemente enthält.
235	Dimensionsfehler. Nicht genügend Elemente in den Listen.
240	Dimensionsfehler Zwei oder mehr Argumente müssen die gleiche Dimension haben. So ist beispielsweise [1,2]+[1,2,3] ein Dimensionsfehler, weil die Matrizen eine unterschiedliche Anzahl von Elementen enthalten.
250	Division durch Null
260	Bereichsfehler Ein Argument muss in einem festgelegten Bereich sein. rand(0) ist zum Beispiel nicht gültig.
270	Variablename doppelt vergeben
280	Else und Elself außerhalb If..EndIf-Block ungültig
290	Zu EndTry fehlt passende Else-Anweisung
295	Zu viele Iterationen

Fehlercode	Beschreibung
300	2- oder 3-elementige Liste bzw. Matrix erwartet
310	Das erste Argument von nSolve muss eine Gleichung in einer einzigen Variablen sein. Es darf keine andere Variable ohne Wert außer der interessierenden Variablen enthalten.
320	1. Argument von Löse oder cLöse muss Gleichung/Ungleichung sein Löse($3x-4, x$) ist beispielsweise ungültig, weil das erste Argument keine Gleichung ist.
345	Einheiten passen nicht zusammen
350	Index nicht im gültigen Bereich
360	Umleitungs-String kein gültiger Variablenname
380	Undefinierte Antw Entweder hat die vorangegangene Berechnung keine Antw (Ans) erzeugt oder es fand keine vorangegangene Berechnung statt.
390	Zuweisung ungültig
400	Zuweisungswert ungültig
410	Befehl ungültig
430	Ungültig für aktuelle Modus-Einstellungen
435	Schätzwert ungültig
440	Implizierte Multiplikation ungültig Beispielsweise ist ' $x(x+1)$ ' ungültig, während ' $x*(x+1)$ ' eine korrekte Syntax ist. So wird eine Verwechslung zwischen impliziter Multiplikation und Funktionsaufrufen vermieden.
450	In Funktion oder aktuellem Ausdruck ungültig In einer benutzerdefinierten Funktion sind nur bestimmte Befehle zulässig.
490	In Try..EndTry Block ungültig
510	Liste oder Matrix ungültig
550	Ungültig außerhalb Funktion oder Programm Einige Befehle sind nur in einer Funktion oder einem Programm gültig. Beispielsweise kann Lokal (Local) nur in einer Funktion oder einem Programm verwendet werden.
560	Nur in Loop..EndLoop-, For..EndFor- oder While..EndWhile-Block gültig Beispielsweise ist der Befehl Abbruch (Exit) nur in diesen Schleifenblöcken gültig.
565	Nur in einem Programm gültig

Fehlercode	Beschreibung
570	Ungültiger Pfadname \\var ist beispielsweise ungültig.
575	Polarkomplex ungültig
580	Programmaufruf ungültig Programme können nicht innerhalb von Funktionen oder Ausdrücken wie z.B. '1+p(x)' aufgerufen werden, wenn p ein Programm ist.
600	Tabelle ungültig
605	Verwendung der Einheiten ungültig
610	Variablenname in Lokal-Anweisung ungültig
620	Variablen- bzw. Funktionsname ungültig
630	Variablenverweis ungültig
640	Vektorsyntax ungültig
650	Kabelübertragung gestört Eine Übertragung zwischen zwei Geräten wurde nicht abgeschlossen. Überprüfen Sie, dass das Kabel an beiden Seiten fest angeschlossen ist.
665	Diagonalisierung der Matrix nicht möglich
670	Wenig Speicher 1. Löschen Sie Daten in diesem Dokument 2. Speichern und schließen Sie dieses Dokument Wenn 1 und 2 fehlschlagen, nehmen Sie die Batterien heraus und setzen Sie sie wieder ein
672	Ressourcenauslastung
673	Ressourcenauslastung
680	fehlt (
690	fehlt)
700	fehlt "
710	fehlt]
720	fehlt }
730	Anfang oder Ende des Blocks fehlt
740	Then im If..EndIf-Block fehlt

Fehlercode	Beschreibung
750	Name verweist nicht auf Funktion oder Programm
765	Keine Funktionen ausgewählt
780	Keine Lösung gefunden
800	Nicht-reelles Ergebnis Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{(-1)}$ ungültig. Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).
830	Überlauf
850	Programm nicht gefunden Ein Programmverweis in einem anderen Programm wurde während der Ausführung im angegebenen Pfad nicht gefunden.
855	Zufallsfunktionen sind im Graphikmodus nicht zulässig
860	Rekursion zu tief
870	Reservierter Name oder Systemvariable
900	Argumentfehler Das Median-Median-Modell konnte nicht auf den Datensatz angewendet werden.
910	Syntaxfehler
920	Text nicht gefunden
930	Zu wenig Argumente Der Funktion oder dem Befehl fehlen ein oder mehr Argumente.
940	Zu viele Argumente Der Ausdruck oder die Gleichung enthält eine überschüssige Anzahl von Argumenten und kann nicht ausgewertet werden.
950	Zu viele Indizierungen
955	Zu viele undefinierte Variable
960	Variable ist nicht definiert Der Variablen wurde kein Wert zugewiesen. Verwenden Sie einen der folgenden Befehle: <ul style="list-style-type: none">• sto →• :=• Definiere

Fehlercode	Beschreibung
	um Variablen Werte zuzuweisen.
965	Betriebssystem nicht lizenziert
970	Variable ist aktiv, daher keine Verweise oder Änderungen zulässig
980	Variable ist geschützt
990	Ungültiger Variablenname Stellen Sie sicher, dass der Name die maximale Zeichenlänge nicht überschreitet
1000	Fenstervariable nicht im Bereich
1010	Zoom
1020	Interner Fehler
1030	Verletzung des Zugriffsschutzes auf geschützten Speicher
1040	Nicht unterstützte Funktion. Für diese Funktion ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1045	Nicht unterstützter Operator. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1050	Nicht unterstütztes Merkmal. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1060	Das Eingabeargument muss numerisch sein. Nur Eingaben, die numerische Werte enthalten, sind zulässig.
1070	Argument der trig. Funktion ist zu groß für eine exakte Vereinfachung
1080	Keine Unterstützung von Antw (Ans). Diese Applikation unterstützt nicht Antw (Ans).
1090	Funktion ist nicht definiert. Verwenden Sie einen der folgenden Befehle: <ul style="list-style-type: none">• Definiere• :=• sto → um eine Funktion zu definieren.
1100	Nicht-reelle Berechnung Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{(-1)}$ ungültig. Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).
1110	Ungültige Grenzen
1120	Keine Zeichenänderung

Fehlercode	Beschreibung
1130	Argument kann weder eine Liste noch eine Matrix sein
1140	Argumentfehler Das erste Argument muss ein Polynomausdruck im zweiten Argument sein. Wenn das zweite Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1150	Argumentfehler Die ersten zwei Argumente müssen Polynomausdrücke im dritten Argument sein. Wenn das dritte Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1160	Bibliotheks-Pfadname ungültig Ein Pfadname muss in der Form xxx\yyy angegeben werden, wobei: <ul style="list-style-type: none"> • Der xxx Teil kann 1 bis 16 Zeichen haben. • Der yyy Teil kann 1 bis 15 Zeichen haben. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1170	Verwendung des Bibliotheks-Pfadnamens ungültig <ul style="list-style-type: none"> • Ein Wert kann einem Pfadnamen nicht mit Definiere (Define), := oder sto → zugewiesen werden. • Ein Pfadname kann nicht als lokale Variable festgelegt oder als Parameter in einer Funktions- oder Programmdefinition verwendet werden.
1180	Bibliotheks-Variablenname ungültig. Vergewissern Sie sich, dass der Name: <ul style="list-style-type: none"> • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 15 Zeichen Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1190	Bibliotheks-Dokument nicht gefunden: <ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliothek im Ordner MyLib befindet. • Aktualisieren Sie die Bibliotheken. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1200	Bibliotheksvariable nicht gefunden: <ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliotheksvariable im ersten Problem in der Bibliothek befindet. • Überprüfen Sie, dass die Bibliotheksvariable als LibPub oder LibPriv definiert wurde.

Fehlercode	Beschreibung
	<ul style="list-style-type: none"> • Aktualisieren Sie die Bibliotheken. <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1210	<p>Unzulässiger Name für Bibliothekskurzform.</p> <p>Vergewissern Sie sich, dass der Name:</p> <ul style="list-style-type: none"> • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 16 Zeichen • nicht reserviert ist <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation.</p>
1220	<p>Bereichsfehler:</p> <p>Die Funktionen tangentLine und normalLine unterstützen nur Funktionen mit reellen Werten.</p>
1230	<p>Bereichsfehler.</p> <p>Im Grad- und Neugradmodus werden die trigonometrischen Konversionsoperatoren nicht unterstützt.</p>
1250	<p>Argumentfehler</p> <p>System linearer Gleichungen verwenden.</p> <p>Beispiel für ein System zweier linearer Gleichungen mit den Variablen x und y:</p> $3x+7y=5$ $2y-5x=-1$
1260	<p>Argumentfehler:</p> <p>Das erste Argument von nfMin oder nfMax muss ein Ausdruck in einer einzigen Variablen sein. Es darf keine andere Variable ohne Wert außer der interessierenden Variablen enthalten.</p>
1270	<p>Argumentfehler</p> <p>Ordnung der Ableitung muss gleich 1 oder 2 sein.</p>
1280	<p>Argumentfehler</p> <p>Verwenden Sie ein Polynom in entwickelter Form in einer Variablen.</p>
1290	<p>Argumentfehler</p> <p>Verwenden Sie ein Polynom in einer Variablen.</p>
1300	<p>Argumentfehler</p> <p>Die Koeffizienten des Polynoms müssen numerische Werte ergeben.</p>

Fehlercode	Beschreibung
1310	Argumentfehler: Eine Funktion konnte für ein oder mehrere Argumente nicht ausgewertet werden.
1380	Argumentfehler: Verschachtelte Aufrufe der domain() Funktion sind nicht erlaubt.

Warncodes und -meldungen

Über die Funktion **warnCodes()** können Sie die bei der Auswertung eines Ausdrucks erzeugten Warnungen speichern. In dieser Tabelle sind alle numerischen Warncodes und die zugehörigen Meldungen aufgelistet.

Ein Beispiel zum Speichern von Warncodes finden Sie unter **warnCodes()** (Seite 184).

Warncode	Meldung
10000	Operation könnte falsche Lösungen erzeugen.
10001	Differenzieren einer Gleichung kann eine falsche Gleichung erzeugen.
10002	Zweifelhafte Lösung
10003	Zweifelhafte Genauigkeit
10004	Operation könnte Lösungen unterdrücken.
10005	clöse (cSolve) liefert u.U. mehrere Nullstellen.
10006	Löse (Solve) liefert u.U. mehrere Nullstellen.
10007	Weitere Lösungen möglich. Versuchen Sie, Ober- und Untergrenzen und/oder einen Schätzwert anzugeben. Beispiele mit solve(): <ul style="list-style-type: none">• solve(Gleichung, Var=Schätzwert) UntereGrenze<Var<ObereGrenze• solve(Gleichung, Var) UntereGrenze<Var<ObereGrenze• solve(Gleichung, Var=Schätzwert)
10008	Definitionsbereich des Ergebnisses kann kleiner sein als der der Eingabe.
10009	Definitionsbereich des Ergebnisses kann größer sein als der der Eingabe.
10012	Nicht-reelle Berechnung
10013	$\wedge 0$ oder $\text{undef} \wedge 0$ durch 1 ersetzt
10014	$\text{undef} \wedge 0$ durch 1 ersetzt
10015	1^\wedge oder 1^\wedgeundef durch 1 ersetzt
10016	1^\wedgeundef durch 1 ersetzt
10017	Überlauf durch ∞ oder $-\infty$ $\rho\sigma$
10018	Operation verlangt und liefert 64 Bit Wert.
10019	Ressourcen ausgeschöpft, Vereinfachung könnte unvollständig sein.
10020	Argument der trig. Funktion ist zu groß für eine exakte Vereinfachung.
10021	Eingabe enthält einen nicht definierten Parameter. Ergebnis gilt möglicherweise nicht für alle möglichen Parameterwerte.

Warncode	Meldung
10022	Eventuell erhalten Sie eine Lösung, wenn Sie geeignete Ober- und Untergrenzen festlegen.
10023	Skalar wurde mit Einheitsmatrix multipliziert.
10024	Ergebnis über approximierte Arithmetik erhalten.
10025	Äquivalenz kann im Modus EXAKT nicht verifiziert werden.
10026	Einschränkung wird möglicherweise ignoriert. Geben Sie Einschränkungen in der Form "\ 'Variable Konstante MatheTestSymbol' oder einer Verbindung dieser Formen an, z. B. 'x<3 und x>-12'

Allgemeine Informationen

Online-Hilfe

education.ti.com/eguide

Wählen Sie Ihr Land aus, um weitere Produktinformationen zu erhalten.

Kontakt mit TI Support aufnehmen

education.ti.com/ti-cares

Wählen Sie Ihr Land aus, um auf technische und sonstige Support-Ressourcen zuzugreifen.

Service- und Garantieinformationen

education.ti.com/warranty

Wählen Sie Ihr Land aus, informationen zur Dauer und zu den Bedingungen der Garantie bzw. zum Produktservice zu erhalten.

Eingeschränkte Garantie. Diese Garantie hat keine Auswirkungen auf Ihre gesetzlichen Rechte.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Index

		I	
-			
, subtrahieren	194	, womit-Operator	214
!		,	
!, Fakultät	204	', Minuten-Schreibweise	212
"		+	
", Sekunden-Schreibweise	212	+, addieren	193
#		<	
#, Umleitung	210	<, kleiner als	201
#, Umleitungsoperator	240	=	
%		=, gleich	200
%, Prozent	199	≠, ungleich[*]	200
*		>	
*, multiplizieren	194	>, größer als	202
.		Π	
.-, Punkt-Subtraktion	198	Π, Produkt	207
.*, Punkt-Multiplikation	198	Σ	
./, Punkt-Division	198	Σ(), Summe	207
.^, Punkt-Potenz	199	ΣInt()	208
.+, Punkt-Addition	197	ΣPrn()	209
/		√	
/, dividieren	195	√(), Quadratwurzel	206
:		∠	
:=, zuweisen	216	∠, winkel	212
^		ʃ	
^-1, Kehrwert	214	ʃ, Integral	206
^, Potenz	196	≤	
		≤, kleiner oder gleich	201

\geq		$^{\circ}$, Grad/Minute/Sekunde	212
\geq , größer oder gleich	202	0	
►		0b, binäre Anzeige	217
►, in Neugrad umwandeln	75	0h, hexadezimale Anzeige	217
►approxFraction()	13	1	
►Base10, Anzeige als ganze Dezimalzahl [Base10]	18	10^(), Potenz von zehn	213
►Base16, Hexadezimaldarstellung [Base16]	19	A	
►Base2, Binärdarstellung[Base2]	17	Abbruch, Exit	53
►Cylind, Anzeige als Zylindervektor [Cylind (Zylindervektor)] ...	38	Ableitungen erste Ableitung, d()	205
►DD, Anzeige als Dezimalwinkel[DD (Dezimalwinkel)]	39	numerische Ableitung, nDeriv()	112
►Decimal, Anzeige als Dezimalzahl [Dezimal]	39	numerische Ableitung, nDerivative()	110
►DMS, Anzeige als Grad/Minute/Sekunde[DMS (GMS)]	47	abrufen/zurückgeben Variableninformationen, getVarInfo()	70
►Polar, Anzeige als Polarvektor[Polar]	123	Abrufen/zurückgeben Variableninformationen, getVarInfo()	73
►Rad, in Bogenmaß umwandeln ...	132	abs(), Absolutwert	7
►Rect, Anzeige als kartesischer Vektor	135	Absolutwert Vorlage für	3-4
►Sphere, Anzeige als sphärischer Vektor[Sphere (Kugelkoordinaten)]	161	addieren, +	193
⇒		als kartesischen Vektor anzeigen, ►Rect	135
⇒, logische Implikation	203, 237	Amortisationstabelle, amortTbl() ..	7, 16
→		amortTbl(), Amortisationstabelle ..	7, 16
→, speichern	215	and, Boolean operator	8
↔		and, Boolesches und	8
↔, logische doppelte Implikation[*]	204	angle(), Winkel	9
©		ANOVA, einfache Varianzanalyse ..	9
©, Kommentar	216	ANOVA2way, zweifache Varianzanalyse	10
°		Ans, letzte Antwort	12
°, Grad-Schreibweise	211	Antwort (letzte), Ans	12
		Anz, Daten anzeigen	148
		Anzeige als binär, ►Base2	17
		Dezimalwinkel, ►DD	39
		ganze Dezimalzahl, ►Base10	18

Grad/Minute/Sekunde, ►DMS	47	Bestimmtes Integral	
hexadezimal, ►Base16	19	Vorlage für	6
Polarvektor, ►Polar	123	Bibliothek	
sphärischer Vektor, ►Sphere	161	erstelle Tastaturbefehle für	
Zylindervektor, ►Cylind	38	Objekte	85
Anzeige als kartesischer Vektor,		binär	
►Rect	135	Anzeige, Ob	217
Anzeige als sphärischer Vektor,		Darstellung, ►Base2	17
►Sphere	161	binomCdf()	19, 80-81
Anzeige als Zylindervektor, ►Cylind	38	binomPdf()	20
approx(), approximieren	13	Bogenmaß, r	211
approximieren, approx()	13	Boolean operators	
approxRational()	13	and	8
arccos()	13	Boolesch	
arccosh()	14	und, and	8
arccot()	14	Boolesche Operatoren	
arccoth()	14	⇒	203, 237
arccsc()	14	↔	204
arccsch()	14	nand	109
arcsec()	14	nicht	115
arcsech()	14	nor	113
arcsin()	14	oder	119
arcsinh()	14	xor	186
arctan()	15	Brüche	
arctanh()	15	propFrac (Echter Bruch)	127
Argumente in TVM-Funktionen	179	Vorlage für	1
Arkuskosinus, cos ⁻¹ ()	29	C	
Arkussinus, sin ⁻¹ ()	156	Cdf()	57
Arkustangens, tan ⁻¹ ()	169	ceiling(), Obergrenze	20
augment(), erweitern/verketten	15	centralDiff()	21
Ausdrücke		char(), Zeichenstring	21
String in Ausdruck, expr()	54	X ² way	22
Ausschließung mit „ “ Operator	214	ClearAZ	24
Auswertungsreihenfolge	239	colAugment	25
avgRC(), durchschnittliche		colDim(), Spaltendimension der	
Änderungsrate	15	Matrix	25
B		colNorm(), Spaltennorm der Matrix	25
Befehl Stopp	165	conj(), Komplex Konjugierte	26
benutzerdefinierte Funktionen	40	constructMat(), Matrix erstellen	26
benutzerdefinierte Funktionen und		corrMat(), Korrelationsmatrix	27
Programme	41	cos ⁻¹ , Arkuskosinus	29
cos(), Kosinus		cos(), Kosinus	27
cosh ⁻¹ (), Arkuskosinus hyperbolicus		cosh ⁻¹ (), Arkuskosinus hyperbolicus	30

cosh(), Cosinus hyperbolicus	30	Dimension, dim()	44
cot ⁻¹ (), Arkuskotangens	31	DispAt	45
cot(), Kotangens	31	dividieren, /	195
coth ⁻¹ (), Arkuskotangens hyperbolicus	32	dotP(), Skalarprodukt	47
coth(), Kotangens hyperbolicus	32	drehen, rotate()	143
countIf(), Elemente in einer Liste bedingt zählen	33	durchschnittliche Änderungsrate, avgRC()	15
cPolyRoots()	33		
crossP(), Kreuzprodukt	34	E	
csc ⁻¹ (), inverser Kosekans	34	e Exponent Vorlage für	2
csc(), Kosekans	34	e hoch x, e^()	48, 54
csch ⁻¹ (), inverser Kosekans hyperbolicus	35	E, Exponent	210
csch(), Kosekans hyperbolicus	35	e^(), e hoch x	48
CubicReg, kubische Regression	36	echter Bruch, propFrac	127
Cycle, Zyklus	37	eff(), Nominal- in Effektivsatz konvertieren	48
		Effektivsatz, eff()	48
		Eigenvektor, eigVc()	49
		Eigenwert, eigVl()	49
d(), erste Ableitung	205	eigVc(), Eigenvektor	49
Daten anzeigen, Anz	148	eigVl(), Eigenwert	49
Daten anzeigen, Disp	45	Einheitsmatrix, identity()	75
dbd(), Tage zwischen Daten	38	Einheitsvektor, unitV()	181
Define, definiere	40	Einstellungen, hole aktuellen	71
Definiere	40	Elemente in einer Liste bedingt zählen, countIf()	33
Definiere LibPriv (Define LibPriv)	41	Elemente in einer Liste zählen, zähle()	32
Definiere LibPub (Define LibPub)	41	else if, ElseIf	50
Definiere, Define	40	else, Else	75
definieren		ElseIf, else if	50
öffentliche Funktion /		end	
öffentliches Programm		for, EndFor	59
private Funktion oder		if, EndIf	75
Programm	41	Schleife, EndLoop	98
deltaList()	42	while, EndWhile	185
DelVar, Variable löschen	42	end if, EndIf	75
delVoid(), ungültige Elemente entfernen	43	end while, EndWhile	185
det(), Matrixdeterminante	43	Ende	
Dezimal		Funktion, EndFunc	63
Anzeige als ganze Zahl, ►Base10	18	Ende der Schleife, EndLoop	98
Winkelanzeige, ►DD	39	EndWhile, end while	185
diag(), Matrixdiagonale	44		
Diagonalform, ref()	136		
dim(), Dimension	44		

Entfernen			
ungültige Elemente aus Liste	43	For, for	59
EOS (Equation Operating System) ..	239	format(), Formatstring	60
Equation Operating System (EOS) ..	239	Formatstring, format()	60
Ergebnisse mit zwei Variablen,		fpart(), Funktionsteil	60
TwoVar	180	freqTable()	61
Ergebnisse, Statistik	162	Frobeniusnorm, norm()	114
Ergebniswerte, Statistik	163	Füllen	227-228
Ersetzung durch „ “ Operator	214	Func, Funktion	63
erste Ableitung		Func, Programmfunktion	63
Vorlage für	5	Funktion beenden, EndFunc	63
erweitern/verketten, augment()	15	Funktionen	
euler(), Euler function	51	benutzerdefiniert	40
Exit, Abbruch	53	Programmfunktion, Func	63
exp(), e hoch x	54	Teil, fpart()	60
Exponent, E	210	Funktionen und Variablen	
Exponenten		kopieren	27
Vorlage für	1		G
Exponentielle Regression, ExpReg ..	54	g, Neugrad	210
expr(), String in Ausdruck	54	ganze Zahl, int()	78
ExpReg, exponentielle Regression ..	54	Ganzzahl teilen, intDiv()	79
		ganzzahliger Teil, iPart()	82
F		gcd(), größter gemeinsamer Teiler ..	64
factor(), Faktorisiere	56	gehe zu, Goto	74
Faktorisiere, factor()	56	geomCdf()	65
Fakultät, !	204	geomPdf()	65
Fehler übergeben, Ügebefehl	122	Get	65, 229
Fehler und Fehlerbehebung		getDenom(), Nenner	
Fehler löschen, LöFehler	24	holen/zurückgeben	67
Fehler übergeben, Ügebefehl	122	getLangInfo(), Sprachinformationen	
festlegen		abrufen/zurückgeben	70
Modus, setMode()	151	getLockInfo(), testet den Gesperrt-	
Fill, Matrix füllen	57	Status einer Variablen oder	
Finanzfunktionen, tvmFV()	178	Variabengruppe	71
Finanzfunktionen, tvmI()	178	getMode(), getMode-Einstellungen ..	71
Finanzfunktionen, tvmN()	178	getNum(), Zähler	
Finanzfunktionen, tvmPmt()	179	holen/zurückgeben	72
Finanzfunktionen, tvmPV()	179	GetStr	73
FiveNumSummary	57	getType(), get type of variable	73
floor(), Untergrenze	58	getVarInfo(),	
Folge, seq()	149	Variableninformationen	
For	59	abrufen/zurückgeben	73
for, For	59	gleich, =	200

Gleichungssystem (2 Gleichungen)		Integral, \int	206
Vorlage für	3	Interpolieren(), interpolieren	79
Gleichungssystem (n Gleichungen)		invF()	80
Vorlage für	3	invNorm(), inverse kumulative Normalverteilung)	81
Gleichungssystem, simulat()	155	invvt()	81
Goto, gehe zu	74	Invx ² ()	80
Grad-/Minuten-/Sekundenanzeige, ►DMS	47	iPart(), Ganzzahliger Teil	82
Grad-Schreibweise, °	211	irr(), interner Zinsfluss interner Zinsfluss, irr()	82
größer als, >	202	isPrime(), Primzahltest	82
Größer oder gleich, ≥	202	isVoid(), Test auf Ungültigkeit	83
größter gemeinsamer Teiler, gcd() ..	64		
Gruppen, Gesperrt-Status testen ..	71		
Gruppen, sperren und entsperren ..	95, 182		
K			
Häufigkeit()	62	kartesische x-Koordinate, P►Rx() ...	121
hexadezimal		kartesische y-Koordinate, P►Ry() ...	121
Anzeige, ►Base16	19	Kehrwert, \wedge^{-1}	214
Anzeige, 0h	217	Ketten drehen, rotate()	143
holen/zurückgeben		kleiner als, <	201
Nenner, getDenom()	67	Kleiner oder gleich, ≤	201
Zähler, getNum()	72	kleinstes gemeinsames Vielfaches, lcm	84
holeTast	67	Kombinationen, nCr()	109
Hyperbolisch		Kommentar, ©	216
Arkuskosinus, cosh ⁻¹ ()	30	komplex Konjugierte, conj()	26
Arkussinus, sinh ⁻¹ ()	158	konvertieren ►Rad	132
Arkustangens, tanh ⁻¹ ()	170	Korrelationsmatrix, corrMat()	27
Cosinus, cosh()	30	Kosinus, cos()	27
Sinus, sinh()	157	Kotangens, cot()	31
Tangens, tanh()	170	Kreuzprodukt, crossP()	34
I			
identity(), Einheitsmatrix	75	kubische Regression, CubicReg	36
if, If	75	kumulierte Summe, cumulativeSum()	37
If, if	75	kumulierteSumme(), kumulierte Summe	37
ifFn()	76		
imag(), Imaginärteil	77		
Imaginärteil, imag()	77		
in String, inString()	78	L	
inString(), in String	78	Lbl, Marke	83
int(), ganze Zahl	78	lcm, kleinstes gemeinsames Vielfaches	84
intDiv(), Ganzzahl teilen	79		

leere (ungültige) Elemente	235	Lock, Variable oder Variablengruppe sperren	95
left(), links	84	LöFehler, Fehler löschen	24
LibPriv	41	Logarithmen	92
LibPub	41	Logarithmische Regression, LnReg ..	93
libShortcut(), erstelle Tastaturlbefehle für Bibliotheksobjekte	85	Logarithmus Vorlage für	2
lineare Regression, LinRegAx	86	logische doppelte Implikation, \Leftrightarrow ..	204
lineare Regression, LinRegBx	85	logische Implikation, \Rightarrow ..	203, 237
Lineare Regression, LinRegBx	88	Logistic, logistische Regression	96
links, left()	84	LogisticD, logistische Regression	97
LinRegBx, lineare Regression	85	Logistische Regression, Logistic	96
LinRegMx, lineare Regression	86	Logistische Regression, LogisticD	97
LinRegIntervals, lineare Regression	88	lokal, Local	94
LinRegtTest	89	lokale Variable, Local	94
linSolve()	91	Loop, Schleife	98
list►mat(), Liste in Matrix	92	löschen Variable, DelVar	42
Liste in Matrix, list►mat()	92	Löschen	222
Liste, Elemente bedingt zählen	33	Fehler, LöFehler	24
Liste, Elemente zählen in	32	ungültige Elemente aus Liste	43
Listen		LU, untere/obere Matrixzerlegung ..	99
Differenzen in einer Liste, Δlist()	91		
erweitern/verketten, augment()	15		
in absteigender Reihenfolge	160	M	
sortieren, SortD	160	Marke, Lbl	83
in aufsteigender Reihenfolge	160	mat►list(), Matrix in Liste	100
sortieren, SortA	34	Matrix	
Kreuzprodukt, crossP()	34	Diagonalfomr, ref()	136
kumulierte Summe,	37	reduzierte Diagonalfomr, rref()	146
cumulativeSum()	37	Matrix (1 x 2)	
leere Elemente in	235	Vorlage für	4
Liste in Matrix, list►mat()	92	Matrix (2 x 1)	
Matrix in Liste, mat►list()	100	Vorlage für	4
Maximum, max()	100	Matrix (2 x 2)	
Minimum, min()	104	Vorlage für	4
neu, newList()	111	Matrix ($m \times n$)	
Produkt, product()	127	Vorlage für	4
Skalarprodukt, dotP()	47	Matrix erstellen, constructMat()() ..	26
Summe, sum()	166	Matrix in Liste, mat►list()	100
Summierung, sum()	167	Matrixzeilenaddition, rowAdd()	145
Teil-String, mid()	103	Matrixzeilentausch, rowSwap()	146
ln(), natürlicher Logarithmus	92	Matrizen	
LnReg, logarithmische Regression ..	93	Determinante, det()	43
Local, lokale Variable	94	Diagonale, diag()	44

Dimension, dim()	44	mit,	214
Eigenvektor, eigVc()	49	Mittellinienregression, MedMed	102
Eigenwert, eigVl()	49	Mittelwert, mean()	101
Einheitsmatrix, identity()	75	mod(), Modulo	105
erweitern/verketten, augment()	15	Modi	
füllen, Fill	57	festlegen, setMode()	151
kumulierte Summe, cumulativeSum()	37	Modifizierter interner Zinsfluss, mirr())	104
Liste in Matrix, list2mat()	92	Modulo, mod()	105
Matrix in Liste, mat2list()	100	Moduseinstellungen, getMode()	71
Matrixzeilenmultiplikation und - addition, mRowAdd()	106	mRow(), Matrixzeilenoperation	105
Maximum, max()	100	mRowAdd(), Matrixzeilenmultiplikation und -addition	106
Minimum, min()	104	Multipler linearer Regressions-t-Test	107
neu, newMat()	111	multiplizieren, *	194
Produkt, product()	127	MultReg (Mehrfachregression)	106
Punkt-Addition, .+	197	MultRegIntervals() (Mehrfachregressionsinterv all)	106
Punkt-Division, ./	198	MultRegTests()	107
Punkt-Multiplikation, .*	198		
Punkt-Potenz, .^	199		
Punkt-Subtraktion, .-	198		
QR-Faktorisierung, QR	128	N	
Spaltendimension, colDim()	25		
Spaltennorm, colNorm()	25	n-te Wurzel	
Summe, sum()	166	Vorlage für	2
Summierung, sum()	167	nand, Boolescher Operator	109
Transponierte, T	168	natürlicher Logarithmus, ln()	92
untere/obere Matrixzerlegung, LU	99	nCr(), Kombinationen	109
Untermatrix, subMat()	166, 168	nDerivative(), numerische Ableitung	110
Zeilenoperation, mRow()	105	Negation, Eingabe von negativen Zahlen	240
max(), Maximum	100	Nettobarwert, npv()	117
Maximum, max()	100	neu	
mean(), Mittelwert	101	Liste, newList()	111
median(), Median	101	Matrix, newMat()	111
Median, median()	101	Neugrad-Schreibweise, g	210
MedMed, Mittellinienregression	102	newList(), neue Liste	111
mid(), Teil-String	103	newMat(), neue Matrix	111
min(), Minimum	104	nfMax(), numerisches Funktionsmaximum	112
Minimum, min()	104	nfMin(), numerisches Funktionsminimum	112
Minuten-Schreibweise,	212	nicht, Boolescher Operator	115
mirr(), modifizierter interner Zinsfluss	104	nint(), numerisches Integral	112

nom), Effektivzins in Nominalzins		polar	
konvertieren	113	Vektoranzeige, ►Polar	123
Nominalzinssatz, nom()	113	Polarkoordinate, R►Pr()	132
nor, Boolescher Operator	113	Polarkoordinate, R►Pθ()	131
norm(), Frobeniusnorm	114	polyEval(), Polynom auswerten	124
Normalverteilung invertieren		Polynom auswerten, polyEval()	124
(invNorm())	81	Polynome	
Normalverteilungswahrscheinlichkeit,		auswerten, polyEval()	124
normCdf()	114	PolyRoots()	124
normCdf()		Potenz von zehn, 10^()	213
(Normalverteilungswahrscheinlichkeit)	114	Potenz, ^	196
normPdf()		Potenzregression, 124-125, 138, 140,	
(Wahrscheinlichkeitsdichte)	115	PowerReg ..	171
nPr(), Permutationen	116	PowerReg, Potenzregression	125
npv(), Nettobarwert	117	Prgm, Definiere Programm	126
nSolve(), numerische Lösung	117	Primzahltest, isPrime()	82
numerisch		prodSeq()	127
Ableitung, nDeriv()	112	product(), Produkt	127
Ableitung, nDerivative()	110	Produkt $\prod()$	
Integral, nInt()	112	Vorlage für	5
Lösung, nSolve()	117	Produkt, $\prod()$	207
Produkt, product()		Produkt, product()	127
O		Programme	
Obergrenze, ceiling()	20-21, 33	öffentliche Bibliothek definieren	41
Objekte		Private Bibliothek definieren	41
erstelle Tastaturlbefehle für		Programme und Programmieren	
Bibliothek	85	E/A-Bildschirm anzeigen, Anz	148
oder (Boolesch), oder	119	E/A-Bildschirm anzeigen, Zeige	45
oder, Boolescher Operator	119	Fehler löschen, LöFehler	24
OneVar, Statistik mit einer Variable ..	118	programmieren	
Operatoren		Daten anzeigen, Disp	45
Auswertungsreihenfolge	239	Definiere Programm, Prgm	126
ord(), numerischer Zeichencode ...	121	Fehler übergeben, ÜbgebFeh	122
P		Programmierung	
P►Rx(), kartesische x-Koordinate ...	121	Daten anzeigen, Anz	148
P►Ry(), kartesische y-Koordinate ...	121	propFrac, echter Bruch	127
Pdf()	61	Prozent, %	199
Permutationen, nPr()	116	Punkt	
piecewise() (Stückweise)	122	Addition, .+	197
poissCdf()	123	Division, ./	198
poissPdf()	123	Multiplikation, .*	198
		Potenz, ^	199
		Subtraktion, .-	198

Q		MultReg (Mehrfachregression) 106
QR-Faktorisierung, QR	128	Potenzregression, 124-125, 138,
QR,QR-Faktorisierung	128	PowerReg 140, 171
Quadratische Regression, QuadReg .	129	quadratische, QuadReg 129
Quadratwurzel		sinusförmige, SinReg 158
Vorlage für	1	vierter Ordnung, QuartReg 130
Quadratwurzel, $\sqrt{ }()$	206	remain(), Rest 138
Quadratwurzel, $\sharp()$	161	Request 138
QuadReg, quadratische Regression .	129	RequestStr 140
QuartReg, Regression vierter		Rest, remain() 138
Ordnung	130	Return, Rückgabe 141
R		right(), rechts 141
r, Bogenmaß	211	right, right() 51, 184
R>Pr(), Polarkoordinate	132	rk23(), Runge-Kutta-Funktion 142
R>Pθ(), Polarkoordinate		rotate(), drehen 143
rand(), Zufallszahl	131	round(), runden 145
randBin, Zufallszahl		rowAdd(), Matrixzeilenaddition 145
randInt(), ganzzahlige Zufallszahl ..	133	rowDim(), Zeilendimension der
randMat(), Zufallsmatrix	133	Matrix 146
randNorm(), Zufallsnorm	134	rowNorm(), Zeilennorm der Matrix 146
randPoly(), Zufallspolynom	134	rowSwap(), Matrixzeilentausch 146
randSamp()	134	rref(), reduzierte Diagonalform 146
RandSeed, Zufallszahl	135	Rückgabe, Return 141
real(), reel	135	runden, round() 145
rechts, right() 79, 141-142		
reduzierte Diagonalform, rref()	146	
reell, real()	135	Schleife, Loop 98
ref(), Diagonalform	136	Schreibweise Grad/Minute/Sekunde 212
RefreshProbeVars	137	sec ⁻¹ (), Arkussekans 147
Regression vierter Ordnung,		sec(), Sekans 147
QuartReg	130	sech ⁻¹ (), Arkussekans hyperbolicus 148
Regressionen		sech(), Sekans hyperbolicus 148
exponentielle, ExpReg	54	Sekunden-Schreibweise, " 212
kubische, CubicReg	36	seq(), Folge 149
lineare Regression, LinRegAx ..	86	seqGen() 149
lineare Regression, LinRegBx ..	85	seqn() 150
Lineare Regression, LinRegBx ..	88	sequence, seq() 149-150
logarithmische, LnReg	93	setMode(), Modus festlegen 151
Logistic (Logistisch)	96	shift(), verschieben 153
logistische, Logistic	97	sign(), Zeichen 154
Mittellinie, MedMed	102	simult(), Gleichungssystem 155
		sin ⁻¹ (), Arkussinus 156

sin(), Sinus	156	String	
sinh ⁻¹ (), Arkussinus hyperbolicus	158	Dimension, dim()	44
sinh(), Sinus hyperbolicus	157	Länge	44
SinReg, sinusförmige Regression	158	string(), Ausdruck in String	165
Sinus, sin()	156	Stringlänge	44
Sinusförmige Regression, SinReg	158	strings	
Skalar		right, right()	51, 184
Produkt, dotP()	47	Strings	
SortA, in aufsteigender Reihenfolge		Ausdruck in String, string()	165
sortieren	160	Format, format()	60
SortD, in absteigender Reihenfolge		Formatieren	60
sortieren	160	in, inString	78
sortieren		links, left()	84
in absteigender Reihenfolge		rechts, right()	79, 141-142
sortieren, SortD	160	String in Ausdruck, expr()	54
in aufsteigender Reihenfolge,		Teil-String, mid()	103
SortA	160	Umleitung, #	210
speichern		verschieben, shift()	153
Symbol, →	215	Zeichencode, ord()	121
Sprache		Zeichenstring, char()	21
Sprachinformation abrufen ...	70	Stückweise definierte Funktion (2 Teile)	
sqrt(), Quadratwurzel	161	Vorlage für	2
Standardabweichung, stdDev()	163-164, 182	Stückweise definierte Funktion (n Teile)	
stat.results	162	Vorlage für	3
stat.values	163	Student-t-	
Statistik		Wahrscheinlichkeitsdichte,	
Ergebnisse mit zwei Variablen,		tPdf()	174
TwoVar	180	subMat(), Untermatrix	166, 168
Fakultät, !	204	subtrahieren, -	194
Kombinationen, nCr()	109	sum(), Summe	166
Median, median()	101	sumIf()	167
Mittelwert, mean()	101	Summe Σ()	
Permutationen, nPr()	116	Vorlage für	5
Standardabweichung,		Summe der Tilgungszahlungen	209
stdDev()	163-164, 182	Summe der Zinszahlungen	208
Statistik mit einer Variable,		Summe, Σ()	207
OneVar	118	Summe, sum()	166
Varianz, variance()	182	sumSeq()	167
Zufallsnorm, randNorm()	134		T
Zufallszahl, RandSeed	135		
Statistik mit einer Variable, OneVar	118	t test, t-Test	176
stdDevPop(), Populations-Standardabweichung	163		
stdDevSamp(), Stichproben-Standardabweichung	164		

T, Transponierte	168	unLock, Variable oder Variablengruppe entsperren	182
Tage zwischen Daten, dbd()	38	Untergrenze, floor()	58
tan ⁻¹ (), Arkustangens	169	Untermatrix, subMat()	166, 168
tan(), Tangens	168		
Tangens, tan()	168	V	
tanh ⁻¹ (), Arkustangens hyperbolicus	170		
tanh(), Tangens hyperbolicus	170	Variable	
Tastenkürzel	170	Name aus String erstellen	240
Tastenkürzel, Tastatur	237	Variable oder Funktion kopieren, CopyVar	27
tCdf(), Wahrscheinlichkeit einer Student t-Verteilung	171	Variablen	
Teil-String, mid()	103	alle einbuchstabigen löschen	24
Test auf Ungültigkeit, isVoid()	83	lokal, Local	94
Test_2S, Zwei-Stichproben F-Test	62	löschen, DelVar	42
Text, Befehl	171	Variablen und Funktionen	
tInterval, Konfidenzintervall t	172	kopieren	27
tInterval_2Samp, ZweiStichproben- t-Konfidenzintervall	173	Variablen und Variablengruppen	
trace()	174	entsperren	182
Transponierte, T	168	Variablen und Variablengruppen	
Try, Befehl zur Fehlerbehandlung	175	sperren	95
tTest, t-Test	176	Variablen, sperren und	
tTest_2Samp, Zwei-Stichproben-t- Test	177	entsperren	71, 95, 182
TVM-Argumente	179	Varianz, variance()	182
tvmFV()	178	varPop() (Populationsvarianz)	182
tvmI()	178	varSamp(), Stichproben-Varianz ...	182
tvmN()	178		
tvmPmt()	179	Vektoren	
tvmPV()	179	Anzeige als Zylindervektor, ►Cylind	38
TwoVar, Ergebnisse mit zwei Variablen	180	Einheit, unitV()	181
		Kreuzprodukt, crossP()	34
		Skalarprodukt, dotP()	47
		verschieben, shift()	153
U		Verteilungsfunktionen	
Ügebefehl, Fehler übergeben	122	binomCdf()	19, 80-81
Umleitung, #	210	binomPdf()	20
Umleitungsoperator (#)	240	invNorm()	81
umwandeln		invt()	81
►Grad (Neugrad)	75	Invχ ² ()	80
ungleich, ≠	200	normCdf()	
ungültige Elemente	235	(Normalverteilungswahrschei nlichkeit)	114
ungültige Elemente, entfernen	43	normPdf()	
unitV(), Einheitsvektor	181	(Wahrscheinlichkeitsdic hte)	115
		poissCdf()	123

poissPdf()	123	while, While	185
tCdf()	171	While, while	185
tPpdf()	174	winkel, \angle	212
χ^2 2way()	22	Winkel, angle()	9
χ^2 Cdf()	22	womit-Operator „ “	214
χ^2 GOF()	23	womit-Operator,	
χ^2 Pdf()	24	Auswerungsreihenfolge ...	239
void, test for	83		
Vorlagen		X	
Absolutwert	3-4	χ^2 , Quadrat	197
Bestimmtes Integral	6	XNOR	204
Bruch	1	xor, Boolesches exklusives oder	186
e Exponent	2		
erste Ableitung	5	Z	
Exponent	1	Zähle Tage zwischen Daten, dbd()	38
Gleichungssystem (2 Gleichungen)	3	zähl(), Elemente in einer Liste zählen	32
Gleichungssystem (n Gleichungen)	3	Zeichen	
Logarithmus	2	String, char()	21
Matrix (1×2)	4	Zeichencode, ord()	121
Matrix (2×1)	4	Zeichen, sign()	154
Matrix (2×2)	4	Zeichenfolgen	
Matrix ($m \times n$)	4	zum Erstellen von Variablennamen	
n-te Wurzel	2	verwenden	240
Produkt $\prod()$	5	Zeichenstring, char()	21
Quadratwurzel	1	Zeichnen	223-225
Stückweise definierte Funktion (2 Teile)	2	Zeige, Daten anzeigen	45
Stückweise definierte Funktion (n Teile)	3	Zeilendimension der Matrix, rowDim	
Summe $\sum()$	5	()	146
zweite Ableitung	6	Zeilennorm der Matrix, rowNorm()	146
		Zeitwert des Geldes, Anzahl	
		Zahlungen	178
		Zeitwert des Geldes, Barwert	179
		Zeitwert des Geldes, Endwert	178
		Zeitwert des Geldes, Zahlungsbetrag	179
		Zeitwert des Geldes, Zinsen	178
		zInterval, z-Konfidenzintervall	187
		zInterval_1Prop, z-Konfidenzintervall	
		für eine Proportion	187
		zInterval_2Prop, z-Konfidenzintervall	
		für zwei Proportionen	188
		zInterval_2Samp, z-	
		Konfidenzintervall für zwei	
		Stichproben	189
	W		
Wahrscheinlichkeit einer Student-t-Verteilung, tCdf()	171		
Wahrscheinlichkeitsdichte, normPpdf()	115		
Warncodes und -meldungen	254		
warnCodes(), Warning codes	184		
Warte-Befehl	183		
wenn, when()	184		
when(), wenn	184		

zTest	189
zTest_1Prop, z-Test für eine Proportion	190
zTest_2Prop, z-Test für zwei Proportionen	191
zTest_2Samp, z-Test für zwei Stichproben	192
Zufallsmatrix, randMat()	134
Zufallsnorm, randNorm()	134
Zufallspolynom, randPoly()	134
Zufallsstichprobe	134
Zufallszahl, RandSeed	135
zuweisen, :=	216
Zwei-Stichproben F-Test	62
zweite Ableitung	
Vorlage für	6
Zyklus, Cycle	37

Δ

Δlist(), Listendifferenz	91
---------------------------------	----

X

χ²Cdf()	22
χ²GOF	23
χ²Pdf()	24