

TI-Nspire™ CX Referenzhandbuch

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.

© 2023 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	21
D	38
E	48
F	56
G	64
I	76
L	84
M	101
N	110
O	120
P	123
Q	130
R	134
S	150
T	172
U	186
V	186
W	188
X	190
Z	191
Sonderzeichen	197
TI-Nspire™ CX II – Zeichenbefehle	224
Grafikprogrammierung	224
Grafikbildschirm	224
Standardansicht und Einstellungen	225
Fehlermeldungen des Grafikbildschirms	226
Im Grafikmodus ungültige Befehle	226
C	228
D	229
F	233
G	235
P	236
F:	238
U	240

Leere (ungültige) Elemente	241
Tastenkürzel zum Eingeben mathematischer Ausdrücke	243
Auswertungsreihenfolge in EOS™ (Equation Operating System)	245
TI-Nspire CX II – TI-Basic Programmierfunktionen	247
Automatisches Einrücken im Programmierungseditor	247
Verbesserte Fehlermeldungen für TI-Basic	247
Konstanten und Werte	250
Fehlercodes und -meldungen	251
Warncodes und -meldungen	260
Allgemeine Informationen	262
Inhalt	263

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 200.

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 201.

Beispiel:

$$2^3 = 8$$

Vorlage Quadratwurzel

ctrl **x²** Tasten



Hinweis: Siehe auch $\sqrt{\quad}$ (Quadratwurzel), Seite 211.

Beispiel:

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

Vorlage n-te Wurzel

ctrl **^** **Tasten**

$\sqrt{\quad}$

$\sqrt{\quad}$ **Hinweis:** Siehe auch **root()**, Seite 146.

Beispiel:

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

Vorlage e Exponent

e^x **Tasten**

e \square

Potenz zur natürlichen Basis e

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

Example:

$$e^1 \quad 2.71828182846$$

Vorlage Logarithmus

ctrl **10^x** **Taste**

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

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

Hinweis: Siehe auch **log()**, Seite 96.

Beispiel:

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

Vorlage Stückweise (2 Teile)

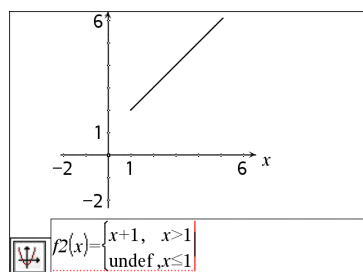
Katalog **>** $\left\{ \begin{array}{l} \square \\ \square \end{array} \right.$

$\left\{ \begin{array}{l} \square \\ \square \end{array} \right.$

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 125.

Beispiel:



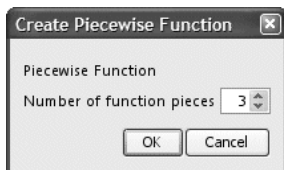
Vorlage Stückweise (n Teile)

Katalog > 

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

Beispiel:

Siehe Beispiel für die Vorlage Stückweise (2 Teile).



Hinweis: Siehe auch `piecewise()`, Seite 125.

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.

Beispiel:

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

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

Hinweis: Siehe auch `system()`, Seite 172.

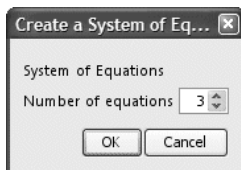
Vorlage System von n Gleichungen

Katalog > 

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

Beispiel:

Siehe Beispiel für die Vorlage Gleichungssystem (2 Gleichungen).



Hinweis: Siehe auch `system()`, Seite 172.

Vorlage Absolutwert

Katalog > 



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

Beispiel:

Vorlage Absolutwert

Katalog > 

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

Vorlage dd°mm'ss.ss''

Katalog > 

Beispiel:

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

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

Vorlage Matrix (2 x 2)

Katalog > 

Beispiel:

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

Erzeugt eine 2 x 2 Matrix.

Vorlage Matrix (1 x 2)

Katalog > 

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 > 

Beispiel:

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

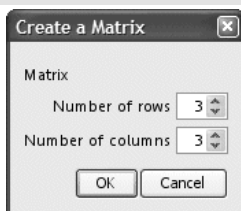
Vorlage Matrix (m x n)

Katalog > 

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

Beispiel:

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



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

Vorlage Summe (Σ)

$$\sum_{i=1}^n (i)$$

Beispiel:

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

Hinweis: Siehe auch $\Sigma()$ (**sumSeq**), Seite 212.

Vorlage Produkt (Π)

$$\prod_{i=1}^n (i)$$

Beispiel:

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

Hinweis: Siehe auch $\Pi()$ (**prodSeq**), Seite 212.

Vorlage Erste Ableitung

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

Beispiel:

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

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 210.

Vorlage Zweite Ableitung

Katalog > 

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

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 210.

Beispiel:

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

Vorlage Bestimmtes Integral

Katalog > 

$$\int_0^1 x dx$$

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 114.


Beispiel:

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

Alphabetische Auflistung

Elemente, deren Namen nicht alphabetisch sind (wie +, !, und >) finden Sie am Ende dieses Abschnitts (Seite 197). Wenn nicht anders angegeben, wurden sämtliche Beispiele im standardmäßigen Reset-Modus ausgeführt, wobei alle Variablen als nicht definiert angenommen wurden.

A


abs() (Absolutwert)	Katalog > 
abs(Wert1) ⇒ Wert	$\left\{ \left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\} \right\}$ { 1.5708, 1.0472 }
abs(Liste1) ⇒ Liste	$ 2-3 \cdot i $ 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()	Katalog > 																																																				
amortTbl(NPmt, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) ⇒ Matrix	amortTbl(12,60,10,5000,,12,12)																																																				
Amortisationsfunktion, die eine Matrix als Amortisationstabelle für eine Reihe von TVM-Argumenten zurückgibt.	<table border="1"> <tr><td>0</td><td>0.</td><td>0.</td><td>5000.</td></tr> <tr><td>1</td><td>-41.67</td><td>-64.57</td><td>4935.43</td></tr> <tr><td>2</td><td>-41.13</td><td>-65.11</td><td>4870.32</td></tr> <tr><td>3</td><td>-40.59</td><td>-65.65</td><td>4804.67</td></tr> <tr><td>4</td><td>-40.04</td><td>-66.2</td><td>4738.47</td></tr> <tr><td>5</td><td>-39.49</td><td>-66.75</td><td>4671.72</td></tr> <tr><td>6</td><td>-38.93</td><td>-67.31</td><td>4604.41</td></tr> <tr><td>7</td><td>-38.37</td><td>-67.87</td><td>4536.54</td></tr> <tr><td>8</td><td>-37.8</td><td>-68.44</td><td>4468.1</td></tr> <tr><td>9</td><td>-37.23</td><td>-69.01</td><td>4399.09</td></tr> <tr><td>10</td><td>-36.66</td><td>-69.58</td><td>4329.51</td></tr> <tr><td>11</td><td>-36.08</td><td>-70.16</td><td>4259.35</td></tr> <tr><td>12</td><td>-35.49</td><td>-70.75</td><td>4188.6</td></tr> </table>	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
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																																																		
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 183) 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 $\Sigma Int()$ und $\Sigma Prn()$, Seite 213, 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*

Im Hex-Modus:

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

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

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

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; 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).

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(Wert I) ⇒ 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(Liste I) ⇒ Liste

angle(Matrix I) ⇒ Matrix

angle() (Winkel)

Katalog > 

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

Katalog > 

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 166)

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)

Katalog > 

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 166)

LevZei=0 für Block

LevZei=2,3,...,*Len*-1, für Faktor zwei, wobei $Len = \text{length}(Liste1) = \text{length}(Liste2) = \dots = \text{length}(Liste10)$ und $Len / LevZei \in \{2,3,\dots\}$

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
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** ⇒ Wert56 56

Gibt das Ergebnis des zuletzt ausgewerteten Ausdrucks zurück.

56+4 6060+4 64**approx() (Approximieren)****Katalog >** **approx(Wert1)** ⇒ Zahlapprox($\frac{1}{3}$) 0.333333Gibt die Auswertung des Arguments ungeachtet der aktuellen Einstellung des Modus **Auto** oder **Näherung** als Dezimalwert zurück, sofern möglich.approx($\left\{\frac{1}{3}, \frac{1}{9}\right\}$) {0.333333, 0.111111}Gleichwertig damit ist die Eingabe des Arguments und Drücken von **ctrl enter**.approx($\{\sin(\pi), \cos(\pi)\}$) {0, -1}approx($[\sqrt{2}, \sqrt{3}]$) [1.41421 1.73205]approx($\left[\frac{1}{3}, \frac{1}{9}\right]$) [0.333333 0.111111]**approx(Liste1)** ⇒ Listeapprox($\{\sin(\pi), \cos(\pi)\}$) {0, -1}**approx(Matrix1)** ⇒ Matrixapprox($[\sqrt{2}, \sqrt{3}]$) [1.41421 1.73205]

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

approxFraction()**Katalog >** **Wert** ▶ **approxFraction([Tol])** ⇒ Wert $\frac{1}{2} + \frac{1}{3} + \tan(\pi)$ 0.833333**Liste** ▶ **approxFraction([Tol])** ⇒ Liste0.833333333333333 ▶ **approxFraction(5.E-14)****Matrix** ▶ **approxFraction([Tol])** ⇒ MatrixGibt die Eingabe als Bruch mit der Toleranz *Tol* zurück. Wird *tol* weggelassen, so wird die Toleranz 5.E-14 verwendet. $\frac{5}{6}$ $\{\pi, 1.5\}$ ▶ **approxFraction(5.E-14)** $\left\{\frac{5419351}{1725033}, \frac{3}{2}\right\}$ **Hinweis:** Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **@>approxFraction(...)** eintippen.

approxRational()Katalog > **approxRational**(Wert[, Tol]) \Rightarrow Wert

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

approxRational(Liste[, Tol]) \Rightarrow Liste

$$\text{approxRational}(\{0.2, 0.33, 4.125\}, 5 \cdot 10^{-14})$$
$$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$$

approxRational(Matrix[, Tol]) \Rightarrow Matrix

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

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 $\operatorname{csch}^{-1}()$, Seite 36.**arcsec()**Siehe $\sec^{-1}()$, Seite 151.**arcsech()**Siehe $\operatorname{sech}^{-1}()$, Seite 151.

arcsin()Siehe $\sin^{-1}()$, Seite 160.**arcsinh()**Siehe $\sinh^{-1}()$, Seite 162.**arctan()**Siehe $\tan^{-1}()$, Seite 173.**arctanh()**Siehe $\tanh^{-1}()$, Seite 174.**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*.

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

avgRC() (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] [,

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

Schritt) ⇒ *Matrix*

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*,*N*,*I*,*PV*,*Pmt*), [*FV*], [*PpY*],
[*CpY*], [*PmtAt*], [*WertRunden*] ⇒ *Wert*

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

bal(*NPmt*,*AmortTabelle*) ⇒ *Wert*

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

Amortisationsfunktion, die den Saldo
nach einer angegebenen Zahlung
berechnet.

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

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* und *PmtAt*
werden in der TVM-Argumentetabelle
(Seite 183) beschrieben.

bal(4,*tbl*) 1674.27

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 183) 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.

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 213.

►Base2

Ganzzahl1 ►Base2 ⇒ *Ganzzahl*

256►Base2

0b10000000

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

0h1F►Base2

0b11111

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 Binärkomplement angezeigt. Beispiel:

-1 wird angezeigt als

0hFFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 Einsen) im Binärmodus

-2⁶³ wird angezeigt als

0h8000000000000000 im Hex-Modus

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:

2⁶³ wird zu -2⁶³ und wird angezeigt als

0h8000000000000000 im Hex-Modus

0b100...000 (63 Nullen) im Binärmodus

2⁶⁴ wird zu 0 und wird angezeigt als

0h0 im Hex-Modus

0b0 im Binärmodus

-2⁶³ - 1 wird zu 2⁶³ - 1 und wird angezeigt als

0h7FFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 1's) im Binärmodus

Ganzzahl | ►Base10 ⇒ Ganzzahl

0b10011 ►Base10 19

0h1F ►Base10 31

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*

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.

Ganzzahl1 ►Base16⇒*Ganzzahl*

256►Base16	0h100
0b111100001111►Base16	0hFOF

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

Wandelt *Ganzzahl1* in eine Hexadezimalzahl um. Dual- oder Hexadezimalzahlen weisen stets das Präfix 0b bzw. 0h auf.

0b *binäre_Zahl*

0h *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

($n,p,untereGrenze,obereGrenze$)⇒*Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

binomCdf($n,p,obereGrenze$) für $P(0 \leq X \leq obereGrenze) \Rightarrow$ *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 $P(X \leq obereGrenze)$ setzen Sie *untereGrenze*=0

binomPdf()

binomPdf(n,p)⇒*Liste*

binomPdf($n,p,XWert$)⇒*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.

ceiling() (Obergrenze)Katalog > **ceiling**(*Wert1*) \Rightarrow *Wert* $\text{ceiling}(.456)$

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(*Liste1*) \Rightarrow *Liste* $\text{ceiling}(\{-3.1, 1, 2.5\})$ $\{-3., 1, 3.\}$ **ceiling**(*Matrix1*) \Rightarrow *Matrix* $\text{ceiling}\left(\begin{bmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{bmatrix}\right)$ $\begin{bmatrix} 0 & -3. \cdot i \\ 2. & 4 \end{bmatrix}$

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

centralDiff()Katalog > **centralDiff**(*Ausdr1*, *Var* [=*Wert*]
[,*Schritt*]) \Rightarrow *Ausdruck* $\text{centralDiff}(\cos(x), x) | x = \frac{\pi}{2}$

-1.

centralDiff(*Ausdr1*, *Var*
[,*Schritt*]) | *Var* = *Wert* \Rightarrow *Ausdruck***centralDiff**(*Ausdr1*, *Var* [=*Wert*]
[,*Liste*]) \Rightarrow *Liste***centralDiff**(*Liste1*, *Var* [=*Wert*]
[,*Schritt*]) \Rightarrow *Liste***centralDiff**(*Matrix1*, *Var* [=*Wert*]
[,*Schritt*]) \Rightarrow *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)

char(*Ganzzahl*) ⇒ *Zeichen*

char(38)

"&"

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 2way

χ^2 2way *BeobMatrix*

chi22way *BeobMatrix*

Berechnet eine χ^2 Testgröße auf Grundlage einer beobachteten Matrix *BeobMatrix*.

Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert.

(Seite 166.)

Informationen zu den Auswirkungen leerer Elemente in einer Matrix finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: $\text{sum}(\text{beobachtet} - \text{erwartet})^2 / \text{erwartet}$
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

$\chi^2\text{Cdf}$

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

 chi2Cdf

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

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

 $\chi^2\text{GOF}$

$\chi^2\text{GOF}$ *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 166)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: $\text{sum}((\text{beobachtet} - \text{erwartet})^2/\text{erwartet})$
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.ComplList	Liste der Chi-Quadrat-Summanden in der Testgröße

χ^2 Pdf()

Katalog > 

χ^2 Pdf(*XWert*,*FreiGrad*) \Rightarrow Zahl, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

chi2Pdf(*XWert*,*FreiGrad*) \Rightarrow Zahl, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion (Pdf) einer χ^2 -Verteilung an einem bestimmten *XWert* für die vorgegebenen Freiheitsgrade *FreiGrad*.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

ClearAZ (LöschAZ)

Katalog > 

ClearAZ

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

Löscht alle Variablen mit einem Zeichen im aktuellen Problembereich.

<i>b</i>	5
----------	---

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

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 186

<i>b</i>	"Error: Variable is not defined"
----------	----------------------------------

ClrErr (LöFehler)

Katalog > 

ClrErr

Ein Beispiel für **ClrErr** finden Sie als Beispiel 2 im Abschnitt zum Befehl **Versuche (Try)**, Seite 179.

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

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** (**ÜbgebFehler**) 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 124, und **Try**, Seite 179.

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*.

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

colDim() (Spaltendimension)

colDim(*Matrix*) \Rightarrow *Ausdruck*

Gibt die Anzahl der Spalten von *Matrix* zurück.

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

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

colNorm() (Spaltennorm)Katalog > **colNorm(Matrix)** ⇒ Ausdruck

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

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

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

conj() (Komplex Konjugierte)Katalog > **conj(Wert1)** ⇒ Wert**conj(Liste1)** ⇒ Liste**conj(Matrix1)** ⇒ Matrix

Gibt das komplex Konjugierte des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

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

constructMat()Katalog > **constructMat**

(
Ausdr
,Var1,Var2,AnzZeilen,AnzSpalten)
⇒ Matrix

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*.

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

CopyVar *Var1, Var2*

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

CopyVar *Var1., Var2.*

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

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

CopyVar <i>a,c: c(4)</i>	$\frac{1}{4}$
--------------------------	---------------

CopyVar <i>b,c: c(4)</i>	16
--------------------------	----

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 Variablennamen vereinfachen lässt, der den Regeln entspricht.

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

<i>aa.a:=45</i>	45
-----------------	----

<i>aa.b:=6.78</i>	6.78
-------------------	------

CopyVar <i>aa.,bb.</i>	Done
------------------------	------

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.

getVarInfo()	<i>aa.a</i> "NUM" "☐" 0
	<i>aa.b</i> "NUM" "☐" 0,
	<i>bb.a</i> "NUM" "☐" 0
	<i>bb.b</i> "NUM" "☐" 0

corrMat() (Korrelationsmatrix)

corrMat{*Liste1,Liste2[,...[,Liste20]]*}

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

cos() (Kosinus)

 Taste

cos(Wert1)⇒Wert

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 vorübergehend aufzuheben.

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:

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

Im Grad-Modus:

$\cos\left(\left(\frac{\pi}{4}\right)^r\right)$	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() (Kosinus)



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)



$\cos^{-1}(\text{Wert1}) \Rightarrow \text{Wert}$

Im Grad-Modus:

$\cos^{-1}(\text{Liste1}) \Rightarrow \text{Liste}$

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

$\cos^{-1}(\text{Wert1})$ gibt den Winkel zurück, dessen Kosinus *Wert1* ist.

Im Neugrad-Modus:

$\cos^{-1}(\text{Liste1})$ gibt in Form einer Liste für jedes Element aus *Liste1* 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.

$\cos^{-1}(\text{Quadratmatrix1}) \Rightarrow \text{Quadratmatrix}$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

$\cos^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$
 $\begin{bmatrix} 1.73485+0.064606 \cdot i & -1.49086+2.10514 \\ -0.725533+1.51594 \cdot i & 0.623491+0.77836 \cdot i \\ -2.08316+2.63205 \cdot i & 1.79018-1.27182 \cdot i \end{bmatrix}$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

 Taste

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

$\cosh()$ (Cosinus hyperbolicus)

Katalog > 

$\cosh(\text{Wert}) \Rightarrow \text{Wert}$

Im Grad-Modus:

$\cosh(\text{Liste}) \Rightarrow \text{Liste}$

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

$\cosh(\text{Wert})$ gibt den Cosinus hyperbolicus des Arguments zurück.

$\cosh(\text{Liste})$ gibt in Form einer Liste für jedes Element aus *Liste* den Cosinus hyperbolicus zurück.

$\cosh(\text{Quadratmatrix}) \Rightarrow \text{Quadratmatrix}$

Im Bogenmaß-Modus:

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

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

Quadratmatrix muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$\cosh^{-1}()$ (Arkuskosinus hyperbolicus)

Katalog > 

$\cosh^{-1}(\text{Wert}) \Rightarrow \text{Wert}$

$$\begin{array}{ll} \cosh^{-1}(1) & 0 \\ \cosh^{-1}(\{1,2,1,3\}) & \{0,1.37286,1.76275\} \end{array}$$

$\cosh^{-1}(\text{Liste}) \Rightarrow \text{Liste}$

$\cosh^{-1}(\text{Wert})$ gibt den inversen Cosinus hyperbolicus des Arguments zurück.

$\cosh^{-1}(\text{Liste})$ gibt in Form einer Liste für jedes Element aus *Liste* 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}
(*Quadratmatrix1*) \Rightarrow *Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

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

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

cot() (Kotangens)

 Taste

cot(*Wert1*) \Rightarrow *Wert*

Im Grad-Modus:

cot(*Liste1*) \Rightarrow *Liste*

cot(45) 1.

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

Im Neugrad-Modus:

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können $^{\circ}$, $^{\text{G}}$ oder $^{\text{r}}$ benutzen, um den Winkelmodus vorübergehend aufzuheben.

cot(50) 1.

Im Bogenmaß-Modus:

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

cot⁻¹() (Arkuskotangens)

 Taste

cot⁻¹(*Wert1*) \Rightarrow *Wert*

Im Grad-Modus:

cot⁻¹(*Liste1*) \Rightarrow *Liste*

cot⁻¹(1) 45.

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

Im Neugrad-Modus:

cot⁻¹(1) 50.

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

Im Bogenmaß-Modus:

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

 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 > 

$\coth(Wert1) \Rightarrow Wert$

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

$\coth(Liste1) \Rightarrow Liste$

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

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

$\coth^{-1}()$ (Arkuskotangens hyperbolicus)

Katalog > 

$\coth^{-1}(Wert1) \Rightarrow Wert$

$\coth^{-1}(3,5)$	0.293893
-------------------	----------

$\coth^{-1}(Liste1) \Rightarrow Liste$

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

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 `arccoath (...)` eintippen.

$\text{count}()$ (zähle)

Katalog > 

$\text{count}(Wert1 \text{ oder } Liste1$
 $[, Wert2 \text{ oder } Liste2 [, \dots]]) \Rightarrow Wert$

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

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

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

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

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.

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 241).

countIf()

countIf(Liste,Kriterien)⇒Wert

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

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.

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 241).

Hinweis: Siehe auch **sumIf()**, Seite 171, und **frequency()**, Seite 62.

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

Zählt die Anzahl der Elemente, die 3 entsprechen.

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.

cPolyRoots()

Katalog > 

cPolyRoots(*Poly*,*Var*) \Rightarrow Liste

cPolyRoots(*KoeffListe*) \Rightarrow Liste

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

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 127.

crossP() (Kreuzprodukt)

Katalog > 

crossP(*Liste1*, *Liste2*) \Rightarrow Liste

$\text{crossP}(\{0,1,2,2,-5\},\{1,-0,5,0\})$
 $\{-2,5,-5,-2,25\}$

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*) \Rightarrow Vektor

$\text{crossP}(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \begin{bmatrix} -3 & 6 & -3 \end{bmatrix})$
 $\text{crossP}(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -2 \end{bmatrix})$

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)

 Taste

csc(*Wert1*) \Rightarrow Wert

Im Grad-Modus:

csc() (Kosekans) **Taste****csc**(*Liste1*) ⇒ *Liste*

 $\text{csc}(45)$

1.41421

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

Im Neugrad-Modus:

 $\text{csc}(50)$

1.41421

Im Bogenmaß-Modus:

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

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

Im Grad-Modus:

csc⁻¹(*Liste1*) ⇒ *Liste* $\text{csc}^{-1}(1)$ 90.

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

Im Neugrad-Modus:

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

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

Im Bogenmaß-Modus:

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

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

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

 $\text{csch}(3)$

0.099822**csch**(*Liste1*) ⇒ *Liste*

 $\text{csch}(\{1, 2, 1, 4\})$

 $\{0.850918, 0.248641, 0.036644\}$

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

$\text{csch}^{-1}()$ (Inverser Kosekans hyperbolicus)

Katalog > 

$\text{csch}^{-1}(\text{Wert}) \Rightarrow \text{Wert}$

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

$\text{csch}^{-1}(\text{Liste}) \Rightarrow \text{Liste}$

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

Gibt den inversen hyperbolischen Kosekans von *Wert* oder eine Liste der inversen hyperbolischen Kosekans aller Elemente in *Liste* 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 166.)

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 241).

Ausgabevariable	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

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

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

cumulativeSum(Matrix1) ⇒ Matrix

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

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 241).

Cycle (Zyklus)

Katalog > 

Cycle (Zyklus)

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

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

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

►Cylind (Zylindervektor)

Katalog > 

Vektor ►Cylind

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

Zeigt den Zeilen- oder Spaltenvektor in Zylinderkoordinaten $[r, \angle\theta, z]$ an.

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

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

D

dbd()

Katalog > 

dbd(Datum1,Datum2)⇒Wert

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

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.

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

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*

Im Grad-Modus:

Liste ►DD ⇒ *Liste*

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

Matrix ►DD ⇒ *Matrix*

$(45^\circ 22' 14.3'')$ ►DD	45.3706°
-----------------------------	-----------------

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

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

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 Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

►Decimal (Dezimal)

Wert ►Decimal ⇒ *Wert*

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

Liste ►Decimal ⇒ *Wert*

Matrix ►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.

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.

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

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**).

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

```

```
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 ΔList(), Seite 92.

DelVar

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

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

DelVar *Var*.

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

Löscht die angegebene Variable oder Variablengruppe im Speicher.

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

$(a+2)^2$	"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 186.

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>	<table border="1"><tr><td><i>aa.a</i></td><td>"NUM"</td><td>"{:}"</td></tr><tr><td><i>aa.b</i></td><td>"NUM"</td><td>"{:}"</td></tr><tr><td><i>aa.c</i></td><td>"NUM"</td><td>"{:}"</td></tr></table>	<i>aa.a</i>	"NUM"	"{:}"	<i>aa.b</i>	"NUM"	"{:}"	<i>aa.c</i>	"NUM"	"{:}"
<i>aa.a</i>	"NUM"	"{:}"								
<i>aa.b</i>	"NUM"	"{:}"								
<i>aa.c</i>	"NUM"	"{:}"								
<i>DelVar aa.</i>	<i>Done</i>									
<i>getVarInfo()</i>	"NONE"									

delVoid()

Katalog > 

delVoid(*Listel*) \Rightarrow *Liste*

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

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

Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

det() (Matrixdeterminante)

Katalog > 



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

<i>det</i> $\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	-2
------------------------------------------------------------------------	----

Gibt die Determinante von *Quadratmatrix* zurück.

$\begin{bmatrix} 1.\text{E}20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow \text{mat1}$	$\begin{bmatrix} 1.\text{E}20 & 1 \\ 0 & 1 \end{bmatrix}$
<i>det</i> (<i>mat1</i>)	0
<i>det</i> (<i>mat1</i> ,.1)	1. $\text{E}20$

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. Anderenfalls wird *Toleranz* ignoriert.

- Wenn Sie   verwenden oder den Modus **Autom. oder Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Toleranz* weggelassen oder nicht

det() (Matrixdeterminante)

Katalog > 

verwendet, so wird die Standardtoleranz folgendermaßen berechnet:

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

diag() (Matrixdiagonale)

Katalog > 

diag(Liste) ⇒ Matrix

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

diag(Zeilenmatrix) ⇒ Matrix

diag(Spaltenmatrix) ⇒ Matrix

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

diag(Quadratmatrix) ⇒ Zeilenmatrix

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

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

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

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

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

dim(String) ⇒ Ganzzahl

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

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

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
```

chars(240,243)	
240	ø
241	ñ
242	ò
243	ó
<i>Done</i>	

DispAt

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.

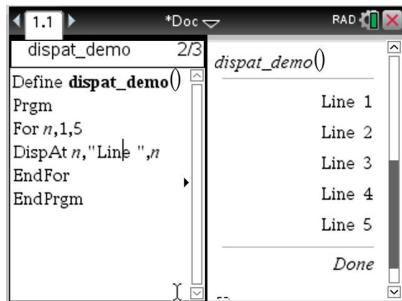
Beachten Sie, dass die Zeilennummer nicht für den gesamten Bildschirm gedacht ist, sondern für den Bereich unmittelbar nach dem Befehl/Programm.

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

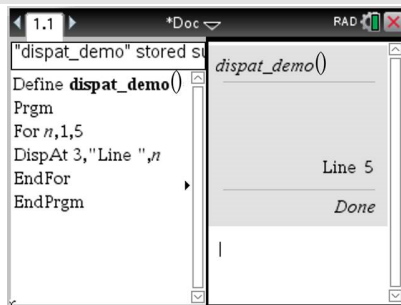
DispAt und **Disp** können im gleichen Programm verwendet werden.

DispAt

Beispiel



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:

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

Fehlermeldungen:

Fehlermeldung	Beschreibung
DispAt Zeilennummer muss zwischen 1 und 8 liegen	Term bewertet die Zeilennummer außerhalb des Bereichs 1-8 (inklusive)
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 wie Disp.
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)

Zahl ►DMS

Im Grad-Modus:

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

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 (GGGGG°MM'SS.ss"). Siehe °, ', " (Seite 218) 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(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**e^()** Taste**e^(Wert1)** ⇒ Werte¹ 2.71828

Gibt e hoch Wert1 zurück.

e^{3²} 8103.08**Hinweis:** Siehe auch Vorlage **e Exponent**, Seite 2.**Hinweis:** Das Drücken von  zum Anzeigen von e^ ist nicht das gleiche wie das Drücken von  auf der Tastatur.Sie können eine komplexe Zahl in der polaren Form $re^{i\theta}$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.**e^(Liste1)** ⇒ Liste

e {1,1,.05} {2.71828,2.71828,1.64872}

Gibt e hoch jedes Element der Liste1 zurück.

e^(Quadratmatrix1) ⇒ QuadratmatrixErgibt 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()**.

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

*Quadratmatrix*1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

eff()

Katalog >

eff(*Nominalzinssatz*, *CpY*) \Rightarrow 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 115.

eigVc() (Eigenvektor)

Katalog >

eigVc(*Quadratmatrix*) \Rightarrow Matrix

Im Komplex-Formatmodus "kartesisch":

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.

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

$$\text{eigVc}(m1) \begin{bmatrix} -0.800906 & 0.767947 & (\\ 0.484029 & 0.573804+0.052258 \cdot i & 0.57338 \\ 0.352512 & 0.262687+0.096286 \cdot i & 0.2626 \end{bmatrix}$$

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

eigVI() (Eigenwert)**Katalog >** **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":

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

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

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

Else**Siehe If, Seite 76.****Elseif****Katalog >** **If Boolescher Ausdr1 Then***Block1***Elseif Boolescher Ausdr2 Then***Block2*

⋮

Elseif 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 $x < 0$ ThenReturn $-x$ Elseif $x \geq 0$ and $x \neq 10$ ThenReturn x Elseif $x = 10$ Then

Return 3

Endif

EndFunc

*Done***EndFor****Siehe For, Seite 59.****EndFunc****Siehe Func, Seite 64.**

euler ()

Katalog > 

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

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

euler(*AusdrListe*, *Var*, *ListeAbhVar*, {*Var0*, *VarMax*}, *ListeAbhVar0*, *VarSchritt* [, *eulerSchritt*]) ⇒*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.

Differentialgleichung:

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

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

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

Gleichungssystem:

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

mit *y1*(0)=2 und *y2*(0)=5

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.

{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 $\text{sign}(\text{VarSchritt}) = \text{sign}(\text{VarMax} - \text{Var0})$ und Lösungen an $\text{Var0} + i \cdot \text{VarSchritt}$ für alle $i=0,1,2,\dots$ zurückgegeben werden, sodass $\text{Var0} + i \cdot \text{VarSchritt}$ in $[\text{var0}, \text{VarMax}]$ 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 $\text{VarSchritt} / \text{eulerSchritt}$.

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

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

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

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

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.

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 66), **GetStr** (Seite 73) und **Send** (Seite 152).

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

Setzen Sie das blaue Element auf AUS zurück.

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

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

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

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

Exit 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	<i>Done</i>
Local $temp,i$	
$0 \rightarrow temp$	
For $i,1,100,1$	
$temp+i \rightarrow temp$	
If $temp>20$ Then	
Exit	
EndIf	
EndFor	
EndFunc	
<hr/>	
$g()$	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 $rei \theta$ 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
<hr/>	
e^{3^2}	8103.08

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

$e \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>782.209</td> <td>559.617</td> <td>456.509</td> </tr> <tr> <td>680.546</td> <td>488.795</td> <td>396.521</td> </tr> <tr> <td>524.929</td> <td>371.222</td> <td>307.879</td> </tr> </table>	782.209	559.617	456.509	680.546	488.795	396.521	524.929	371.222	307.879
782.209	559.617	456.509								
680.546	488.795	396.521								
524.929	371.222	307.879								

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)

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 der Variablen *stat.results* gespeichert. (Seite 166.)

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 241).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot (b)^x$
stat.a, stat.b	Regressionskoeffizienten

Ausgabevariable	Beschreibung
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten (x , $\ln(y)$)
stat.Resid	Mit dem exponentiellen Modell verknüpfte Residuen
stat.ResidTrans	Residuum für die lineare Anpassung der transformierten Daten.
stat.XReg	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
stat.YReg	Liste der Datenpunkte in der modifizierten Y List, 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.

<code>factor(152417172689)</code>	123457·1234577
<code>isPrime(152417172689)</code>	false

So halten Sie eine Berechnung manuell an:

- **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.

factor() (Faktorisiere)

Katalog >

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.

F Cdf()

Katalog >

F Cdf

(
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.

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

Fill (Füllen)

Katalog > 

Fill *Zahl*, *ListeVar* ⇒ *Liste*

{1,2,3,4,5} → *alist*

{1,2,3,4,5}

Ersetzt jedes Element in der Variablen *ListeVar* durch *Zahl*.

Fill 1.01,*alist*

Done

alist

{1.01,1.01,1.01,1.01,1.01}

ListeVar muss bereits vorhanden sein.

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 166.)

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 241).

Ausgabevariable	Beschreibung
stat.MinX	Minimum der x-Werte
stat.Q1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q3X	3. Quartil von x

Ausgabevariable	Beschreibung
stat.MaxX	Maximum der x-Werte

floor() (Untergrenze)

Katalog > 

floor(Wert1) ⇒ *Ganzzahl*

$\text{floor}(-2.14)$ -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(Liste1) ⇒ *Liste*

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

floor(Matrix1) ⇒ *Matrix*

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

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()**.

For

Katalog > 

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.

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

$g()$ 5050

Hinweis zum Eingeben des Beispiels:

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

format() (Format)

format(Wert[, FormatString])⇒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)Katalog > **fPart**(*AusdrI*) ⇒ *Ausdruck*

 $fPart(-1.234)$ -0.234

fPart(*ListeI*) ⇒ *Liste*

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

fPart(*MatrixI*) ⇒ *Matrix*

Gibt den Bruchanteil des Arguments zurück.

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

Das Argument kann eine reelle oder eine komplexe Zahl sein.

FPdf()Katalog > **F**Pdf

(

XWert, *FreiGradZähler*, *FreiGradNenner*) ⇒ *Zahl*,wenn *XWert* eine Zahl ist, *Liste*, wenn*XWert* eine Liste ist**FPdf**

(

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(*ListeI*, *HäufGanzzahlListe*) ⇒ *Liste*

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

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

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

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.

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 241).

frequency() (Häufigkeit)

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

<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.

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 241).

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

Hinweis: Siehe auch `countf()`, Seite 33.

FTest_2Samp (Zwei-Stichproben F-Test)

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

`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 166.)

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 241).

Ausgabevariable	Beschreibung
Statistik.F	Berechnete \hat{U} Statistik für die Datenfolge

Ausgabevariable	Beschreibung
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.dfNumer	Freiheitsgrade des Zählers = $n_1 - 1$
stat.dfDenom	Freiheitsgrade des Nenners = $n_2 - 1$
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.x1_bar stat.x2_bar	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Stichprobenumfang

Func

Katalog >

Func

Block

EndFunc

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.

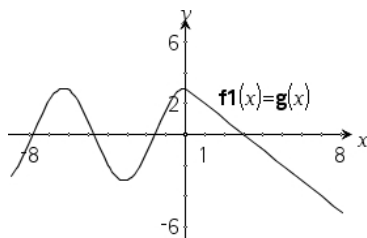
Hinweis zum Eingeben des Beispiels:

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

Definieren Sie eine stückweise definierte Funktion:

```
Define g(x)=Func Done
  If x<0 Then
    Return 3*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

gcd() (Größter gemeinsamer Teiler)

Katalog > 

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*

$$\text{gcd}(\{12,14,16\},\{9,7,5\}) \quad \{3,7,1\}$$

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

gcd(Matrix1, Matrix2) ⇒ *Matrix*

$$\text{gcd}\left(\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}\right) \quad \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

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

geomCdf()

Katalog > 

geomCdf

(p,untereGrenze,obereGrenze) ⇒ *Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

geomCdf(p,obereGrenze) für $P(1 \leq X \leq \text{obereGrenze})$ ⇒ *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()

Katalog > 

geomPdf(p,XWert) ⇒ *Zahl*, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeit an einem X Wert, 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 –
- 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.

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.

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.

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

In der zweiten Syntax 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 $Fkt(arg1, \dots, argn) =$
empfänger 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 152.

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

getKey()

Katalog > 

getKey([01]) ⇒ returnString

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

Beispiel:

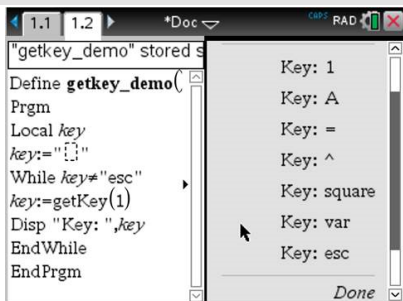
- gedrückteTaste := **getKey()** gibt eine Taste oder eine leere Zeichenkette zurück, wenn keine Taste gedrückt wurde. Dieser

getKey()

Beispiel:

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.



Handhabung von Tastenbetätigungen:

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

Handheld/Emulatortaste	Desktop	Rückgabewert
Vorlagen	-	„Vorlage“
Katalog	-	„cat“
^	^	"^"
X^2	-	„Quadrat“
/ (Divisionstaste)	/	"/"
* (Multiplikationstaste)	*	"*"
e^x	-	„Ausdr“
10^x	-	„10power“
+	+	"+"
-	-	"_"
(("("
))	")"
.	.	". "
(-)	-	„-“ (Negativ-Zeichen)
Eingabetaste	Eingabetaste	„Eingabe“
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
,	,	" , "

Handheld/Emulatortaste	Desktop	Rückgabewert
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 'Prüfungsmodus 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"

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 96, und **unLock**, Seite 186.

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

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.

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 155.

<code>getMode(0)</code>	<code>{ 1,7,2,1,3,1,4,1,5,1,6,1,7,1 }</code>
<code>getMode(1)</code>	<code>7</code>
<code>getMode(7)</code>	<code>1</code>

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)

Katalog > 

getNum(*Bruch1*) ⇒ Wert

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

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

GetStr

Hub-Menü

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

Zum Beispiel siehe **Get**.

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

Programmierbefehl: Verhält sich genauso wie der Befehl **Get**, der aber gefundene 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 66 und **Send**, Seite 152.

getType()

Katalog > 

getType(*var*) ⇒ *String*

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

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

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

getVarInfo()

Katalog > 

getVarInfo() ⇒ *Matrix* oder *String*

getVarInfo(*BiblioNameString*) ⇒ *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(*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.

getVarInfo()	"NONE"												
Define <i>x</i> =5	Done												
Lock <i>x</i>	Done												
Define LibPriv <i>y</i> ={1,2,3}	Done												
Define LibPub <i>z</i> (<i>x</i>)=3· <i>x</i> ² - <i>x</i>	Done												
getVarInfo()	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td><i>x</i></td><td>"NUM"</td><td>"{ }"</td><td>1</td></tr> <tr><td><i>y</i></td><td>"LIST"</td><td>"LibPriv"</td><td>0</td></tr> <tr><td><i>z</i></td><td>"FUNC"</td><td>"LibPub"</td><td>0</td></tr> </table>	<i>x</i>	"NUM"	"{ }"	1	<i>y</i>	"LIST"	"LibPriv"	0	<i>z</i>	"FUNC"	"LibPub"	0
<i>x</i>	"NUM"	"{ }"	1										
<i>y</i>	"LIST"	"LibPriv"	0										
<i>z</i>	"FUNC"	"LibPub"	0										
getVarInfo(<i>tmp3</i>)	"Error: Argument must be a string"												
getVarInfo("tmp3")	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td><i>volcyl2</i></td><td>"NONE"</td><td>"LibPub"</td><td>0</td></tr> </table>	<i>volcyl2</i>	"NONE"	"LibPub"	0								
<i>volcyl2</i>	"NONE"	"LibPub"	0										

getVarInfo()

Katalog > 

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.

$a:=1$	1												
$b:=[1\ 2]$	$[1\ 2]$												
$c:=[1\ 3\ 7]$	$[1\ 3\ 7]$												
$vs:=getVarInfo()$	<table border="1"> <tr> <td><i>a</i></td> <td>"NUM"</td> <td>"[]"</td> <td>0</td> </tr> <tr> <td><i>b</i></td> <td>"MAT"</td> <td>"[]"</td> <td>0</td> </tr> <tr> <td><i>c</i></td> <td>"MAT"</td> <td>"[]"</td> <td>0</td> </tr> </table>	<i>a</i>	"NUM"	"[]"	0	<i>b</i>	"MAT"	"[]"	0	<i>c</i>	"MAT"	"[]"	0
<i>a</i>	"NUM"	"[]"	0										
<i>b</i>	"MAT"	"[]"	0										
<i>c</i>	"MAT"	"[]"	0										
$vs[1]$	$[1\ "NUM"\ "[]"\ 0]$												
$vs[1,1]$	1												
$vs[2]$	"Error: Invalid list or matrix"												
$vs[2,1]$	$[1\ 2]$												

Goto (Gehe zu)

Katalog > 

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 Beispiels:

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

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

►Grad (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:

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

Im Bogenmaß-Modus:

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

identity()Katalog > **identity**(*Ganze Zahl*) ⇒ *Matrix*

Gibt die Einheitsmatrix mit der Dimension *Ganzzahl* zurück.

Ganzzahl muss eine positive ganze Zahl sein.

identity(4)	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1

IfKatalog > 

If *BooleanExpr*
Anweisungen

If *BooleanExpr* **Then**
Block

EndIf

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.

Define $g(x)$ =Func	<i>Done</i>
If $x < 0$ Then	
Return x^2	
EndIf	
EndFunc	
$g(-2)$	4

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

Done

If $x < 0$ ThenReturn $-x$

Else

Return x

EndIf

EndFunc

 $g(12)$

12

 $g(-12)$

12

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

Return 5

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

Return 3

EndIf

EndFunc

Done

 $g(-4)$

4

 $g(10)$

3

ifFn()Katalog > 

ifFn(BoolescherAusdruck, Wert_wenn_wahr [, Wert_wenn_falsch [, Wert_wenn_unbekannt]]) \Rightarrow *Ausdruck, Liste oder Matrix*

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.

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.

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

- 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.

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()

imag(ValueI) ⇒ Wert

Gibt den Imaginärteil des Arguments zurück.

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

imag(ListI) ⇒ Liste

Gibt eine Liste der Imaginärteile der Elemente zurück.

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

imag(MatrixI) ⇒ Matrix

Gibt eine Matrix der Imaginärteile der Elemente zurück.

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

inString()Katalog > 

inString(*Quellstring*, *Teilstring*[, *Start*])
 ⇒ *Ganzzahl*

<code>inString("Hello there", "the")</code>	7
<code>inString("ABCEFG", "D")</code>	0

Gibt 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(*List1*) ⇒ *Liste*
int(*Matrix1*) ⇒ *Matrix*

<code>int(-2.5)</code>	-3.
<code>int([-1.234 0 0.37])</code>	[-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.

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*) ⇒ *Ganzzahl*
intDiv(*Liste1*, *Liste2*) ⇒ *Liste*
intDiv(*Matrix1*, *Matrix2*) ⇒ *Matrix*

<code>intDiv(-7,2)</code>	-3
<code>intDiv(4,5)</code>	0
<code>intDiv({12,-14,-16},{5,4,-3})</code>	{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 ($Argument1 \div Argument2$) zurückgegeben.

Interpolieren ()

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.undef**.

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

$xWert$ kann eine Zahl oder eine Zahlenliste sein.

Differentialgleichung:

$$y' = -3 \cdot y + 6 \cdot t + 5 \text{ und } y(0) = 5$$

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

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

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

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

$$xvalueList := seq(i, i, 0, 10, 0.5)$$

$$\{0, 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6., 6.5, \}$$

$$xlist := mat \blacktriangleright list(rk[1])$$

$$\{0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.\}$$

$$ylist := mat \blacktriangleright list(rk[2])$$

$$\{5., 3.19499, 5.00394, 6.99957, 9.00593, 10.9978\}$$

$$yprimelist := -3 \cdot y + 6 \cdot t + 5 | y = ylist \text{ and } t = xlist$$

$$\{-10., 1.41503, 1.98819, 2.00129, 1.98221, 2.006\}$$

$$interpolate(xvalueList, xlist, ylist, yprimelist)$$

$$\{5., 2.67062, 3.19499, 4.02782, 5.00394, 6.00011\}$$

invχ²()

invχ²(*Fläche*, *FreiGrad*)

invChi2(*Fläche*, *FreiGrad*)

inv χ^2 ()Katalog > 

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*) \Rightarrow *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.

OutputForm=1, gibt Ergebnis als Matrix 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?

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

invBinomN()

Katalog > 

invBinomN(CumulativeProb, Prob, NumSuccess, OutputForm) \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()

Katalog > 

iPart(Zahl) \Rightarrow Ganzzahl

iPart(Liste1) \Rightarrow Liste

iPart(Matrix1) \Rightarrow Matrix

Gibt den ganzzahligen Teil des Arguments zurück.

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

iPart()

Katalog > 

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

Das Argument kann eine reelle oder eine komplexe Zahl sein.

irr()

Katalog > 

$\text{irr}(CF0, CFListe [, CFFreq]) \Rightarrow Wert$

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 $CFListe$ ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzzahlen < 10.000 sein.

Hinweis: Siehe auch `mirr()`, Seite 106.

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

isPrime()

Katalog > 

$\text{isPrime}(Zahl) \Rightarrow \text{Boolescher konstanter Ausdruck}$

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

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

$\text{isPrime}(5)$	true
$\text{isPrime}(6)$	false

Funktion zum Auffinden der nächsten Primzahl nach einer angegebenen Zahl:

isPrime()

Katalog > 

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) ⇒ Boolescher konstanter Ausdruck

isVoid(Ausdruck) ⇒ Boolescher konstanter Ausdruck

isVoid(Liste) ⇒ Liste Boolescher konstanter Ausdrücke

$a := _$	$_$
isVoid(a)	true
isVoid($\{1, _, 3\}$)	{ 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 Seite 241.

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 Variablennamen.

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

Hinweis zum Eingeben des Beispiels:

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

lcm() (Kleinstes gemeinsames Vielfaches)

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

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

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

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*]) ⇒ 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")
┌──────────┬──────────┬──────────┬──────────┬──────────┬──────────┐
│ clearmat  │ "FUNC"   │ "LibPub  │ "         │ "         │ "         │
│ gauss1    │ "PRGM"   │ "LibPriv │ "         │ "         │ "         │
│ gauss2    │ "FUNC"   │ "LibPub  │ "         │ "         │ "         │
└──────────┴──────────┴──────────┴──────────┴──────────┴──────────┘
```

```
libShortcut("linalg2", "la")
      { la.clearmat, la.gauss2 }
```

```
libShortcut("linalg2", "la", 1)
      { la.clearmat, la.gauss1, la.gauss2 }
```

LinRegBx

LinRegBx *X*, *Y*, [*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 166.)

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 241).

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 X -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.YReg	Liste der Datenpunkte in der modifizierten Y -Liste, die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
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 166.)

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 241).

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 X -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.YReg	Liste der Datenpunkte in der modifizierten Y -Liste, die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>

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 166.)

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 241).

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

Ausgabevariable	Beschreibung
stat.Resid	Residuen von der Regression

Nur für Steigung

Ausgabevariable	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

Ausgabevariable	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.ŷ	$a + b \cdot X\text{Wert}$

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 die Nullhypothese ($H_0: \beta = \rho = 0$) untersucht wird.

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

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

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

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 166.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot x$
stat.t	<i>t</i> -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**(*SystemLinearerG1*, *Var1*, *Var2*, ...) ⇒ *Liste*

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

linSolve(*LineareG11* and *LineareG12* and ..., *Var1*, *Var2*, ...) ⇒ *Liste*

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

linSolve({*LineareG11*, *LineareG12*, ...}, *Var1*, *Var2*, ...) ⇒ *Liste*

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

linSolve(*SystemLinearerG1*, {*Var1*, *Var2*, ...}) ⇒ *Liste*

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

linSolve(*LineareG11* and *LineareG12* and ..., {*Var1*, *Var2*, ...}) ⇒ *Liste***linSolve**({*LineareG11*, *LineareG12*, ...}, {*Var1*, *Var2*, ...}) ⇒ *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. Anderenfalls tritt ein Argumentfehler auf.

Die Auswertung von **linSolve**(*x*=1 and *x*=2,*x*) führt beispielsweise zu dem Ergebnis "Argumentfehler" .**Δlist() (Listendifferenz)**Katalog > **Δlist**(*Liste1*) ⇒ *Liste*

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

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)

Katalog > 

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.

<code>list▶mat({1,2,3})</code>	$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
<code>list▶mat({1,2,3,4,5},2)</code>	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix}$

ln() (Natürlicher Logarithmus)

  **Tasten**

ln(*Wert1*) ⇒ *Wert*

<code>ln(2.)</code>	0.693147
---------------------	----------

ln(*Liste1*) ⇒ *Liste*

Gibt den natürlichen Logarithmus des Arguments zurück.

Bei Komplex-Formatmodus reell:

Gibt für eine Liste die natürlichen Logarithmen der einzelnen Elemente zurück.

<code>ln({-3,1.2,5})</code>	"Error: Non-real calculation"
-----------------------------	-------------------------------

Bei Komplex-Formatmodus kartesisch:

<code>ln({-3,1.2,5})</code>	$\{1.09861+3.14159 \cdot i, 0.182322, 1.60944\}$
-----------------------------	--------------------------------------------------

ln(*Quadratmatrix1*) ⇒ *Quadratmatrix*

Ergibt den natürlichen Matrix-Logarithmus von *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung des natürlichen Logarithmus jedes einzelnen Elements. Näheres zum Berechnungsverfahren finden Sie im Abschnitt **cos()**.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

*Quadratmatrix*1 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.

LnReg

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 166.)

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 241).

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 ($\ln(x)$, <i>y</i>)

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 verwendet wurde</i>
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 verwendet wurde</i>
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 <i>rollcount()</i> =Func	
Local <i>i</i>	
1 → <i>i</i>	
Loop	
If randInt(1,6)=randInt(1,6)	
Goto <i>end</i>	
<i>i</i> +1 → <i>i</i>	
EndLoop	
Lbl <i>end</i>	
Return <i>i</i>	
EndFunc	
	<i>Done</i>
<i>rollcount()</i>	16
<i>rollcount()</i>	3

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

a:=65 65

Lock *Var*.

Lock *a* Done

Sperret die angegebenen Variablen bzw. die Variablengruppe. Gesperrte Variablen können nicht geändert oder gelöscht werden.

getLockInfo(*a*) 1

a:=75 "Error: Variable is locked."

DelVar *a* "Error: Variable is locked."

Unlock *a* Done

Die Systemvariable *Ans* können Sie nicht sperren oder entsperren, ebenso können Sie die Systemvariablengruppen *stat*. oder *tvm*. nicht sperren.

a:=75 75

DelVar *a* Done

Hinweis: Der Befehl **Sperren (Lock)** löscht den Rückgängig/Wiederholen-Verlauf, wenn er für nicht gesperrte Variablen verwendet wird.

Siehe **unLock, Seite 186, und getLockInfo(), Seite 72.**

log() (Logarithmus)

Tasten

log(*Wert1* [, *Wert2*]) ⇒ *Wert*

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

log(*Liste1* [, *Wert2*]) ⇒ *Liste*

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

Gibt für den Logarithmus des Arguments zur Basis *Ausdr2* zurück.

$\log_{\frac{1}{3}} (10) - \log_{\frac{1}{3}} (5)$ 0.63093

Hinweis: Siehe auch **Vorlage Logarithmus, Seite 2.**

Bei Komplex-Formatmodus reell:

Gibt bei einer Liste den Logarithmus der Elemente zur Basis *Wert2* zurück.

$\log_{10} (\{-3,1.2,5\})$
"Error: Non-real calculation"

Wenn *Wert* weggelassen wird, wird 10 als Basis verwendet.

Bei Komplex-Formatmodus kartesisch:

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

log(*Quadratmatrix1* [, *Zahl2*]) ⇒ *Quadratmatrix*

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()**.

$$\log_{10} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 0.795387+0.753438 \cdot i & 0.003993-0.6474 \cdot i \\ 0.194895-0.315095 \cdot i & 0.462485+0.2707 \cdot i \\ -0.115909-0.904706 \cdot i & 0.488304+0.7774 \cdot i \end{pmatrix}$$

Quadratmatrix1 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.

Wenn das Basisargument weggelassen wird, wird 10 als Basis verwendet.

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 166.)

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 241).

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 verwendet wurde</i>
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 verwendet wurde</i>
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LogisticD

LogisticD *X*, *Y* [, [*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 166.)

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 241).

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

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.

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
```

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

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 Zeilentauch dokumentiert ist) in *pMatrix*.

$$lMatrix \cdot uMatrix = pMatrix \cdot Matrix$$

Sie haben die Option, dass jedes Matricelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommalelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Anderenfalls wird *Tol* ignoriert.

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

- 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.

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.

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

<code>mat>list([1 2 3])</code>	<code>{1,2,3}</code>
<code>[1 2 3] → m1</code>	<code>[1 2 3]</code>
<code>[4 5 6]</code>	<code>[4 5 6]</code>
<code>mat>list(m1)</code>	<code>{1,2,3,4,5,6}</code>

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.

max(Liste) ⇒ *Ausdruck*

<code>max(2.3,1.4)</code>	<code>2.3</code>
<code>max({1,2},{-4,3})</code>	<code>{1,3}</code>

<code>max({0,1,-7,1.3,0.5})</code>	<code>1.3</code>
------------------------------------	------------------

max() (Maximum)

Katalog > 

Gibt das größte Element von *Liste* zurück.

max(*MatrixI*) \Rightarrow *Matrix*

Gibt einen Zeilenvektor zurück, der das größte Element jeder Spalte von *MatrixI* enthält.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

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

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

mean() (Mittelwert)

Katalog > 

mean(*Liste*,
Häufigkeitsliste) \Rightarrow *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*)
 \Rightarrow *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 241).

$$\begin{array}{l} \text{mean}\{0.2,0.1,-0.3,0.4\} \quad 0.26 \\ \text{mean}\{1,2,3\},\{3,2,1\} \quad \frac{5}{3} \end{array}$$

Im Vektorformat kartesisch:

$$\text{mean}\left(\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}\right) \quad \left[-0.133333 \ 0.833333\right]$$

$$\text{mean}\left(\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & \frac{-1}{2} \end{bmatrix}\right) \quad \left[\frac{-2}{15} \ \frac{5}{6}\right]$$

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

median() (Median)

Katalog > 

median(*Liste*, *freqList*) \Rightarrow *Ausdruck*

Gibt den Medianwert der Elemente in *Liste* zurück.

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

Jedes *freqList*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

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

Gibt einen Zeilenvektor zurück, der die Medianwerte der einzelnen Spalten von *Matrix1* enthält.

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

Jedes *freqMatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

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 241).

MedMed

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

Berechnet die Median-Median-Linie = $(m \cdot x + b)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 166.)

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 Kategorie-codes. Nur solche Datenelemente, deren Kategorie-code 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 241).

Ausgabevariable	Beschreibung
stat.RegEqn	Median-Median-Linien-Gleichung: $m \cdot x + b$
stat.m, stat.b	Modellkoeffizienten
stat.Resid	Residuen von der Median-Median-Linie
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 verwendet wurde</i>
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 verwendet wurde</i>
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

mid() (Teil-String)

mid(Quellstring, Start[, Anzahl]) ⇒ 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() (Teil-String)

Katalog > 

mid(Quellliste, Start [, Anzahl])⇒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(QuellstringListe, Start[, Anzahl])⇒Liste

Gibt *Anzahl* Strings aus der Stringliste *QuellstringListe* ab dem Element mit der Nummer *Start* zurück.

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

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

min() (Minimum)

Katalog > 

min(Wert1, Wert2)⇒Ausdruck

$\text{min}(2.3,1.4)$	1.4
-----------------------	-----

min(Liste1, Liste2)⇒Liste

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

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

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

Gibt das kleinste Element von *Liste* zurück.

min(Matrix1)⇒Matrix

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

Gibt einen Zeilenvektor zurück, der das kleinste Element jeder Spalte von *Matrix1* enthält.

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

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 Ganzzahlen < 10.000 sein.

Hinweis: Siehe auch **irr()**, Seite 83.

mod() (Modulo)

mod(Wert1, Wert2) ⇒ Ausdruck

$mod(7, 0)$	7
-------------	---

mod(Liste1, Liste2) ⇒ Liste

$mod(7, 3)$	1
-------------	---

mod(Matrix1, Matrix2) ⇒ Matrix

$mod(-7, 3)$	2
--------------	---

Gibt das erste Argument modulo das zweite Argument gemäß der folgenden Identitäten zurück:

$mod(7, -3)$	-2
--------------	----

$mod(-7, -3)$	-1
---------------	----

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

$$mod(x, 0) = x$$

$$mod(x, y) = x - y \text{ floor}(x/y)$$

mod() (Modulo)

Katalog > 

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 141

mRow() (Matrixzeilenoperation)

Katalog > 

`mRow(Zahl, Matrix1, Index) ⇒ Matrix`

Gibt eine Kopie von *Matrix1* zurück, in der jedes Element der Zeile *Index* von *Matrix1* mit *Zahl* multipliziert ist.

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

mRowAdd() (Matrixzeilenaddition)

Katalog > 

`mRowAdd(Zahl, Matrix1, Index1, Index2) ⇒ Matrix`

Gibt eine Kopie von *Matrix1* zurück, wobei jedes Element in Zeile *Index2* von *Matrix1* ersetzt wird durch:

Zahl × Zeile *Index1* + Zeile *Index2*

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

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 166.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	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	Multiplres Bestimmtheitsmaß
stat.yList	$\hat{y}List = b_0+b_1 \cdot x_1+ \dots$
stat.Resid	Residuen von der Regression

MultRegIntervals

Katalog > 

MultRegIntervals $Y, XI[,X2[,X3,...$
 $[,X10]]], XWertListe[,KNiveau]$

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 166.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.y	Eine Punktschätzung: $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ 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, stat.UpperrPred	Vorhersageintervall für eine einzelne Beobachtung
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage

Ausgabevariable	Beschreibung
stat.bList	Liste der Regressionskoeffizienten, {b0,b1,b2,...}
stat.Resid	Residuen von der Regression

MultRegTests

Katalog > 

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 166.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

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	Multipl. 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

Ausgabevariable	Beschreibung
stat.bList	{b0,b1,...} 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.yList	$\hat{y}List = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Residuen von der Regression
stat.sResid	Standardisierte Residuen; wird durch Division eines Residuums 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)

N

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* ⇒ *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) ⇒ *Ausdruck*

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).

<i>nCr(z,3)</i> ; z=5	10
<i>nCr(z,3)</i> ; z=6	20

nCr(Wert, 0) ⇒ 1

nCr(Wert, negGanzzahl) ⇒ 0

nCr(Wert, posGanzzahl) ⇒ $Wert \cdot (Wert-1) \dots (Wert-posGanzzahl+1) / posGanzzahl!$

nCr(Wert, keineGanzzahl) ⇒ *Ausdruck!*
 ((
Wert
 -keineGanzzahl!) · keineGanzzahl!)

nCr(Liste1, Liste2) ⇒ *Liste*

<i>nCr</i> {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.

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

nDerivative()

nDerivative(Ausdr1, Var=Wert [, Ordnung])⇒Wert

nDerivative(Ausdr1, Var[, Ordnung] | Var=Wert)⇒Wert

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.

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.

nDerivative(x , x=1)	1
nDerivative(x , x) x=0	undef
nDerivative(√x-1, x) x=1	undef

nDerivative(x·(x ² +x) ^{1/3} , x, 1) x=0	undef
centralDiff(x·(x ² +x) ^{1/3} , x) x=0	0.000033

nDerivative()

Katalog > 

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.

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

nfMax(Ausdr, Var, UntereGrenze)⇒Wert

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

nfMax(Ausdr, Var, UntereGrenze, ObereGrenze)⇒Wert

nfMax(Ausdr, Var) | UntereGrenze ≤ Var ≤ ObereGrenze ⇒ Wert

nfMax() (Numerisches Funktionsmaximum)

Katalog > 

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.

nfMin() (Numerisches Funktionsminimum)

Katalog > 

$\text{nfMin}(\textit{Ausdr}, \textit{Var}) \Rightarrow \textit{Wert}$

$\text{nfMin}(\textit{Ausdr}, \textit{Var}, \textit{UntereGrenze}) \Rightarrow \textit{Wert}$

$\text{nfMin}(\textit{Ausdr}, \textit{Var}, \textit{UntereGrenze}, \textit{ObereGrenze}) \Rightarrow \textit{Wert}$

$\text{nfMin}(\textit{Ausdr}, \textit{Var} \mid \textit{UntereGrenze} \leq \textit{Var} \leq \textit{ObereGrenze}) \Rightarrow \textit{Wert}$

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.

$\text{nfMin}(x^2 + 2 \cdot x + 5, x)$	-1.
$\text{nfMin}(0.5 \cdot x^3 - x - 2, x, -5.5)$	-5.

nInt() (Numerisches Integral)

Katalog > 

$\text{nInt}(\textit{Ausdr1}, \textit{Var}, \textit{Untere}, \textit{Obere}) \Rightarrow \textit{Ausdruck}$

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

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 $\int(\textit{Ausdr1}, \textit{Var}, \textit{Untere}, \textit{Obere})$ zurück. Diese Näherung ist der gewichtete Durchschnitt von Stichprobenwerten des Integranden im Intervall $\textit{Untere} < \textit{Var} < \textit{Obere}$.

nInt() (Numerisches Integral)

Katalog > 

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.

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

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.

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

nom()

Katalog > 

nom(*Effektivzins*, *CpY*) ⇒ *Wert*



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

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 49.

nor

  **Tasten**

BoolescherAusdr1 **nor** *BoolescherAusdr2*
ergibt *Boolescher Ausdruck*

BoolescheListe1 **nor** *BoolescheListe2*
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 Ob bzw. Oh 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*

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

norm(*Vektor*) \Rightarrow *Ausdruck*

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

Gibt die Frobeniusnorm zurück.

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

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog > 

normCdf(*untereGrenze*,*obereGrenze* [, μ [, σ]]) \Rightarrow *Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog > 

Berechnet die Normalverteilungswahrscheinlichkeit zwischen *untereGrenze* und *obereGrenze* für die angegebenen μ (Standard = 0) und σ (Standard = 1).

Für $P(X \leq \textit{obereGrenze})$ setzen Sie *untereGrenze* = -9E999.

normPdf() (Wahrscheinlichkeitsdichte)

Katalog > 

normPdf(*XWert*[, μ [, σ]]) \Rightarrow *Zahl*, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion für die Normalverteilung an einem bestimmten *XWert* 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 *Ganzzahl1* \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 \geq 3)	true
not 0hB0 \blacktriangleright Base16	0hFFFFFFFFFFFFFF4F
not not 2	2

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O.

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

Im Bin-Modus:

0b100101 \blacktriangleright Base10	37
not 0b100101	
0b11111111111111111111111111111111 \blacktriangleright	
not 0b100101 \blacktriangleright Base10	-38

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

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

nPr(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.

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

npv(Zinssatz, CFO, CFListe[, CFFreq])

Finanzfunktion zur Berechnung des Nettobarwerts; die Summe der Barwerte für die Bar-Zuflüsse und -Abflüsse. Ein positives Ergebnis für npv zeigt eine rentable Investition an.

Zinssatz ist der Satz, zu dem die Cash-Flows (der Geldpreis) für einen Zeitraum.

CFO 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 *CFO*.

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 Ganzzahlen < 10.000 sein.

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

nSolve() (Numerische Lösung)

**nSolve(Gleichung, Var
[=Schätzwert])** ⇒ Zahl oder Fehler_
String

**nSolve(Gleichung, Var
[=Schätzwert], UntereGrenze)** ⇒ Zahl
oder Fehler_String

**nSolve(Gleichung, Var
[=
Schätzwert
, UntereGrenze, ObereGrenze]** ⇒ Zahl
oder Fehler_String

**nSolve(Gleichung, Var[=Schätzwert]) |
UntereGrenze ≤ Var ≤ ObereGrenze**
⇒ Zahl oder Fehler_String

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

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.

$\text{nSolve}(x^2+5\cdot x-25=9,x) x<0$	-8.84429
$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right) r>0 \text{ and } r<0.25$	0.006886
$\text{nSolve}(x^2=-1,x)$	"No solution found"

O

OneVar (Eine Variable)

OneVar [1,]X[,*[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 166.)

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 241).

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*

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* ⇒ *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).

```
Define g(x)=Func
  If x<=0 or x>=5
  Goto end
  Return x*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.

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**.

ord() (Numerischer Zeichencode)

ord(String) ⇒ *Ganzzahl*

ord("hello")	104
--------------	-----

ord(Liste1) ⇒ *Liste*

char{104}	"h"
-----------	-----

ord(char{24})	24
---------------	----

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({"alpha", "beta"})	{97,98}
------------------------	---------

P**►Rx()** (Kartesische x-Koordinate)

►Rx(rAusdr, θAusdr) ⇒ *Ausdruck*

Im Bogenmaß-Modus:

►Rx(rListe, θListe) ⇒ *Liste*

►Rx(4,60°)	2.
------------	----

►Rx(rMatrix, θMatrix) ⇒ *Matrix*

►Rx($\{-3, 10, 1.3\}, \{\frac{\pi}{3}, \frac{\pi}{4}, 0\}$)	$\{-1.5, 7.07107, 1.3\}$
---------------------------------------------------------------	--------------------------

Gibt die äquivalente x-Koordinate des Paares (r, θ) zurück.

Hinweis: Das θ-Argument wird gemäß der aktuellen Winkelmodus-Einstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein Ausdruck, können Sie °, g oder r benutzen, um die Winkelmodus-Einstellung temporär zu ändern.

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

P►Ry() (Kartesische y-Koordinate)

Katalog > 

P►Ry(*rWert*, *θWert*)⇒*Wert*

Im Bogenmaß-Modus:

P►Ry(*rListe*, *θListe*)⇒*Liste*

P►Ry(4,60°) 3.4641

P►Ry(*rMatrix*, *θMatrix*)⇒*Matrix*

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

Gibt die äquivalente y-Koordinate des Paares (*r*, *θ*) zurück.

Hinweis: Das *θ*-Argument wird gemäß der aktuellen Winkelmodus-Einstellung als Grad, Neugrad oder Bogenmaß interpretiert.

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

PassErr (ÜbgebFeh)

Katalog > 

PassErr

Ein Beispiel zu **PassErr** finden Sie im Beispiel 2 unter Befehl **Versuche (Try)**, Seite 179.

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 179.

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

piecewise() (Stückweise)

Katalog > 

piecewise(*Ausdr1* [, *Bedingung1* [, *Ausdr2* [, *Bedingung2* [, ...]]]])

Gibt Definitionen für eine stückweise 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()

Katalog > 

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()

Katalog > 

poissPdf($\lambda, X\text{Wert}$) $\Rightarrow \text{Zahl}$, 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

Katalog > 

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 Anzeigeformatierung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen, und sie nimmt keine Aktualisierung von *ans* vor.

Hinweis: Siehe auch ►Rect, Seite 138.

komplexerWert ►Polar

Zeigt *komplexerVektor* in Polarform an.

- Der Grad-Modus für Winkel gibt $(r \angle \theta)$ zurück.
- Der Bogenmaß-Modus für Winkel gibt $re^{i\theta}$ zurück.

komplexerWert kann jede komplexe Form haben. Eine $re^{i\theta}$ -Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Hinweis: Für eine Eingabe in Polarform müssen Klammern $(r \angle \theta)$ verwendet werden.

Im Bogenmaß-Modus:

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

Im Neugrad-Modus:

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

Im Grad-Modus:

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

polyEval() (Polynom auswerten)

polyEval(Liste1, Ausdr1)⇒Ausdruck

polyEval(Liste1, Liste2)⇒Ausdruck

Interpretiert das erste Argument als Koeffizienten eines nach fallenden Potenzen geordneten Polynoms und gibt das Polynom bezüglich des zweiten Arguments zurück.

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

polyRoots()

Katalog > 

polyRoots(Poly,Var) ⇒Liste

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

polyRoots(KoeffListe) ⇒Liste

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

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 34.

PowerReg

Katalog > 

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 166.)

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 241).

Ausgabevariable	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 X -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.YReg	Liste der Datenpunkte in der modifizierten Y -Liste, die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

Prgm
Block
EndPrgm

GCD berechnen und Zwischenergebnisse anzeigen.

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.

```

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

Done

```

proggcd(4560,450)
  
```

450 60

60 30

30 0

GCD=30

Done

prodSeq()

Siehe Π(), Seite 212.

Product (PI) (Produkt)

Siehe Π(), Seite 212.

product() (Produkt)

product(*Liste*[, *Start*[, *Ende*]])⇒*Ausdruck*

Gibt das Produkt der Elemente von *Liste* zurück. *Start* und *Ende* sind optional. Sie geben einen Elementebereich an.

product(*Matrix1*[, *Start*[, *Ende*]])⇒*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.

product({1,2,3,4})	24
product({4,5,8,9},2,3)	40

product($\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$)	[28 80 162]
product($\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$,1,2)	[4 10 18]

product() (Produkt)Katalog > 

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

propFrac() (Echter Bruch)Katalog > 

propFrac(Wert1[, Var]) ⇒ 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.

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.

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

$\text{propFrac}\left(\frac{11}{7}\right)$	$1 + \frac{4}{7}$
$\text{propFrac}\left(3 + \frac{1}{11} + 5 + \frac{3}{4}\right)$	$8 + \frac{37}{44}$
$\text{propFrac}\left(3 + \frac{1}{11} - \left(5 + \frac{3}{4}\right)\right)$	$-2 - \frac{29}{44}$

Q**QR**Katalog > 

QR Matrix, *q*Matrix, *r*Matrix[, Tol]



Die Fließkommazahl (9,) in m1 bewirkt, dass das Ergebnis in Fließkommaform berechnet wird.

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.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix}$$

QR	<i>m1,qm,rm</i>	Done
<i>qm</i>	$\begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$	
<i>rm</i>	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$	

Sie haben die Option, dass jedes Matricelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommalelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Anderenfalls wird *Tol* ignoriert.

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

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 *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 166.)

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 241).

Ausgabevariable	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 X -Liste, 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 Y -Liste, 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 X, Y [, Häuf] [, Kategorie, Mit]

Berechnet die polynomiale Regression vierter Ordnung $y = a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ auf Listen X und Y mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 166.)

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 241).

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 X -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde

Ausgabevariable	Beschreibung
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 verwendet wurde</i>
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.

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

Im Grad-Modus:

$$\underline{\text{R}\blacktriangleright\text{P}\theta(2,2)} \quad 45.$$

Im Neugrad-Modus:

$$\underline{\text{R}\blacktriangleright\text{P}\theta(2,2)} \quad 50.$$

Im Bogenmaß-Modus:

$$\underline{\text{R}\blacktriangleright\text{P}\theta(3,2)} \quad 0.588003$$

$$\text{R}\blacktriangleright\text{P}\theta\left(\left[3 \ -4 \ 2\right], \left[0 \ \frac{\pi}{4} \ 1.5\right]\right)$$

$$\underline{\hspace{10em} [0. \ 2.94771 \ 0.643501]}$$

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:

$$\underline{\text{R}\blacktriangleright\text{Pr}(3,2)} \quad 3.60555$$

$$\text{R}\blacktriangleright\text{Pr}\left(\left[3 \ -4 \ 2\right], \left[0 \ \frac{\pi}{4} \ 1.5\right]\right)$$

$$\underline{\hspace{10em} \left[3 \ 4.07638 \ \frac{5}{2}\right]}$$

▶ Rad

Katalog >

Wert ▶ Rad ⇒ *Wert*

Im Grad-Modus:

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

► Rad

Katalog > 

Wandelt das Argument ins Bogenmaß um.

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

Im Neugrad-Modus:

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

rand() (Zufallszahl)

Katalog > 

rand() ⇒ *Ausdruck*

rand(#Trials) ⇒ *Liste*

rand() gibt einen Zufallswert zwischen 0 und 1 zurück.

rand(#Trials) gibt eine Liste zurück, die #Trials Zufallswerte zwischen 0 und 1 enthält.

Setzt Ausgangsbasis für Zufallszahlengenerierung.

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

randBin() (Zufallszahl aus Binomialverteilung)

Katalog > 

randBin(*n*, *p*) ⇒ *Ausdruck*

randBin(*n*, *p*, #Trials) ⇒ *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*)
⇒ *Ausdruck*

randInt

(*lowBound*,*upBound*,
#Trials) ⇒ *Liste*

randInt(3,10)	3.
randInt(3,10,4)	{9.,3.,4.,7.}

randInt() (Ganzzahlige Zufallszahl)

Katalog > 

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*) ⇒ *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	<i>Done</i>									
randMat(3,3)	<table border="1"><tr><td>8</td><td>-3</td><td>6</td></tr><tr><td>-2</td><td>3</td><td>-6</td></tr><tr><td>0</td><td>4</td><td>-6</td></tr></table>	8	-3	6	-2	3	-6	0	4	-6
8	-3	6								
-2	3	-6								
0	4	-6								

Hinweis: Die Werte in dieser Matrix ändern
sich mit jedem Drücken von .

randNorm() (Zufallsnorm)

Katalog > 

randNorm(μ, σ) ⇒ *Ausdruck*
randNorm($\mu, \sigma, \#Trials$) ⇒ *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 [$\mu-3\cdot\sigma$,
 $\mu+3\cdot\sigma$].

RandSeed 1147	<i>Done</i>
randNorm(0,1)	0.492541
randNorm(3,4.5)	-3.54356

randNorm() (Zufallsnorm)

Katalog > 

randNorm(μ , σ , #Trials) gibt eine Liste mit #Trials Dezimalzahlen aus der angegebenen Normalverteilung zurück.

randPoly() (Zufallspolynom)

Katalog > 

randPoly(Var, Ordnung) ⇒ Ausdruck

Gibt 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]) ⇒ Liste

Gibt 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() (Reell)

Katalog >

real(List I) ⇒ Liste

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

Gibt für jedes Element den Realteil zurück.

real(Matrix I) ⇒ Matrix

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

Gibt für jedes Element den Realteil zurück.

► Rect

Katalog >

Vektor ► Rect

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

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

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 125.

komplexer Wert ► Rect

Zeigt *komplexerWert* in der kartesischen Form a+bi an. *komplexerWert* kann jede komplexe Form haben. Eine rei^θ -Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Im Bogenmaß-Modus:

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

Hinweis: Für eine Eingabe in Polarform müssen Klammern ($r \angle \theta$) verwendet werden.

Im Neugrad-Modus:

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

Im Grad-Modus:

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

Hinweis: Wählen Sie zur Eingabe von \angle das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ref() (Diagonalform)

ref(*Matrix1* [, *Tol*]) ⇒ *Matrix*

Gibt die Diagonalform von *Matrix1* 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. Anderenfalls wird *Tol* ignoriert.

- Wenn Sie 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:
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

Vermeiden Sie nicht definierte Elemente in *Matrix1*. 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:

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

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

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.

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

Hinweis: Siehe auch **rref()** page 150.

RefreshProbeVars

RefreshProbeVars

Ermöglicht den Zugriff auf Sensordaten von allen verbundenen Sensorsonden in Ihrem TI-Basic-Programm.

StatusVar Value

Status

- statusVar* =0 Normal (Programmausführung fortsetzen)
Die Applikation Vernier DataQuest™ befindet sich im Data Collection-Modus.
- statusVar* =1 **Hinweis:** Die Applikation Vernier DataQuest™ muss sich im Messgerätmodus befinden, damit dieser Befehl funktioniert. 
- statusVar* =2 Die Applikation Vernier DataQuest™ wurde nicht gestartet.
- statusVar* =3 Die Applikation Vernier DataQuest™ wurde gestartet, ist jedoch noch nicht mit Sonden verbunden.

Beispiel

```
Define temp()=
Prgm
© 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"
EndIf
© 1 Sekunde zwischen den Messungen warten
Wait 1
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) ⇒ Wert
remain(Liste1, Liste2) ⇒ Liste
remain(Matrix1, Matrix2) ⇒ Matrix

Gibt den Rest des ersten Arguments
bezüglich des zweiten Arguments gemäß
folgender Definitionen zurück:

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

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 106.

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

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

Request

Request *promptString*, *var* [, *FlagAnz*
[, *statusVar*]]

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

Programmierbefehl: Pausiert das
Programm und zeigt ein Dialogfeld mit
der Meldung *promptString* sowie einem
Eingabefeld für die Antwort des
Benutzers an.

Definieren Sie ein Programm:

```
Define request_demo()=Prgm
  Request "Radius: ",r
  Disp "Fläche = ",pi*r^2
EndPrgm
```

Starten Sie das Programm und geben Sie eine
Antwort ein:

```
request_demo()
```

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.

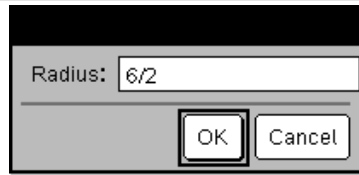
- Wenn *FlagAnz* fehlt oder den Wert **1** ergibt, werden die Eingabeaufforderung und die Benutzerantwort im Calculator-Protokoll angezeigt.
- Wenn *FlagAnz* den Wert **0** ergibt, werden die Aufforderung und die Antwort nicht im Protokoll angezeigt.

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.
- Anderenfalls 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:

```
Define Fkt(Arg1, ...Argn) =
  Benutzerantwort
```



Ergebnis nach Auswahl von **OK**:

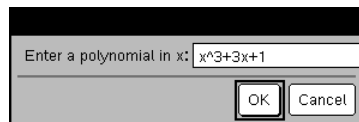
```
Radius: 6/2
Fläche = 28.2743
```

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**:

```
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  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: Siehe auch **RequestStr**, page 143.

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 ("").

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()
```

Hinweis: Sie können den Befehl **RequestStr** in benutzerdefinierten Programmen verwenden, jedoch nicht in Funktionen.

Zum Anhalten eines Programms mit dem Befehl **RequestStr** in einer Endlosschleife:

- **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: Siehe auch **Request**, page 141.



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 Beispiels: 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]) ⇒ *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) ⇒ *Ausdruck*

Gibt die rechte Seite einer Gleichung oder Ungleichung zurück.

rk23 ()

rk23(Ausdr, Var, abhVar, {Var0, VarMax}, abhVar0, VarSchritt [, diftol]) ⇒ *Matrix*

Differentialgleichung:

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

rk23(AusdrSystem, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt[, diftol]) ⇒ *Matrix*

rk23(0.001·y·(100-y),t,y,{0,100},10,1)

0.	1.	2.	3.	4.
10.	10.9367	11.9493	13.042	14.2

rk23(AusdrListe, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt[, diftol]) ⇒ *Matrix*

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

Verwendet die Runge-Kutta-Methode zum Lösen des Systems

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

mit $\text{abhVar}(\text{Var0}) = \text{abhVar0}$ auf dem Intervall $[\text{Var0}, \text{VarMax}]$. 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.

Dieselbe Gleichung mit *diftol* auf $1.E-6$

rk23(0.001·y·(100-y),t,y,{0,100},10,1,1.E-6)

0.	1.	2.	3.	4.
10.	10.9367	11.9495	13.0423	14.2189

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

Gleichungssystem:

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

mit $y1(0) = 2$ und $y2(0) = 5$

rk23($\begin{cases} -y1+0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}$,t,{y1,y2},{0,5},{2,5},1)

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

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*).

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

Wenn *VarSchritt* Null ergibt, werden Lösungen an den „Runge-Kutta“ *Var*-Werten zurückgegeben.

diffol ist die Fehlertoleranz (standardmäßig 0.001).

root() (Wurzel)

root(Wert) \Rightarrow *Wurzel*

root(Wert1, Wert2) \Rightarrow *Wurzel*

root(Wert) gibt die Quadratwurzel von *Wert* zurück.

$\sqrt[3]{8}$	2
$\sqrt[3]{3}$	1.44225

root() (Wurzel)

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>

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.

```
rotate(0b11111111111111111111111111111111)
0b100000000000000000000000000000001P
rotate(256,1)                                0b1000000000
```

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

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. Ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Bit nach rechts rotieren).

Im Hex-Modus:

```
rotate(0h78E)                                0h3C7
rotate(0h78E,-2)                             0h800000000000001E3
rotate(0h78E,2)                              0h1E38
```

Beispielsweise in einer Rechtsrotation:

Jedes Bit rotiert nach rechts.

0b00000000000001111010110000110101

Bit ganz rechts rotiert nach ganz links.

ergibt sich:

0b100000000000000111101011000011010

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

rotate(*Liste1*,#*Rotationen*) ⇒ *Liste*

Im Dec-Modus:

Gibt eine um *#Rotationen* Elemente nach rechts oder links rotierte Kopie von *Liste1* zurück. Verändert *Liste1* nicht.

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

rotate() (Rotieren)

Katalog > 

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(*StringI*[,*#Rotationen*]) \Rightarrow *String*

Gibt eine um *#Rotationen* Zeichen nach rechts oder links rotierte Kopie von *StringI* zurück. Verändert *StringI* nicht.

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. Ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Zeichen nach rechts rotieren)

<code>rotate({1,2,3,4})</code>	<code>{4,1,2,3}</code>
<code>rotate({1,2,3,4},-2)</code>	<code>{3,4,1,2}</code>
<code>rotate({1,2,3,4},1)</code>	<code>{2,3,4,1}</code>

<code>rotate("abcd")</code>	<code>"dabc"</code>
<code>rotate("abcd",-2)</code>	<code>"cdab"</code>
<code>rotate("abcd",1)</code>	<code>"bcda"</code>

round() (Runden)

Katalog > 

round(*WertI*[,*Stellen*]) \Rightarrow *Wert*

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(*ListeI*[,*Stellen*]) \Rightarrow *Liste*

Gibt eine Liste von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

round(*MatrixI*[,*Stellen*]) \Rightarrow *Matrix*

Gibt eine Matrix von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

<code>round(1.234567,3)</code>	1.235
--------------------------------	-------

<code>round({pi,sqrt(2),ln(2)},4)</code>	<code>{3.1416,1.4142,0.6931}</code>
------------------------------------------	-------------------------------------

<code>round([[ln(5) ln(3)],[pi e^1]],1)</code>	<code>[1.6 1.1][3.1 2.7]</code>
------------------------------------------------	---------------------------------

rowAdd() (Zeilenaddition)Katalog > **rowAdd**(*Matrix1*, *rIndex1*, *rIndex2*) ⇒ *Matrix*

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

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*) ⇒ *Ausdruck*

Gibt die Anzahl der Zeilen von *Matrix* zurück.

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

Hinweis: Siehe auch **colDim()** Seite 25.

rowNorm() (Zeilennorm)Katalog > **rowNorm**(*Matrix*) ⇒ *Ausdruck*

Gibt das Maximum der Summen der Absolutwerte der Elemente der Zeilen von *Matrix* zurück.

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

Hinweis: Alle Matrixelemente müssen zu Zahlen vereinfachbar sein. Siehe auch **colNorm()** Seite 26.

rowSwap() (Zeilentausch)Katalog > **rowSwap**(*Matrix1*, *rIndex1*, *rIndex2*) ⇒ *Matrix*

Gibt *Matrix1* zurück, in der die Zeilen *rIndex1* und *rIndex2* vertauscht sind.

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

rref() (Reduzierte Diagonalform)



Katalog > 

$\text{rref}(\text{Matrix1}, \text{Tol}) \Rightarrow \text{Matrix}$

Gibt die reduzierte Diagonalform von *Matrix1* zurück.

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

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 Fließkommalelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Anderenfalls wird *Tol* ignoriert.

- Wenn Sie   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:
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

Hinweis: Siehe auch **rref()** page 139.

S

sec() (Sekans)

 Taste

$\text{sec}(\text{Wert1}) \Rightarrow \text{Wert}$

Im Grad-Modus:

$\text{sec}(\text{Liste1}) \Rightarrow \text{Liste}$

$$\text{sec}(45) \quad 1.41421$$

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

$$\text{sec}\{1,2,3,4\} \quad \{1.00015, 1.00081, 1.00244\}$$

sec() (Sekans)

 Taste

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 vorübergehend aufzuheben.

sec⁻¹() (Arkussekans)

 Taste

sec⁻¹(Wert1) ⇒ Wert

Im Grad-Modus:

sec⁻¹(Liste1) ⇒ Liste

sec⁻¹(1) 0.

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

Im Neugrad-Modus:

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

sec⁻¹(√2) 50.

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

Im Bogenmaß-Modus:

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

sech⁻¹(Liste1) ⇒ Liste

sech⁻¹(1) 0
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*] ...

Programmierbefehl: Sendet einen oder mehrere TI-Innovator™ Hub Befehle an den verbundenen Hub.

exprOrString muss ein gültiger TI-Innovator™ Hub Befehl sein. Normalerweise enthält *exprOrString* einen Befehl "SET ..." zum Steuern eines Geräts oder einen Befehl "READ ..." zum Anfordern von Daten.

Die Argumente werden hintereinander an den Hub gesendet.

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

Hinweis: Siehe auch **Get** (Seite 66), **GetStr** (Seite 73) und **eval()** (Seite 52).

Beispiel: Schalten Sie das blaue Element der integrierten RGB LED 0,5 Sekunden lang ein.

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

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
```

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.

```
n:=50 50
m:=4 4
Send "SET SOUND eval(m·n)" Done
iostr.SendAns "SET SOUND 200"
```

seq() (Folge)

Katalog >

seq(Ausdr, Var, Von, Bis[, Schritt]) ⇒ 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.

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

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie .

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie aus.

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

seqGen()

Katalog >

seqGen(Ausdr, Var, abhVar, {Var0, VarMax}[, ListeAnfTerme [, VarSchritt [, ObergrWert]]) ⇒ Liste

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]]) ⇒ Matrix

Generieren Sie die ersten 5 Terme der Folge $u(n) = u(n-1)^2/2$ mit $u(1)=2$ und *VarSchritt*=1.

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

Beispiel mit *Var0*=2:

$$\begin{array}{l} \text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2, 5\}, \{3\}\right) \\ \left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\} \end{array}$$

System zweiter Folgen:

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.

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u_2^{n-1}}{2} + u_1(n-1)\right\}, n, \{u_1, u_2\}, \{1, 5\}, \begin{bmatrix} _ \\ 2 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{2} & \frac{19}{2} \end{bmatrix}$$

Hinweis: Die Lücke ($_$) in der oben aufgeführten Anfangsterm-Matrix zeigt an, dass der Anfangsterm für $u_1(n)$ mit der expliziten Folge-Formel $u_1(n)=1/n$ berechnet wird.

seqn()

seqn(*Ausdr*(*u*, *n* [, *ListeAnfTerme* [, *nMax* [, *ObergrWert*]]]) \Rightarrow *Liste*

Generiert eine Term-Liste für eine Folge $u(n)=\text{Ausdr}(u, n)$ 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*

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 *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

Generieren Sie die ersten 6 Terme der Folge $u(n) = u(n-1)/2$ mit $u(1)=2$.

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

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

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

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

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.

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) \rightarrow *var* gespeichert haben, können Sie **setMode**(*var*) verwenden, um diese Einstellungen wiederherzustellen, bis die Funktion oder das Programm beendet wird. Siehe **getMode**(), Seite 72.

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 wiederhergestellt 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>

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() (Verschieben)

shift(*Ganzzahl*
[,*#Verschiebungen*]) \Rightarrow *Ganzzahl*

Im Bin-Modus:

```

shift(0b1111010110000110101)
                                0b111101011000011010
shift(256,1)                      0b1000000000

```

Im Hex-Modus:

Verschiebt die Bits in einer binären ganzen Zahl. *Ganzzahl* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl* 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.

<code>shift(0h78E)</code>	0h3C7
<code>shift(0h78E,-2)</code>	0h1E3
<code>shift(0h78E,2)</code>	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix `0b` bzw. `0h` ein (Null, nicht der Buchstabe O).

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.

`0b0000000000000111101011000011010`

Setzt 0 ein, wenn Bit ganz links 0 ist, und 1, wenn Bit ganz links 1 ist.

Es ergibt sich:

`0b00000000000000111101011000011010`

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 Dec-Modus:

<code>shift({1,2,3,4})</code>	{undef,1,2,3}
<code>shift({1,2,3,4},-2)</code>	{undef,undef,1,2}
<code>shift({1,2,3,4},2)</code>	{3,4,undef,undef}

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(*StringI* [,*#Verschiebungen*])⇒*String*

Gibt eine um *#Verschiebungen* Zeichen nach rechts oder links verschobene Kopie von *ListeI* zurück. Verändert *StringI* 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)

sign(*WertI*)⇒*Wert*

sign(-3.2)	-1
------------	----

sign(*ListeI*)⇒*Liste*

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

sign(*MatrixI*)⇒*Matrix*

Gibt für reelle und komplexe *WertI* *WertI* / **abs(*WertI*)** zurück, wenn *WertI* ≠ 0.

Bei Komplex-Formatmodus Reell:

sign([-3 0 3])	[-1 undef 1]
----------------	--------------

Gibt 1 zurück, wenn *WertI* positiv ist.

Gibt -1 zurück, wenn *WertI* 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(KoeffMatrix, KonstVektor[, Tol]) ⇒ Matrix

Ergibt einen Spaltenvektor, der die Lösungen für ein lineares Gleichungssystem enthält.

Hinweis: Siehe auch **linSolve()**, Seite 92.

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 Fließkommalelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Anderenfalls 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]) ⇒ Matrix

Löst mehrere lineare Gleichungssysteme, die alle dieselben Gleichungskoeffizienten, aber unterschiedliche Konstanten haben.

Auflösen nach x und y:

$$x + 2y = 1$$

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

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

Die Lösung ist x=-3 und y=2.

Auflösen:

$$ax + by = 1$$

$$cx + dy = 2$$

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

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

Auflösen:

$$x + 2y = 1$$

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

$$x + 2y = 2$$

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

simult() (Gleichungssystem)

Katalog > 

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.

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

Für das erste System ist $x=-3$ und $y=2$. Für das zweite System ist $x=-7$ und $y=9/2$.

sin() (Sinus)

 Taste

$\sin(\text{Wert}) \Rightarrow \text{Wert}$

$\sin(\text{Liste1}) \Rightarrow \text{Liste}$

$\sin(\text{Wert1})$ gibt den Sinus des Arguments zurück.

$\sin(\text{Liste1})$ gibt eine Liste zurück, die für jedes Element von *Liste1* den Sinus enthält.

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können $^{\circ}$, $^{\text{G}}$ oder $^{\text{r}}$ benutzen, um die Winkelmoduseinstellung temporär zu ändern.

$\sin(\text{Quadratmatrix1}) \Rightarrow \text{Quadratmatrix}$

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Grad-Modus:

$\sin\left(\frac{\pi}{4}^{\text{r}}\right)$	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
$\sin(45^{\circ})$	0.707107

Im Bogenmaß-Modus:

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

$\sin^{-1}()$ (Arkussinus)

 Taste

$\sin^{-1}(\text{Wert1}) \Rightarrow \text{Wert}$

Im Grad-Modus:

$\sin^{-1}()$ (Arkussinus)

 **Taste**

$\sin^{-1}(\text{Liste1}) \Rightarrow \text{Liste}$

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

$\sin^{-1}(\text{Wert1})$ gibt den Winkel, dessen Sinus Wert1 ist, zurück.

Im Neugrad-Modus:

$\sin^{-1}(\text{Liste1})$ gibt in Form einer Liste für jedes Element aus Liste1 den inversen Sinus zurück.

$\sin^{-1}\{1\}$ 100.

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

Im Bogenmaß-Modus:

$\sin^{-1}\{0,0,2,0,5\}$ $\{0,,-0,201358,0,523599\}$

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

$\sin^{-1}(\text{Quadratmatrix1}) \Rightarrow \text{Quadratmatrix}$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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()`.

$\sin^{-1}\left(\begin{pmatrix} 1 & 5 \\ 4 & 2 \end{pmatrix}\right)$
 $\begin{bmatrix} -0,174533-0,12198 \cdot i & 1,74533-2,35591 \cdot i \\ 1,39626-1,88473 \cdot i & 0,174533-0,593162 \cdot i \end{bmatrix}$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$\sinh()$ (Sinus hyperbolicus)

Katalog > 

$\sinh(\text{Wert1}) \Rightarrow \text{Wert}$

$\sinh(1,2)$ 1.50946

$\sinh(\text{Liste1}) \Rightarrow \text{Liste}$

$\sinh\{0,1,2,3\}$ $\{0,1,50946,10,0179\}$

$\sinh(\text{Wert1})$ gibt den Sinus hyperbolicus des Arguments zurück.

$\sinh(\text{Liste1})$ gibt in Form einer Liste für jedes Element aus Liste1 den Sinus hyperbolicus zurück.

$\sinh(\text{Quadratmatrix1}) \Rightarrow \text{Quadratmatrix}$

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*

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

sinh⁻¹(*Liste1*) ⇒ *Liste*

$$\sinh^{-1}(\{0,2,1,3\}) \quad \{0,1.48748,1.81845\}$$

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:

$$\sinh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$$

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()**.

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 166.)

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 241).

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

Ausgabevariable	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*] ...

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

SortA *Vektor1* [, *Vektor2*] [, *Vektor3*] ...

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

Sortiert die Elemente des ersten Arguments in aufsteigender Reihenfolge.

<i>list1</i>	$\{1,2,3,4\}$
--------------	---------------

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.

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

SortA <i>list2,list1</i>	Done
--------------------------	------

<i>list2</i>	$\{1,2,3,4\}$
--------------	---------------

<i>list1</i>	$\{4,3,2,1\}$
--------------	---------------

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 241).

SortD (In absteigender Reihenfolge sortieren)

Katalog > 

SortD *Liste1* [, *Liste2*] [, *Liste3*] ...

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

SortD *Vektor1* [, *Vektor2*] [, *Vektor3*] ...

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

Identisch mit **SortA** mit dem Unterschied, dass **SortD** die Elemente in absteigender Reihenfolge sortiert.

SortD <i>list1,list2</i>	Done
--------------------------	------

<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 241).

►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]`.

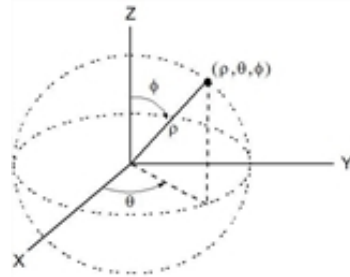
Windows®: Drücken Sie `Strg+Eingabetaste`.

Macintosh®: Drücken Sie `⌘+Eingabetaste`.

iPad®: Halten Sie die `Eingabetaste` gedrückt und wählen Sie `≈` aus.

```
[ 1 2 3 ] ►Sphere  
[ 3.74166  ∠1.10715  ∠0.640522 ]
```

```
( [ 2  ∠  $\frac{\pi}{4}$   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 *Listel* 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.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.X̄

stat.b9	stat.FBlock	Stat. \hat{p}	stat. Σx^2	stat. $\bar{X}1$
stat.b10	stat.Fcol	stat. $\hat{p}1$	stat. Σxy	stat. $\bar{X}2$
stat.bList	stat.FInteract	stat. $\hat{p}2$	stat. Σy	stat. $\bar{X}Diff$
stat. χ^2	stat.FreqReg	stat. $\hat{p}Diff$	stat. Σy^2	stat. $\bar{X}List$
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. \bar{y}
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. \hat{y}
stat.CookDist	stat.MaxX	stat.PValRow	stat.SElope	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])` ⇒ Ausdruck

Ergibt die Populations-Standardabweichung der Elemente in *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Im Bogenmaß- und automatischen Modus:

<code>stDevPop({1,2,5,-6,3,-2})</code>	3.59398
<code>stDevPop({1.3,2.5,-6.4},{3,2,5})</code>	4.11107

stDevPop() (Populations-Standardabweichung)

Katalog > 

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

stDevPop(*MatrixI* [, *Häufigkeitsmatrix*]) ⇒ *Matrix*

Ergibt einen Zeilenvektor der Populations-Standardabweichungen der Spalten in *MatrixI*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *MatrixI* in der gegebenen Reihenfolge entsprechend.

Hinweis: *MatrixI* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

$$\text{stDevPop} \left(\begin{array}{ccc} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{array} \right) \quad [3.26599 \quad 2.94392 \quad 1.63299]$$

$$\text{stDevPop} \left(\begin{array}{cc} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{array} \right) \cdot \begin{array}{c} [4 \ 2] \\ [3 \ 3] \\ [1 \ 7] \end{array} \quad [2.52608 \quad 5.21506]$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog > 

stDevSamp(*Liste* [, *Häufigkeitsliste*]) ⇒ *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 241).

$$\text{stDevSamp}(\{1, 2, 5, -6, 3, -2\}) \quad 3.937$$

$$\text{stDevSamp}(\{1.3, 2.5, -6.4\}, \{3, 2, 5\}) \quad 4.33345$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog > 

stDevSamp(*MatrixI* [, *Häufigkeitsmatrix*]) ⇒ *Matrix*

Ergibt einen Zeilenvektor der Stichproben-Standardabweichungen der Spalten in *MatrixI*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *MatrixI* in der gegebenen Reihenfolge entsprechend.

Hinweis: *MatrixI* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

$\text{stDevSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right)$
$[4. \quad 3.60555 \quad 2.]$
$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right)$
$[2.7005 \quad 5.44695]$

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>i</i> :=0	0
Define <i>progI</i> ()=Prgm	Done
For <i>i</i> ,1,10,1	
If <i>i</i> =5	
Stop	
EndFor	
EndPrgm	
<i>progI</i> ()	Done
<i>i</i>	5

Store (Speichern)

Siehe → (speichern), Seite 221.

string() (String)

Katalog > 

string(*Ausdr*) ⇒ *String*

Vereinfacht *Ausdr* und gibt das Ergebnis als Zeichenkette zurück.

string(1.2345)	"1.2345"
string(1+2)	"3"

subMat() (Untermatrix)Katalog > 

subMat(*MatrixI* [, *vonZeil*] [, *vonSpl*] [, *bisZeil*] [, *bisSpl*]) ⇒ *Matrix*

Gibt die angegebene Untermatrix von *MatrixI* zurück.

Vorgaben: *vonZeil*=1, *vonSpl*=1, *bisZeil*=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 212.**sum() (Summe)**Katalog > 

sum(*Liste* [, *Start*] [, *Ende*]) ⇒ *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 241).

sum(*MatrixI* [, *Start*] [, *Ende*]) ⇒ *Matrix*

Gibt einen Zeilenvektor zurück, der die Summen der Elemente aus den Spalten von *MatrixI* enthält.

Start und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *MatrixI* werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 241).

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($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$)	$[5 \ 7 \ 9]$
sum($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$)	$[12 \ 15 \ 18]$
sum($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, 2,3$)	$[11 \ 13 \ 15]$

**sumIf(Liste,Kriterien[,
SummeListe])**⇒Wert

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

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 241).

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

system() (System)Katalog > **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)**Katalog > **MatrixIT** ⇒ *matrix*

Gibt die komplex konjugierte, transponierte Matrix von *MatrixI* zurück.

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

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

tan() (Tangens) Taste**tan(WertI)** ⇒ *Wert*

Im Grad-Modus:

tan(ListeI) ⇒ *Liste*

tan(WertI) gibt den Tangens des Arguments zurück.

tan(ListeI) gibt in Form einer Liste für jedes Element in *ListeI* 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:

$$\begin{array}{l} \tan\left(\frac{\pi}{4}\right)^{\text{r}} \qquad 1. \\ \tan(45) \qquad 1. \\ \tan(\{0,60,90\}) \qquad \{0.,1.73205,\text{undef}\} \end{array}$$

$$\begin{array}{l} \tan\left(\frac{\pi}{4}\right)^{\text{r}} \qquad 1. \\ \tan(50) \qquad 1. \\ \tan(\{0,50,100\}) \qquad \{0.,1.,\text{undef}\} \end{array}$$

Im Bogenmaß-Modus:

tan() (Tangens)

 Taste

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45^\circ)$	1.
$\tan\left(\left\{\pi, \frac{\pi}{3}, \pi, \frac{\pi}{4}\right\}\right)$	$\{0, 1.73205, 0, 1.\}$

tan(QuadratmatrixI)⇒Quadratmatrix

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

QuadratmatrixI muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$\tan\left(\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}\right)$	$\begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$
---------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------

tan⁻¹() (Arkustangens)

 Taste

tan⁻¹(WertI)⇒Wert

Im Grad-Modus:

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

tan⁻¹(ListeI)⇒Liste

tan⁻¹(WertI) gibt den Winkel zurück, dessen Tangens *WertI* ist.

Im Neugrad-Modus:

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

tan⁻¹(ListeI) gibt in Form einer Liste für jedes Element aus *ListeI* den inversen Tangens zurück.

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

Im Bogenmaß-Modus:

$\tan^{-1}(\{0, 0.2, 0.5\})$	$\{0, 0.197396, 0.463648\}$
------------------------------	-----------------------------

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

tan⁻¹(QuadratmatrixI)⇒Quadratmatrix

Im Bogenmaß-Modus:

$\tan^{-1}()$ (Arkustangens)

 Taste

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

$$\tan^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) = \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$\tanh()$ (Tangens hyperbolicus)

Katalog > 

$\tanh(\text{Wert1}) \Rightarrow \text{Wert}$

$$\tanh(1.2) = 0.833655$$

$\tanh(\text{Liste1}) \Rightarrow \text{Liste}$

$$\tanh(\{0,1\}) = \{0.,0.761594\}$$

$\tanh(\text{Wert1})$ gibt den Tangens hyperbolicus des Arguments zurück.

$\tanh(\text{Liste1})$ gibt in Form einer Liste für jedes Element aus *Liste1* den Tangens hyperbolicus zurück.

$\tanh(\text{Quadratmatrix1}) \Rightarrow \text{Quadratmatrix}$

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

Im Bogenmaß-Modus:

$$\tanh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) = \begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$\tanh^{-1}()$ (Arkustangens hyperbolicus)

Katalog > 

$\tanh^{-1}(\text{Wert1}) \Rightarrow \text{Wert}$

Im Komplex-Formatmodus "kartesisch":

$$\tanh^{-1}(0) = 0.$$

$\tanh^{-1}(\text{Liste1}) \Rightarrow \text{Liste}$

$$\tanh^{-1}(\{1,2,1,3\}) = \{\text{undef},0.518046-1.5708\cdot i,0.346574-1.5708\cdot i\}$$

$\tanh^{-1}(\text{Wert1})$ gibt den inversen Tangens hyperbolicus des Arguments zurück.

$\tanh^{-1}()$ (Arkustangens hyperbolicus)

Katalog > 

$\tanh^{-1}(\text{Liste1})$ gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Tangens hyperbolicus zurück.

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

\tanh^{-1}
(*Quadratmatrix1*) \Rightarrow *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.

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\tanh^{-1}\left(\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}\right)$$
$$\begin{bmatrix} -0.099353+0.164058 \cdot i & 0.267834-1.4908 \\ -0.087596-0.725533 \cdot i & 0.479679-0.9473i \\ 0.511463-2.08316 \cdot i & -0.878563+1.7901 \end{bmatrix}$$

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

tCdf()

Katalog > 

tCdf
(*UntGrenze, ObGrenze, FreiGrad*) \Rightarrow *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 $P(X \leq \text{obereGrenze})$ setzen Sie *untereGrenze* = -9E999.

Text

Katalog > 

Text *EingabeString*[, *FlagAnz*]

Programmierbefehl: Pausiert das Programm und zeigt die Zeichenkette *EingabeString* in einem Dialogfeld an.

Definieren Sie ein Programm, das fünfmal anhält und jeweils eine Zufallszahl in einem Dialogfeld anzeigt.



Wenn der Benutzer **OK** auswählt, wird die Programmausführung fortgesetzt.

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 141, oder **RequestStr**, Seite 143.

Hinweis: Sie können diesen Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

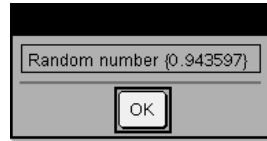
Schließen Sie in der Vorlage Prgm...EndPrgm jede Zeile mit  ab anstatt mit . Auf der Computertastatur halten Sie **Alt** gedrückt und drücken die **Eingabetaste**.

```
Define text_demo()=Prgm
  For i,1,5
    stringo:="Random number " &
string(rand(i))
    Text stringo
  EndFor
EndPrgm
```

Starten Sie das Programm:

```
text_demo()
```

Muster eines Dialogfelds:



tInterval *Liste[,Häuf[,KNiv]]*

(Datenlisteneingabe)

tInterval $\bar{x},sx,n[,KNiv]$

(Zusammenfassende statistische Eingabe)

Berechnet das Konfidenzintervall t . Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 166.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekanntem Populationsmittelwert
stat. \bar{x}	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat. σ	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert

tInterval_2Samp (Zwei-Stichproben-t-Konfidenzintervall)

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 166.)

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 241).

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
stat. $\bar{x}1$, stat. $\bar{x}2$	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat. $\sigma x1$, stat. $\sigma x2$	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</i> :=12	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 251).

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 **ÜbgebFeh (PassErr)** im Betrieb zu sehen, geben Sie das rechts gezeigte Programm `eigenvals()` ein. Sie starten das Programm, indem Sie jeden der folgenden Ausdrücke eingeben.

```
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 **ÜbgebFeh**, Seite 124.

```
Define progI()=Prgm
Try
z:=z+1
Disp "z incremented."
Else
Disp "Sorry, z undefined."
EndTry
EndPrgm
```

Done

```
z:=1:progI()
z incremented.
```

Done

```
DelVar z:progI()
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** μ_0 ,*Liste*[,*Häufigkeit*],[*Hypoth*]

(Datenlisteneingabe)

tTest μ_0 , \bar{x} ,*sx*,*n*,[*Hypoth*]

(Zusammenfassende statistische Eingabe)

Führt einen Hypothesen-Test für einen einzelnen, unbekanntem Populationsmittelwert μ durch, wenn die Populations-Standardabweichung σ unbekannt ist. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 166.)

Getestet wird $H_0: \mu = \mu_0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu < \mu_0$ setzen Sie *Hypoth*<0

Für $H_a: \mu \neq \mu_0$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \mu > \mu_0$ setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann

Ausgabevariable	Beschreibung
stat.df	Freiheitsgrade
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 $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2[, Hypoth [, Verteilt]]$

(Zusammenfassende statistische Eingabe)

Berechnet einen *t*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 166.)

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 241).

Ausgabevariable	Beschreibung
stat.t	Für die Differenz der Mittelwerte berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann

Ausgabevariable	Beschreibung
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>
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=1</i> .

tvfV()

Katalog > 

tvfV($N, I, PV, Pmt, [PpY], [CpY], [PmtAt]$) \Rightarrow Wert

tvfV(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 183) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvml()

Katalog > 

tvml($N, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$) \Rightarrow Wert

tvml(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 183) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvnN()

Katalog > 

tvnN($I, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$) \Rightarrow Wert

tvnN(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 183) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPmt()

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 183) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPV()

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 183) 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

TVM-Argumente*	Beschreibung	Datentyp
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 (Zwei Variable)

Katalog > 

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

Berechnet die 2-Variablen-Statistik. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 166.)

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 241).

Ausgabevariable	Beschreibung
stat. \bar{x}	Mittelwert der x-Werte
stat. x	Summe der x-Werte
stat. x2	Summe der x ² -Werte
stat.sx	Stichproben-Standardabweichung von x
stat. x	Populations-Standardabweichung von x
stat.n	Anzahl der Datenpunkte
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 > 

$\text{unitV}(\text{Vektor1}) \Rightarrow \text{Vektor}$

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.

```
unitV([1 2 1])
[0.408248 0.816497 0.408248]
unitV([1
      2
      3])
[0.267261
 0.534522
 0.801784]
```

unLock

Katalog > 

$\text{unLockVar1 } [, \text{Var2}] [, \text{Var3}] \dots$

unLockVar.

Entsperrt die angegebenen Variablen bzw. die Variablengruppe. Gesperrte Variablen können nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 96, und **getLockInfo()**, Seite 72.

```
a:=65
Lock a
getLockInfo(a)
a:=75
DelVar a
Unlock a
a:=75
DelVar a
65
Done
1
"Error: Variable is locked."
"Error: Variable is locked."
Done
75
Done
```

V

varPop() (Populationsvarianz)

Katalog > 

$\text{varPop}(\text{Liste} [, \text{Häufigkeitsliste}]) \Rightarrow \text{Ausdruck}$

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.

```
varPop({5,10,15,20,25,30})
72.9167
```

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 241).

varSamp() (Stichproben-Varianz)

varSamp(Liste[, Häufigkeitsliste]) ⇒ Ausdruck

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 241).

varSamp(MatrixI[, Häufigkeitsmatrix]) ⇒ Matrix

Gibt einen Zeilenvektor zurück, der die Stichproben-Varianz jeder Spalte von *MatrixI* enthält.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *MatrixI* 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 241).

Hinweis: *MatrixI* muss mindestens zwei Zeilen enthalten.

$\text{varSamp}(\{1,2,5,-6,3,-2\})$	$\frac{31}{2}$
$\text{varSamp}(\{1,3,5\},\{4,6,2\})$	$\frac{68}{33}$

$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right)$	$[4.75 \ 1.03 \ 4]$
$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$	$[3.91731 \ 2.08411]$


WaitKatalog > **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 rundet diesen Wert auf die nächsten 0,1 Sekunden auf.

Zum Abbrechen eines **Wait** das gerade durchgeführt wird,

- **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.

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"

warnCodes ()Katalog > 

warnCodes(*Ausdr1*,
StatusVar)⇒*Ausdruck*

```
warnCodes(det([1.23456E-999]),warn)
1.23456E-999
warn { 10029 }
```


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 260).

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.

$\text{when}(x < 0, x + 3), x = 5$	undef
------------------------------------	-------

$\text{when}(n > 0, n \cdot \text{factorial}(n - 1), 1) \rightarrow \text{factorial}(n)$	Done
$\text{factorial}(3)$	6
3!	6

While *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 Produkthandbuchs.

 Define $sum_of_recip(n)$ = Func
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 $tempsum$

EndFunc

Done

 $sum_of_recip(3)$

11

6**X****xor** (Boolesches exklusives oder)

Boolescher Ausdr1 **xor** *Boolescher Ausdr2*
ergibt *Boolescher Ausdruck*

true xor true

false

5 > 3 xor 3 > 5

true

Boolesche Liste1 **xor** *Boolesche Liste2*
ergibt *Boolesche Liste*

Boolesche Matrix1 **xor** *Boolesche Matrix2*
ergibt *Boolesche Matrix*

Gibt wahr zurück, wenn *Boolescher Ausdr1* wahr und *Boolescher Ausdr2* 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 121.

Ganzzahl1 **xor** *Ganzzahl2* \Rightarrow *Ganzzahl*

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O

0h7AC36 xor 0h3D5F

0h79169

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).

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 121.

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.

Z

zInterval (z-Konfidenzintervall)

zInterval σ ,*Liste*[,*Häufigkeit*[,*K.Stufe*]]

(Datenlisteneingabe)

zInterval σ , \bar{x} ,*n* [,*K.Stufe*]

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 166.)

zInterval (z-Konfidenzintervall)

Katalog > 

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekanntem Populationsmittelwert
stat.x̄	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)

Katalog > 

zInterval_1Prop $x, n [, KStufe]$

Berechnet ein z -Konfidenzintervall für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 166.)

x ist eine nicht negative Ganzzahl.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat.ĥ	Die berechnete Erfolgsproportion
stat.ME	Fehlertoleranz
stat.n	Anzahl der Stichproben in Datenfolge

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen)

Katalog > 

zInterval_2Prop $x1, n1, x2, n2 [, KStufe]$

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen)

Katalog > 

Berechnet das z -Konfidenzintervall für zwei Proportionen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 166.)

$x1$ und $x2$ sind nicht negative Ganzzahlen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

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
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 166.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

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. $\sigma x1$, stat. $\sigma x2$	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 $\mu0, \sigma, Liste, [Häufigkeit[, Hypoth]]$

(Datenlisteneingabe)

zTest $\mu0, \sigma, \bar{x}, n[, Hypoth]$

(Zusammenfassende statistische Eingabe)

Führt einen z -Test mit der Häufigkeit *Häufigkeitsliste* durch. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 166.)

Getestet wird $H_0: \mu = \mu_0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu < \mu_0$ setzen Sie *Hypoth*<0

Für $H_a: \mu \neq \mu_0$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \mu > \mu_0$ setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

Ausgabevariable	Beschreibung
stat.z	$(\bar{x} - \mu_0) / (\sigma / \text{sqrt}(n))$

Ausgabevariable	Beschreibung
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)

Katalog > 

zTest_1Prop $p_0, x, n, [Hypoht]$

Berechnet einen z -Test für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 166.)

x ist eine nicht negative Ganzzahl.

Getestet wird $H_0: p = p_0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: p > p_0$ setzen Sie *Hypoht*>0

Für $H_a: p \neq p_0$ (*Standard*) setzen Sie *Hypoht*=0

Für $H_a: p < p_0$ setzen Sie *Hypoht*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

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 166.)

$x1$ und $x2$ sind nicht negative Ganzzahlen.

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 $H_a: p < p0$ setzen Sie *Hypoth*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 241).

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 166.)

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 241).

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>
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**

<i>Wert1 + Wert2</i> ⇒ <i>Wert</i>	56	56
Gibt die Summe der beiden Argumente zurück.	56+4	60
	60+4	64
	64+4	68
	68+4	72

+ (addieren)**+ Taste** $Liste1 + Liste2 \Rightarrow Liste$

$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow I1$	$\{ 22, 3.14159, 1.5708 \}$
----------------------------------------------------------	-----------------------------

 $Matrix1 + Matrix2 \Rightarrow Matrix$

$\left\{ 10, 5, \frac{\pi}{2} \right\} \rightarrow I2$	$\{ 10, 5, 1.5708 \}$
--------------------------------------------------------	-----------------------

Gibt eine Liste (bzw. eine Matrix) zurück, die die Summen der entsprechenden Elemente von *Liste1* und *Liste2* (oder *Matrix1* und *Matrix2*) enthält.

$I1+I2$	$\{ 32, 8.14159, 3.14159 \}$
---------	------------------------------

Die Argumente müssen die gleiche Dimension besitzen.

 $Wert + Liste1 \Rightarrow Liste$

$15 + \{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
-------------------------	--------------------

 $Liste1 + Wert \Rightarrow Liste$

$\{ 10, 15, 20 \} + 15$	$\{ 25, 30, 35 \}$
-------------------------	--------------------

Gibt eine Liste zurück, die die Summen von *Wert* plus jedem Element der *Liste1* enthält.

 $Wert + Matrix1 \Rightarrow Matrix$

$20 + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
-----------------------------------------------------	--------------------------------------------------

 $Matrix1 + Wert \Rightarrow Matrix$

Gibt eine Matrix zurück, in der *Wert* zu jedem Element der Diagonalen von *Matrix1* addiert ist. *Matrix1* muss eine quadratische Matrix sein.

Hinweis: Verwenden Sie **+** (Punkt Plus) zum Addieren eines Ausdrucks zu jedem Element.

-(subtrahieren)**- Taste** $Wert1 - Wert2 \Rightarrow Wert$

$6 - 2$	4
---------	-----

Gibt *Wert1* minus *Wert2* zurück.

$\pi - \frac{\pi}{6}$	2.61799
-----------------------	-----------

 $Liste1 - Liste2 \Rightarrow Liste$

$\left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10, 5, \frac{\pi}{2} \right\}$	$\{ 12, -1.85841, 0. \}$
-----------------------------------------------------------------------------------	--------------------------

 $Matrix1 - Matrix2 \Rightarrow Matrix$

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

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.

-(subtrahieren)** Taste** $Wert - Liste1 \Rightarrow Liste$

$$15 - \{10, 15, 20\} \quad \{5, 0, -5\}$$

 $Liste1 - Wert \Rightarrow Liste$

$$\{10, 15, 20\} - 15 \quad \{-5, 0, 5\}$$

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.

 $Liste1 \cdot Liste2 \Rightarrow Liste$

$$\{1, 2, 3\} \cdot \{4, 5, 6\} \quad \{4, 10, 18\}$$

Gibt eine Liste zurück, die die Produkte der entsprechenden Elemente aus *Liste1* und *Liste2* enthält.

Die Listen müssen die gleiche Dimension besitzen.

 $Matrix1 \cdot Matrix2 \Rightarrow Matrix$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 7 & 8 \\ 7 & 8 \end{bmatrix} \quad \begin{bmatrix} 42 & 48 \\ 105 & 120 \end{bmatrix}$$

Gibt das Matrizenprodukt von *Matrix1* und *Matrix2* zurück.

Die Spaltenanzahl von *Matrix1* muss gleich die Zeilenanzahl von *Matrix2* sein.

·(multiplizieren)**[x] Taste** $Wert \cdot Liste1 \Rightarrow Liste$

$$\pi \cdot \{4,5,6\} \quad \{12.5664, 15.708, 18.8496\}$$

 $Liste1 \cdot Wert \Rightarrow Liste$

Gibt eine Liste zurück, die die Produkte von *Wert* und jedem Element der *Liste1* enthält.

 $Wert \cdot Matrix1 \Rightarrow Matrix$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 \quad \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

 $Matrix1 \cdot Wert \Rightarrow Matrix$

Gibt eine Matrix zurück, die die Produkte von *Wert* und jedem Element der *Matrix1* enthält.

$$6 \cdot \text{identity}(3) \quad \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Hinweis: Verwenden Sie \cdot (Punkt-Multiplikation) zum Multiplizieren eines Ausdrucks mit jedem Element.

/ (dividieren)**[÷] Taste** $Wert1 / Wert2 \Rightarrow Wert$

$$\frac{2}{3.45} \quad 0.57971$$

Gibt *Wert1* dividiert durch *Wert2* zurück.

Hinweis: Siehe auch **Vorlage Bruch**, Seite 1.

 $Liste1 / Liste2 \Rightarrow Liste$

$$\frac{\{1,2,3\}}{\{4,5,6\}} \quad \left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$$

Gibt eine Liste der Elemente von *Liste1* dividiert durch *Liste2* zurück.

Die Listen müssen die gleiche Dimension besitzen.

 $Wert / Liste1 \Rightarrow Liste$

$$\frac{6}{\{3,6,\sqrt{6}\}} \quad \{2, 1, 2.44949\}$$

 $Liste1 / Wert \Rightarrow Liste$

$$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2} \quad \left\{\frac{1}{18}, \frac{1}{14}, \frac{1}{63}\right\}$$

Gibt eine Liste der Elemente von *Wert* dividiert durch *Liste1* oder *Liste1* dividiert durch *Wert* zurück.

 $Wert / Matrix1 \Rightarrow Matrix$

$$\frac{\{7 \ 9 \ 2\}}{7 \cdot 9 \cdot 2} \quad \left[\frac{1}{18} \ \frac{1}{14} \ \frac{1}{63}\right]$$

 $Matrix1 / Wert \Rightarrow Matrix$

Gibt eine Matrix zurück, die die Quotienten *Matrix1*/*Wert* enthält.

/ (dividieren)



Taste

Hinweis: Verwenden Sie \div (Punkt-Division) zum Dividieren eines Ausdrucks durch jedes Element.

^ (Potenz)



Taste

$Wert1 \wedge Wert2 \Rightarrow Wert$

$$4^2 \qquad 16$$

$Liste1 \wedge Liste2 \Rightarrow Liste$

$$\{2,4,6\} \{1,2,3\} \qquad \{2,16,216\}$$

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.

$Wert \wedge Liste1 \Rightarrow Liste$

$$\pi \{1,2,-3\} \qquad \{3.14159,9.8696,0.032252\}$$

Gibt *Wert* hoch den Elementen von *Liste1* zurück.

$Liste1 \wedge Wert \Rightarrow Liste$

$$\{1,2,3,4\}^{-2} \qquad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$$

Gibt die Elemente von *Liste1* hoch *Wert* zurück.

$Quadratmatrix1 \wedge Ganzzahl \Rightarrow Matrix$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 \qquad \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

Gibt *Quadratmatrix1* hoch *Ganzzahl* zurück.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \qquad \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

Quadratmatrix1 muss eine quadratische Matrix sein.

Ist *Ganzzahl* = -1, wird die inverse Matrix berechnet.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \qquad \begin{bmatrix} 11 & -5 \\ 2 & 2 \\ -15 & 7 \\ 4 & 4 \end{bmatrix}$$

Ist *Ganzzahl* < -1, wird die inverse Matrix hoch der entsprechenden positiven Zahl berechnet.

x² (Quadrat)

x^2 Taste

*Wert1*² ⇒ *Wert*

Gibt das Quadrat des Arguments zurück.

*Liste1*² ⇒ *Liste*

Gibt eine Liste zurück, die die Produkte der Elemente in *Liste1* enthält.

*Quadratmatrix1*² ⇒ *Matrix*

Gibt das Matrix-Quadrat von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Quadrats jedes einzelnen Elements. Verwenden Sie $\wedge 2$, um das Quadrat jedes einzelnen Elements zu berechnen.

4^2	16
$\{2,4,6\}^2$	$\{4,16,36\}$
$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2$	$\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$
$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} \wedge 2$	$\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$

.+ (Punkt-Addition)

$\cdot +$ Tasten

Matrix1 .+ *Matrix2* ⇒ *Matrix*

Wert .+ *Matrix1* ⇒ *Matrix*

Matrix1 .+ *Matrix2* gibt eine Matrix zurück, die die Summe jedes Elementpaares von *Matrix1* und *Matrix2* ist.

Wert .+ *Matrix1* gibt eine Matrix zurück, die die Summe von *Wert* und jedem Element von *Matrix1* ist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot + \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$
$5 \cdot + \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$

.- (Punkt-Subt.)

$\cdot -$ Tasten

Matrix1 .- *Matrix2* ⇒ *Matrix*

Wert .- *Matrix1* ⇒ *Matrix*

Matrix1 .- *Matrix2* gibt eine Matrix zurück, die die Differenz jedes Elementpaares von *Matrix1* und *Matrix2* ist.

Wert .- *Matrix1* gibt eine Matrix zurück, die die Differenz von *Wert* und jedem Element von *Matrix1* ist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	$\begin{bmatrix} -9 & -18 \\ -27 & -36 \end{bmatrix}$
$5 \cdot - \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	$\begin{bmatrix} -5 & -15 \\ -25 & -35 \end{bmatrix}$

.· (Punkt-Mult.)

\cdot	\times	Tasten
---------	----------	--------

Matrix1 \cdot *Matrix2* \Rightarrow *Matrix*

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	\cdot	$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$
------------------------------------------------	---------	----------------------------------------------------	---------------	-----------------------------------------------------

Wert \cdot *Matrix1* \Rightarrow *Matrix*

5	\cdot	$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} 50 & 100 \\ 150 & 200 \end{bmatrix}$
---	---------	----------------------------------------------------	---------------	-------------------------------------------------------

Matrix1 \cdot *Matrix2* gibt eine Matrix zurück, die das Produkt jedes Elementpaares von *Matrix1* und *Matrix2* ist.

Wert \cdot *Matrix1* Gibt eine Matrix zurück, die die Produkte von *Wert* und jedem Element der *Matrix1* enthält.

./ (Punkt-Division)

\cdot	\div	Tasten
---------	--------	--------

Matrix1 \cdot / *Matrix2* \Rightarrow *Matrix*

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	\cdot /	$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{10} \end{bmatrix}$
------------------------------------------------	-----------	----------------------------------------------------	---------------	--------------------------------------------------------------------------------------------

Wert \cdot / *Matrix1* \Rightarrow *Matrix*

5	\cdot /	$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{8} \end{bmatrix}$
---	-----------	----------------------------------------------------	---------------	----------------------------------------------------------------------------------------

Matrix1 \cdot / *Matrix2* gibt eine Matrix zurück, die der Quotient jedes Elementpaares von *Matrix1* und *Matrix2* ist.

Wert \cdot / *Matrix1* gibt eine Matrix zurück, die der Quotient von *Wert* und jedem Element von *Matrix1* ist.

.^ (Punkt-Potenz)

\cdot	\wedge	Tasten
---------	----------	--------

Matrix1 \cdot ^ *Matrix2* \Rightarrow *Matrix*

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	\cdot ^	$\begin{bmatrix} 0 & 2 \\ 3 & -1 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} 1 & 4 \\ 27 & \frac{1}{4} \end{bmatrix}$
------------------------------------------------	-----------	-------------------------------------------------	---------------	-----------------------------------------------------------

Wert \cdot ^ *Matrix1* \Rightarrow *Matrix*

5	\cdot ^	$\begin{bmatrix} 0 & 2 \\ 3 & -1 \end{bmatrix}$	\Rightarrow	$\begin{bmatrix} 1 & 25 \\ 125 & \frac{1}{5} \end{bmatrix}$
---	-----------	-------------------------------------------------	---------------	-------------------------------------------------------------

Matrix1 \cdot ^ *Matrix2* gibt eine Matrix zurück, in der jedes Element aus *Matrix2* Exponent des entsprechenden Elements aus *Matrix1* ist.

Wert \cdot ^ *Matrix1* gibt eine Matrix zurück, in der jedes Element aus *Matrix1* Exponent von *Wert* ist.

-(Negation)

 Taste

$-Wert1 \Rightarrow Wert$

-2.43 -2.43

$-Liste1 \Rightarrow Liste$

$\{-1,0.4,1.2E19\}$ $\{1,-0.4,-1.2E19\}$

$-Matrix1 \Rightarrow Matrix$

Gibt die Negation des Arguments zurück.

Im Bin-Modus:

Bei einer Liste oder Matrix werden alle Elemente negiert zurückgegeben.

Wichtig: Null, nicht Buchstabe O

Ist das Argument eine binäre oder hexadezimale ganze Zahl, ergibt die Negation das Zweierkomplement.

-0b100101
0b11111111111111111111111111111111

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

%(Prozent)

  Tasten

$Wert1 \% \Rightarrow Wert$

$Liste1 \% \Rightarrow Liste$

$Matrix1 \% \Rightarrow Matrix$

argument

Ergibt 100


Bei einer Liste oder einer Matrix wird eine Liste/Matrix zurückgegeben, in der jedes Element durch 100 dividiert ist.

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie  .

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie  aus.

13% 0.13

$\{\{1,10,100\}\}\%$ $\{0.01,0.1,1\}$

= (gleich)

 Taste

$Ausdr1 = Ausdr2 \Rightarrow$ Boolescher Ausdruck

Beispielfunktion mit den mathematischen Vergleichssymbolen: =, ≠, <, ≤, >, ≥

$Liste1 = Liste2 \Rightarrow$ Boolesche Liste

$Matrix1 = Matrix2 \Rightarrow$ Boolesche Matrix

Gibt wahr zurück, wenn $Ausdr1$ bei Auswertung gleich $Ausdr2$ ist.

= (gleich)

 **Taste**

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.

Hinweis zum Eingeben des Beispiels:

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

Define $g(x)=$ Func

If $x \leq -5$ Then

Return 5

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

Return $-x$

ElseIf $x \geq 0$ and $x \neq 10$ Then

Return x

ElseIf $x = 10$ Then

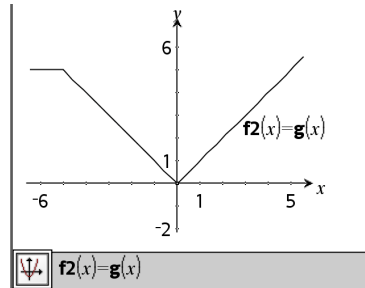
Return 3

EndIf

EndFunc

Done

Ergebnis der graphischen Darstellung $g(x)$



≠ (ungleich)

 **Tasten**

$Ausdr1 \neq Ausdr2 \Rightarrow$ Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

$Liste1 \neq Liste2 \Rightarrow$ Boolesche Liste

$Matrix1 \neq Matrix2 \Rightarrow$ Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung ungleich *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

≠ (ungleich)

ctrl  Tasten

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.

≤ (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.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung größer als *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

\leq (kleiner oder gleich)

  Tasten

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 \leq =

$>$ (größer als)

  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)

  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.

Gibt falsch zurück, wenn $Ausdr1$ bei Auswertung kleiner oder gleich $Ausdr2$ ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

≥ (größer oder gleich)

ctrl  Tasten

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 >=

⇒ (logische Implikation)

ctrl  Tasten

BoolescherAusdr1 ⇒ *BoolescherAusdr2*
ergibt *Boolescher Ausdruck*

5>3 or 3>5	true
------------	------

5>3 ⇒ 3>5	false
-----------	-------

BoolescheListe1 ⇒ *BoolescheListe2*
ergibt *Boolesche Liste*

3 or 4	7
--------	---

3 ⇒ 4	-4
-------	----

BoolescheMatrix1 ⇒
BoolescheMatrix2 ergibt *Boolesche Matrix*

{1,2,3} or {3,2,1}	{3,2,3}
--------------------	---------

{1,2,3} ⇒ {3,2,1}	{-1,-1,-3}
-------------------	------------

Ganzzahl1 ⇒ *Ganzzahl2* ergibt
Ganzzahl

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 =>

⇔ (logische doppelte Implikation, XNOR)

  Tasten

BoolescherAusdr1 ⇔
BoolescherAusdr2 ergibt *Boolescher Ausdruck*

$5 > 3 \text{ xor } 3 > 5$	true
----------------------------	------

$5 > 3 \Leftrightarrow 3 > 5$	false
-------------------------------	-------

BoolescheListe1 ⇔ *BoolescheListe2*
ergibt *Boolesche Liste*

$3 \text{ xor } 4$	7
--------------------	---

$3 \Leftrightarrow 4$	-8
-----------------------	----

BoolescheMatrix1 ⇔
BoolescheMatrix2 ergibt *Boolesche Matrix*

$\{1,2,3\} \text{ xor } \{3,2,1\}$	$\{2,0,2\}$
------------------------------------	-------------

$\{1,2,3\} \Leftrightarrow \{3,2,1\}$	$\{-3,-1,-3\}$
---------------------------------------	----------------

Ganzzahl1 ⇔ *Ganzzahl2* ergibt
Ganzzahl

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 \Leftrightarrow drücken

! (Fakultät)

 Taste

Wert1! ⇒ *Wert*

$5!$	120
------	-----

Liste1! ⇒ *Liste*

$\{\{5,4,3\}\}!$	$\{120,24,6\}$
------------------	----------------

Matrix1! ⇒ *Matrix*

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$	$\begin{pmatrix} 1 & 2 \\ 6 & 24 \end{pmatrix}$
-------------------------------------------------	-------------------------------------------------

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* ⇒ *String*

"Hello "&"Nick"

"Hello Nick"

Gibt einen String zurück, der durch Anfügen von *String2* an *String1* gebildet wurde.

d() (Ableitung)Katalog > 

d(Ausdr1, Var[, Ordnung]) |
Var=Wert⇒*Wert*

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

d(Ausdr1, Var[, Ordnung])⇒*Wert*

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

d(Liste1, Var[, Ordnung])⇒*Liste*

$$x:=3: \frac{d}{dx}(\{x^2, x^3, x^4\}) \quad \{6, 27, 108\}$$

d(Matrix1, Var[, Ordnung])⇒*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.

$$\frac{d}{dx} \left(x \cdot (x^2 + x)^{\frac{1}{3}} \right) |_{x=0} \quad \text{undef}$$

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

d() (Ableitung)

Katalog > 

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.

Wenn Sie bei der Arbeit auf diese Beschränkung stoßen, prüfen Sie die Lösung grafisch. Ggf. können Sie es auch mit **centralDiff()** probieren.

∫() (Integral)

Katalog > 

$\int(\text{Ausdr1}, \text{Var}, \text{Untere}, \text{Obere}) \Rightarrow \text{Wert}$

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.

$\int_0^1 x^2 dx$	0.333333
-------------------	----------

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

Hinweis: Siehe auch **nInt()**, Seite 114, und **Vorlage Bestimmtes Integral**, Seite 6.

√() (Quadratwurzel)

  Tasten

$\sqrt{(\text{Wert})} \Rightarrow \text{Wert}$

$\sqrt{4}$	2
------------	---

$\sqrt{(\text{Liste})} \Rightarrow \text{Liste}$

$\sqrt{\{9,2,4\}}$	$\{3,1.41421,2\}$
--------------------	-------------------

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste* zurückgegeben.

$\sqrt{}$ (Quadratwurzel)

ctrl x² Tasten

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

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 1.

\prod (ProdSeq)

Katalog > 

$\prod(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Bis}) \Rightarrow \text{Ausdruck}$

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 (\prod)**, Seite 5.

$\prod(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Von}-1) \Rightarrow 1$

$\prod(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Bis}) \Rightarrow 1/\prod(\text{Ausdr1}, \text{Var}, \text{Bis}+1, \text{Von}-1)$ if $\text{Bis} < \text{Von}-1$

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.

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

$$\prod_{n=1}^5 \left\{ \left\{ \frac{1}{n}, n, 2 \right\} \right\} = \left\{ \frac{1}{120}, 120, 32 \right\}$$

$$\prod_{k=4}^3 (k) = 1$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) = 6$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) \cdot \prod_{k=2}^4 \left(\frac{1}{k}\right) = \frac{1}{4}$$

Σ (SumSeq)

Katalog > 

$\Sigma(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Bis}) \Rightarrow \text{Ausdruck}$

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

$$\sum_{n=1}^5 \left(\frac{1}{n}\right) = \frac{137}{60}$$

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.

$$\Sigma(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Von}-1) \Rightarrow 0$$

$$\Sigma(\text{Ausdr1}, \text{Var}, \text{Von}, \text{Bis}) \Rightarrow -\Sigma(\text{Ausdr1}, \text{Var}, \text{Bis}+1, \text{Von}-1) \text{ if } \text{Bis} < \text{Von}-1$$

$$\sum_{k=4}^3 (k) = 0$$

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.

$$\sum_{k=4}^1 (k) = -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) = 4$$

$\Sigma\text{Int}()$

$\Sigma\text{Int}(\text{NPmt1}, \text{NPmt2}, \text{N}, \text{I}, \text{PV}, [\text{Pmt}], [\text{FV}], [\text{PpY}], [\text{CpY}], [\text{PmtAt}], [\text{WertRunden}]) \Rightarrow \text{Wert}$

$$\Sigma\text{Int}(1, 3, 12, 4.75, 20000, , 12, 12) = -213.48$$

$\Sigma\text{Int}(\text{NPmt1}, \text{NPmt2}, \text{AmortTabelle}) \Rightarrow \text{Wert}$

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 183) beschrieben.

- Wenn Sie *Pmt* nicht angeben, wird standardmäßig $\text{Pmt} = \text{tvmPmt}(N, I, PV, FV, PpY, CpY, PmtAt)$ eingesetzt.
- Wenn Sie *FV* nicht angeben, wird standardmäßig $FV = 0$ eingesetzt.

tbl := $\text{amortTbl}(12, 12, 4.75, 20000, , 12, 12)$

0	0.	0.	20000.
1	-77.49	-1632.43	18367.6
2	-71.17	-1638.75	16728.8
3	-64.82	-1645.1	15083.7
4	-58.44	-1651.48	13432.2
5	-52.05	-1657.87	11774.4
6	-45.62	-1664.3	10110.1
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	6.55	-1703.37	-12.02

$\Sigma\text{Int}(1, 3, \text{tbl}) = -213.48$

- 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.

$\Sigma\text{Int}(NPmt1, NPmt2, AmortTable)$ 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.

Hinweis: Siehe auch $\Sigma\text{Prn}()$ auf dieser und **Bal()**, Seite 16.

 $\Sigma\text{Prn}()$

$\Sigma\text{Prn}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) \Rightarrow Wert$

$\Sigma\text{Prn}(1,3,12,4,75,20000,,12,12)$	-4916.28
----------------------------------------------	----------

 ΣPrn

$(NPmt1, NPmt2, AmortTabelle) \Rightarrow Wert$

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 183) 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.

$tbl := \text{amortTbl}(12, 12, 4, 75, 20000, 12, 12)$

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

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

$\Sigma\text{Prn}(\text{NPmt1}, \text{NPmt2}, \text{AmortTabelle})$ berechnet die Summe der gezahlten Tilgungsbeträge auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle* muss eine Matrix in der unter **amortTbl()**, Seite 7, beschriebenen Form sein.

Hinweis: Siehe auch $\Sigma\text{Int}()$ auf dieser und **Bal()**, Seite 16.

(Umleitung)

  **Tasten**

varNameString

Greift auf die Variable namens *VarNameString* zu. So können Sie innerhalb einer Funktion Variablen unter Verwendung von Strings erzeugen.

<code>xyz:=12</code>	12
----------------------	----

<code>#("x"&"y"&"z")</code>	12
-------------------------------------	----

Erzeugt oder greift auf die Variable xyz zu.

<code>10→r</code>	10
-------------------	----

<code>"r"→s1</code>	"r"
---------------------	-----

<code>#s1</code>	10
------------------	----

Gibt den Wert der Variable (r) zurück, dessen Name in Variable s1 gespeichert ist.

E (Wissenschaftliche Schreibweise)

 **Taste**

*Mantisse***EE***Exponent*

Gibt eine Zahl in wissenschaftlicher Schreibweise ein. Die Zahl wird als *Mantisse* × 10^{*Exponent*} interpretiert.

<code>23000.</code>	23000.
---------------------	--------

<code>2300000000.+4.1E15</code>	4.1E15
---------------------------------	--------

<code>3·10⁴</code>	30000
-------------------------------	-------

Tipp: Wenn Sie eine Potenz von 10 eingeben möchten, ohne ein Dezimalwettergebnis zu verursachen, verwenden Sie 10^{*Ganzzahl*}.

E (Wissenschaftliche Schreibweise)

 Taste

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)

 Taste

Ausdr1g ⇒ *Ausdruck*

Im Grad-, Neugrad- oder Bogenmaß-Modus:

Ausdr1g ⇒ *Ausdruck*

$\cos(50^{\circ})$	0.707107
--------------------	----------

Liste1g ⇒ *Liste*

$\cos\left(\left\{0, 100^{\circ}, 200^{\circ}\right\}\right)$	$\{1, 0, -1\}$
---------------------------------------------------------------	----------------

Matrix1g ⇒ *Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Bogenmaß-Modus einen Winkel in Neugrad anzugeben.

Im Winkelmodus Bogenmaß wird *Ausdr1* mit $\pi/200$ multipliziert.

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.

r (Bogenmaß)

 Taste

Wert1r ⇒ *Wert*

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

Liste1r ⇒ *Liste*

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

Matrix1r ⇒ *Matrix*

$\cos\left(\left\{0^r, \left(\frac{\pi}{12}\right)^r, -(\pi)^r\right\}\right)$	$\{1, 0.965926, -1\}$
--------------------------------------------------------------------------------	-----------------------

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Neugrad-Modus einen Winkel im Bogenmaß anzugeben.

⌠ (Bogenmaß)

1 Taste

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 @x eintippen.

° (Grad)

1 Taste

*Wert*1°⇒*Wert*

*Liste*1°⇒*Liste*

*Matrix*1°⇒*Matrix*

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.

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

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

Im Winkelmodus Bogenmaß:

°, ', " (Grad/Minute/Sekunde)

ctrl  Tasten

$dd^{\circ}mm'ss.ss'' \Rightarrow$ Ausdruck

dd Eine positive oder negative Zahl

mm Eine nicht negative Zahl

$ss.ss$ Eine nicht negative Zahl

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 (").

Im Grad-Modus:

$25^{\circ}13'17.5''$	25.2215
$25^{\circ}30'$	$\frac{51}{2}$

∠ (Winkel)

ctrl  Tasten

$[Radius, \angle _ Winkel] \Rightarrow$ Vektor

(Eingabe polar)

$[Radius, \angle _ Winkel, Z_$
 $Koordinate] \Rightarrow$ Vektor

(Eingabe zylindrisch)

$[Radius, \angle _ Winkel, \angle _$
 $Winkel] \Rightarrow$ 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 $\Theta <$ eintippen.

$(Größe \angle Winkel) \Rightarrow$ komplexer Wert

Im Bogenmaß-Modus mit Vektorformat eingestellt auf:

kartesisch

$[5 \angle 60^{\circ} \angle 45^{\circ}]$
$[1.76777 \quad 3.06186 \quad 3.53553]$

zylindrisch

$[5 \angle 60^{\circ} \angle 45^{\circ}]$
$[3.53553 \quad \angle 1.0472 \quad 3.53553]$

sphärisch

$[5 \angle 60^{\circ} \angle 45^{\circ}]$
$[5. \quad \angle 1.0472 \quad \angle 0.785398]$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

∠ (Winkel)

ctrl  Tasten

(Eingabe polar)

$$5+3\cdot i \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107\cdot i$$

Dient zur Eingabe eines komplexen Werts in polarer ($r\angle\theta$) Form. Der *Winkel* wird gemäß der aktuellen WinkelmodusEinstellung interpretiert.

$$5+3\cdot i \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107\cdot i$$

_ (Unterstrich als leeres Element)

Siehe "Leere (ungültige) Elemente", Seite 241.

10^()

Katalog > 

$10^{\text{Wert}1} \Rightarrow \text{Wert}$

$$10^{1.5} \quad 31.6228$$

$10^{\text{Liste}1} \Rightarrow \text{Liste}$

Gibt 10 hoch Argument zurück.

Bei einer Liste wird 10 hoch jedem Element von *Liste1* zurückgegeben.

$10^{\text{Quadratmatrix}1} \Rightarrow \text{Quadratmatrix}$

Ergibt 10 hoch *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von 10 hoch jedem Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$$10^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}} \quad \begin{bmatrix} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

^-1 (Kehrwert)

Katalog > 

$\text{Wert}1 \wedge^{-1} \Rightarrow \text{Wert}$

$$(3.1)^{-1} \quad 0.322581$$

$\text{Liste}1 \wedge^{-1} \Rightarrow \text{Liste}$

Gibt den Kehrwert des Arguments zurück.

Bei einer Liste wird für jedes Element von *Liste1* der Kehrwert zurückgegeben.

Quadratmatrix1 $\wedge^{-1} \Rightarrow$ 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)

Ausdr | *BoolescherAusdr1*
[and*BoolescherAusdr2*]...

$x+1 x=3$	4
$x+55 x=\sin(55)$	54.0002

Ausdr | *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:

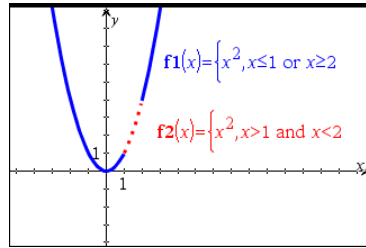
- Ersetzung
- Intervallbeschränkung
- Ausschließung

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.

$x^3-2 \cdot x+7 \rightarrow f(x)$	Done
$f(x) x=\sqrt{3}$	8.73205

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.

$\text{nSolve}(x^3+2 \cdot x^2-15 \cdot x=0,x)$	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“ (\neq oder \neq), um einen bestimmten Wert bei der Operation auszuschließen.

→ (speichern)

Wert → Var

Liste → Var

Matrix → Var

Expr → Funktion(Param1,...)

List → Funktion(Param1,...)

Matrix → Funktion(Param1,...)

$\frac{\pi}{4} \rightarrow myvar$	0.785398
$2 \cdot \cos(x) \rightarrow yI(x)$	Done
$\{1,2,3,4\} \rightarrow lst5$	$\{1,2,3,4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → str1	"Hello"

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* oder *Matrix* ersetzt.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel =: eintippen. Geben Sie zum Beispiel $\pi/4 \text{ =: } myvar$ ein.

:= (zuweisen)ctrl  Tasten*Var := Wert*

$myvar := \frac{\pi}{4}$.785398
--------------------------	---------

Var := Liste

$y1(x) := 2 \cdot \cos(x)$	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  Tasten© [*Text*]

© 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.

Hinweis zum Eingeben des Beispiels:

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

Define $g(n) = \text{Func}$ © *Declare variables*Local *i,result**result*:=0For *i*,1,*n*,1 ©Loop *n times**result*:=*result*+*i*²



EndFor

Return *result*

EndFunc

Done

$g(3)$	14
--------	----

Ob, Oh  Tasten,   Tasten**Ob** *binäre_Zahl*

Im Dec-Modus:

Oh *hexadezimale_Zahl*

$Ob10+OhF+10$	27
---------------	----

Ob, Oh

O **B** Tasten, **O** **H** Tasten

Kennzeichnet eine Dual- bzw. Hexadezimalzahl. Zur Eingabe einer Dual- oder Hexadezimalzahl muss unabhängig vom jeweiligen Basis-Modus das Präfix Ob bzw. Oh verwendet werden. Eine Zahl ohne Präfix wird als dezimal behandelt (Basis 10).

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

Im Bin-Modus:

Ob10+0hF+10	Ob11011
-------------	---------

Im Hex-Modus:

Ob10+0hF+10	Oh1B
-------------	------

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-Dokumentensicht 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).

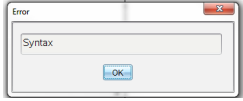
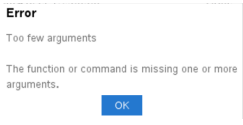
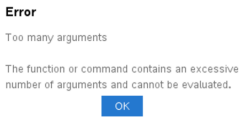
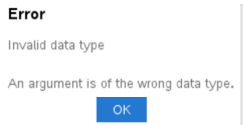
Grafikbildschirm	Standard
Höhe	212
Breite	318
Farbe	Weiß: 255,255,255

Standardansicht und Einstellungen

- Die Statussymbole in der oberen Symbolleiste (Batteriestatus, Prüfungsmodus-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	
Fehler Zu viele Argumente	Die Funktion oder der Befehl enthält zu viele Argumente und kann nicht ausgewertet werden.	
Fehler Ungültiger Datentyp	Ein Argument weist einen falschen Datentyp auf.	

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)**Clear** $x, y, \text{Breite}, \text{Höhe}$

Löschen

Löscht den gesamten Bildschirm, wenn keine Parameter angegeben wurden.

Löscht den gesamten Bildschirm

Werden x, y, Breite und Höhe angegeben, wird das durch die Parameter definierte Rechteck gelöscht.

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 $x, y, \text{Breite}, \text{Höhe}, \text{startAngle}, \text{arcAngle}$

Zeichnet einen Bogen innerhalb eines definierten begrenzenden Rechtecks mit dem angegebenen Start- und Bogenwinkel.

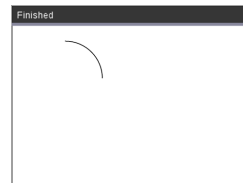
x, y : obere linke Koordinate des begrenzenden Rechtecks

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](#)

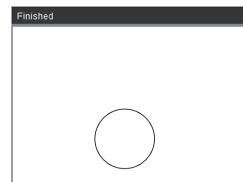
DrawCircle
 Katalog > 
CXII

DrawCircle x, y, Radius

x, y : Koordinate des Mittelpunkts

Radius: Radius des Kreises

DrawCircle 150,150,40



Siehe auch: [FillCircle](#)

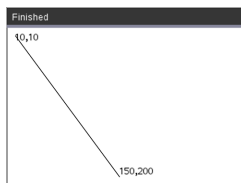
DrawLine $x1, y1, x2, y2$

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**

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 $x1, y1, x2, y2, x3, y3...xn, yn$ wird **NICHT** automatisch mit $x1, y1$ verbunden.

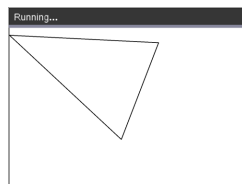
Ausdrücke, die eine Liste von realen Float-Variablen ergeben
 $xlist, ylist$

Ausdrücke, die eine reale einzelne Float-Variable ergeben
 $x1, y1...xn, yn$ = Koordinaten für Polygoneckpunkte

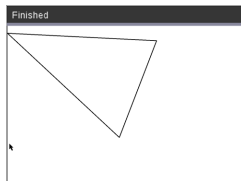
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

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*

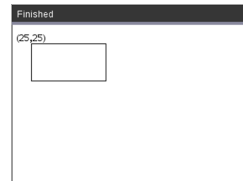
x, y: Obere linke Koordinate des Rechtecks

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

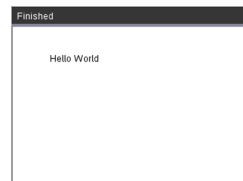
**DrawText****DrawText** *x, y, exprOrString1 [,exprOrString2]...*

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

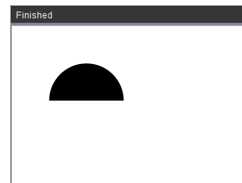
 Katalog >  CXII

FillArc *x, y, Breite, Höhe startAngle, arcAngle*

FillArc 50,50,100,100,0,180

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.

FillCircle

 Katalog >  CXII

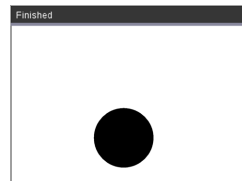
FillCircle *x, y, Radius*

FillCircle 150,150,40

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.



Hier!

FillPoly

 Katalog >  CXII

FillPoly *xlist, ylist*

xlist:={0,200,150,0}

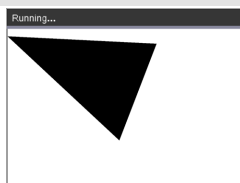
oder

ylist:={10,20,150,10}

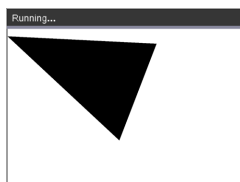
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

FillPoly *xlist,ylist*

Hinweis: Linie und Farbe werden durch [SetColor](#) und [SetPen](#) festgelegt.



```
FillPoly 0,10,200,20,150,150,0,10
```



FillRect

FillRect $x, y, \text{Breite}, \text{Höhe}$

x, y : Obere linke Koordinate des Rechtecks

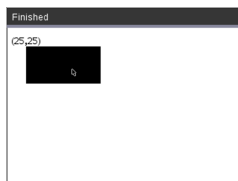
Breite, Höhe: Breite und Höhe des Rechtecks

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
```



G

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

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, unangefüllt
- 10 – Großer Punkt, ausgefüllt
- 11 – Großer Punkt, unangefüllt
- 12 – Größter Punkt, ausgefüllt
- 13 – Größter Punkt, unangefüllt

PlotXY 100,100,1

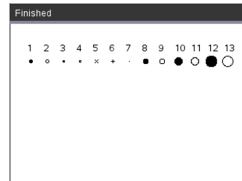



For n,1,13

DrawText 1+22*n,40,n

PlotXY 5+22*n,50,n

EndFor



F:**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

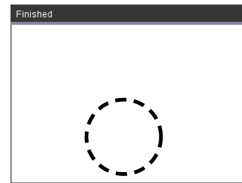
Dicke: 1 <= Dicke <= 3 | 1 ist am dünnsten, 3 ist am dicksten

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 Grafikzeichenbereich 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

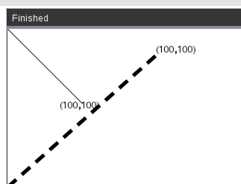
DrawLine 0,0,100,100

Ist x_{\min} größer oder gleich x_{\max} oder y_{\min} größer oder gleich y_{\max} , 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

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 84.

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  .

Kalkulationen mit ungültigen Elementen

Bei der Mehrzahl aller Kalkulationen, die ein ungültiges Element enthalten, wird das Ergebnis ebenfalls ungültig sein. Sonderfälle sind nachstehend aufgeführt.

$_$	-
$\text{gcd}(100,_)$	-
$3+_$	-
$\{5,_,10\}-\{3,6,9\}$	$\{2,_,1\}$

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
$\text{cumulativeSum}\{1,2,_,4,5\}$	$\{1,3,_,7,12\}$
$\text{cumulativeSum}\left(\begin{matrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{matrix}\right)$	$\begin{matrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{matrix}$

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	Done
list1	$\{1,3,4,5,_\}$
list2	$\{1,3,4,5,2\}$

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD list1,list2	Done
list1	$\{5,3,2,1,_\}$
list2	$\{5,3,2,1,4\}$

In Regressionen sorgt ein ungültiges Element in einer Liste X oder Y dafür, dass auch das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,2,3,4,5\}; l2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx l1,l2	Done
stat.Resid	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
stat.XReg	$\{1,_,3,4,5\}$
stat.YReg	$\{2,_,3,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1,1\}$

Eine ausgelassene Kategorie in Regressionen sorgt dafür, dass das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
$cat:=\{"M","M","F","F"\}; incl:=\{"F"\}$	$\{"F"\}$
LinRegMx l1,l2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1,1\}$

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}	Done
stat.Resid	$\{0.069231,_, -0.276923, 0.207692\}$
stat.XReg	$\{1,_,4,5\}$
stat.YReg	$\{2,_,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1\}$

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>
$\sin^{-1}()$, $\cos^{-1}()$, ...	<code>arcsin (...)</code> , <code>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:
$\overset{\frown}{\curvearrowright}$ (Bogenmaß)	@ r
$^{\circ}$ (Grad)	@ d
g (Neugrad)	@ g
\sphericalangle (Winkel)	@<
► (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 (°, ', "), Fakultät (!), Prozent (%), Bogenmaß (°), Tiefstellen ([]), Transponieren (T)
5	Potenzieren, Potenzoperator (^)
6	Negation (-)
7	Stringverkettung (&)
8	Multiplikation (•), Division (/)
9	Addition (+), Subtraktion (-)
10	Gleichheitsbeziehungen: gleich (=), ungleich (≠ oder /≠), kleiner als (<), kleiner oder gleich (≤ oder <=), größer als (>), größer oder gleich (≥ oder >=)
11	Logisches Nicht: not
12	Logische Konjunktion: and
13	Logisch or
14	xor, nor, nand
15	logische Implikation, (⇒)
16	Logische doppelte Implikation, XNOR (⇔)
17	womit-Operator („ “)
18	Speichern (→)

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. Anderenfalls 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 $\boxed{-}$ 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

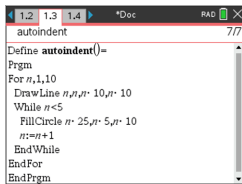
Automatisches Einrücken im Programmiereditor

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 Programmbeehlen automatisch Leerstellen voran. Der Schließbefehl des Blocks wird am Öffnungsbefehl ausgerichtet.

Das unten stehende Beispiel zeigt das automatische Einrücken in Befehlen mit verschachtelten Blöcken.



```
autoident 7/7
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

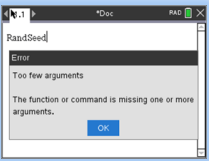
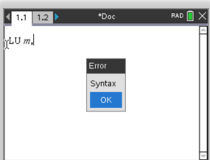
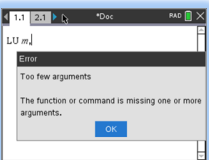
Fehler

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

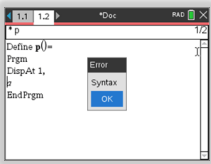
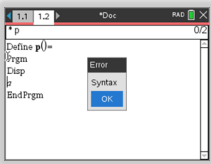
Fehlermeldungen	Neue Meldung
EndLoop fehlt	Erwartete EndLoop , fand aber eine andere End-Anweisung
EndTry fehlt	Erwartete EndTry , fand aber eine andere End-Anweisung
„Then“ nach If <condition> nicht angegeben	If..Then fehlt
„Then“ nach Elseif <condition> nicht angegeben	Then fehlt in Block: Elseif .
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“-Blöcks	Elseif ungültig außerhalb des Blocks: If..Then..Endif
„Then“ erscheint außerhalb des „If....Endif“-Blöcks	Then ungültig außerhalb des Blocks: If..Endif

Syntaxfehler

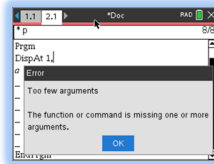
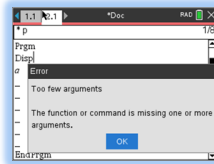
Wenn Befehle, die ein oder mehrere Argumente erfordern, mit einer unvollständigen Argumentenliste aufgerufen werden, wird anstelle eines „**Syntax**“-Fehlers ein „**Zu wenige Argumente**“-Fehler ausgegeben.

Aktuelles Verhalten	Neues CX II-Verhalten
	
	

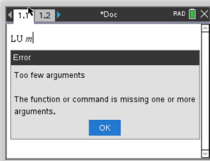
Aktuelles Verhalten




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
_Cc	Coulombsche Konstante	8987551787,3682 _m/_F
_Fc	Faraday-Konstante	96485,33289 _coul/_mol
_g	Erdbeschleunigung	9,80665 _m/_s ²
_Gc	Gravitationskonstante	6,67408E-11 _m ³ /_kg/_s ²
_h	Plancksche Konstante	6,626070040E-34 _J _s
_k	Boltzmann-Konstante	1,38064852E-23 _J/_°K
_μ0	Permeabilität des Vakuums	1,2566370614359E-6 _N/_A ²
_μb	Bohr-Magneton	9,274009994E-24 _J _m ² /_Wb
_Me	Ruhemasse des Elektrons	9,10938356E-31 _kg
_Mμ	Myonmasse	1,883531594E-28 _kg
_Mn	Ruhemasse des Neutrons	1,674927471E-27 _kg
_Mp	Ruhemasse des Protons	1,672621898E-27 _kg
_Na	Avogadro-Zahl	6,022140857E23 /_mol
_q	Elektronenladung	1,6021766208E-19 _coul
_Rb	Bohr-Radius	5,2917721067E-11 _m
_Rc	Molare Gaskonstante	8,3144598 _J/_mol/_°K
_Rdb	Rydberg-Konstante	10973731,568508/_m
_Re	Elektronenradius	2,8179403227E-15 _m
_u	Atommasse	1,660539040E-27 _kg
_Vm	Molvolumen	2,2413962E-2 _m ³ /_mol
_ε0	Permittivität des Vakuums	8,8541878176204E-12 _F/_m
_σ	Stefan-Boltzmann-Konstante	5,670367E-8 _W/_m ² /_°K ⁴
_φ0	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 179).

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 Setzen 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 <code>esc</code> oder <code>on</code> 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	Variablenname doppelt vergeben
280	Else und Elseif außerhalb If..EndIf-Block ungültig
290	Zu EndTry fehlt passende Else-Anweisung

Fehlercode	Beschreibung
295	Zu viele Iterationen
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

Fehlercode	Beschreibung
	Beispielsweise ist der Befehl Abbruch (Exit) nur in diesen Schleifenblöcken gültig.
565	Nur in einem Programm gültig
570	Ungültiger Pfadname \ <code>var</code> 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]

Fehlercode	Beschreibung
720	fehlt }
730	Anfang oder Ende des Blocks fehlt
740	Then im If..EndIf-Block fehlt
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

Fehlercode	Beschreibung
960	<p>Variable ist nicht definiert</p> <p>Der Variablen wurde kein Wert zugewiesen. Verwenden Sie einen der folgenden Befehle:</p> <ul style="list-style-type: none"> • sto → • := • Definiere <p>um Variablen Werte zuzuweisen.</p>
965	Betriebssystem nicht lizenziert
970	Variable ist aktiv, daher keine Verweise oder Änderungen zulässig
980	Variable ist geschützt
990	<p>Ungültiger Variablenname</p> <p>Stellen Sie sicher, dass der Name die maximale Zeichenlänge nicht überschreitet</p>
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	<p>Funktion ist nicht definiert. Verwenden Sie einen der folgenden Befehle:</p> <ul style="list-style-type: none"> • Definiere • := • sto → <p>um eine Funktion zu definieren.</p>

Fehlercode	Beschreibung
1100	<p>Nicht-reelle Berechnung</p> <p>Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{-1}$ ungültig.</p> <p>Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).</p>
1110	Ungültige Grenzen
1120	Keine Zeichenänderung
1130	Argument kann weder eine Liste noch eine Matrix sein
1140	<p>Argumentfehler</p> <p>Das erste Argument muss ein Polynomausdruck im zweiten Argument sein. Wenn das zweite Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.</p>
1150	<p>Argumentfehler</p> <p>Die ersten zwei Argumente müssen Polynomausdrücke im dritten Argument sein. Wenn das dritte Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.</p>
1160	<p>Bibliotheks-Pfadname ungültig</p> <p>Ein Pfadname muss in der Form <code>xxx\yyy</code> angegeben werden, wobei:</p> <ul style="list-style-type: none"> • Der <code>xxx</code> Teil kann 1 bis 16 Zeichen haben. • Der <code>yyy</code> Teil kann 1 bis 15 Zeichen haben. <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1170	<p>Verwendung des Bibliotheks-Pfadnamens ungültig</p> <ul style="list-style-type: none"> • Ein Wert kann einem Pfadnamen nicht mit Definiere (Define), <code>:=</code> oder <code>sto</code> → zugewiesen werden. • Ein Pfadname kann nicht als lokale Variable festgelegt oder als Parameter in einer Funktions- oder Programmdefinition verwendet werden.
1180	<p>Bibliotheks-Variablenname ungültig.</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 15 Zeichen <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1190	Bibliotheks-Dokument nicht gefunden:

Fehlercode	Beschreibung
	<ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliothek im Ordner MyLib befindet. • Aktualisieren Sie die Bibliotheken. <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1200	<p>Bibliothaksvariable nicht gefunden:</p> <ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliotheksvariable im ersten Problem in der Bibliothek befindet. • Überprüfen Sie, dass die Bibliothaksvariable als LibPub oder LibPriv definiert wurde. • Aktualisieren Sie die Bibliotheken. <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1210	<p>Unzulässiger Name für Bibliotheks Kurzform.</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 <code>tangentLine</code> und <code>normalLine</code> 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 <code>nfMin</code> oder <code>nfMax</code> 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>

Fehlercode	Beschreibung
	Ordnung der Ableitung muss gleich 1 oder 2 sein.
1280	Argumentfehler Verwenden Sie ein Polynom in entwickelter Form in einer Variablen.
1290	Argumentfehler Verwenden Sie ein Polynom in einer Variablen.
1300	Argumentfehler Die Koeffizienten des Polynoms müssen numerische Werte ergeben.
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 generierten Warncodes speichern. In dieser Tabelle sind alle numerischen Warncodes und die zugehörigen Meldungen aufgelistet. Ein Beispiel zum Speichern von Warncodes finden Sie in **warnCodes()**, Seite 188.

Warncode	Nachricht
10000	Operation könnte falsche Lösungen erzeugen. Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
10001	Differenzieren einer Gleichung kann eine falsche Gleichung erzeugen.
10002	Zweifelhafte Lösung Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
10003	Zweifelhafte Genauigkeit Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
10004	Operation könnte Lösungen unterdrücken. Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
10005	cLöse (cSolve) liefert u.U. mehrere Nullstellen.
10006	Löse (Solve) liefert u.U. mehrere Nullstellen. Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
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) Falls zutreffend, verifizieren Sie die Ergebnisse mit grafischen Methoden.
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	∞^0 oder undef ⁰ durch 1 ersetzt
10014	undef ⁰ durch 1 ersetzt
10015	1^∞ oder 1^{undef} durch 1 ersetzt

Warncode	Nachricht
10016	1^undef durch 1 ersetzt
10017	Überlauf ersetzt durch ∞ oder $-\infty$
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.
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	Beschränkung kann ignoriert werden. Spezifizieren Sie eine Beschränkung in der Form "\ " 'Variable MathTestSymbol Constant' oder eine Kombination dieser Formen, zum Beispiel „x<3 and 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 für Informationen zur Dauer und den Bedingungen der Garantie bzw. zum Produktservice Ihr Land aus.

Eingeschränkte Garantie. Diese Garantie hat keine Auswirkungen auf Ihre gesetzlichen Rechte.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Inhalt

-		'	
-, subtrahieren	198	' , Minuten-Schreibweise	218
!		+	
!, Fakultät	209	+, addieren	197
"		<	
" , Sekunden-Schreibweise	218	<, kleiner als	206
#		=	
#, Umleitung	215	=, gleich	204
#, Umleitungsoperator	246	≠, ungleich[*]	205
%		>	
%, Prozent	204	>, größer als	207
*		∏	
*, multiplizieren	199	∏, Produkt	212
.		∑	
.-, Punkt-Subtraktion	202	∑(), Summe	212
*, Punkt-Multiplikation	203	∑Int()	213
./, Punkt-Division	203	∑Prn()	214
.^, Punkt-Potenz	203	√	
., Punkt-Addition	202	√(), Quadratwurzel	211
/		∠	
/, dividieren	200	∠ , winkel	218
:		∫	
:=, zuweisen	222	∫, Integral	211
^		≤	
^-1, Kehrwert	219	≤, kleiner oder gleich	206
^, Potenz	201	≥	
		≥, größer oder gleich	207
, womit-Operator	220		

►, in Neugrad umwandeln	75	Ob, binäre Anzeige	222
►approxFraction()	13	Oh, hexadezimale Anzeige	222
►Base10, Anzeige als ganze Dezimalzahl[Base10]	18	1	
►Base16, Hexadezimaldarstellung [Base16]	19	10^(), Potenz von zehn	219
►Base2, Binärdarstellung[Base2]	17	A	
►Cylind, Anzeige als Zylindervektor [Cylind (Zylindervektor)]	38	Abbruch, Exit	53
►DD, Anzeige als Dezimalwinkel[DD (Dezimalwinkel)]	39	Ableitungen	
►Decimal, Anzeige als Dezimalzahl [Dezimal]	39	erste Ableitung, d()	210
►DMS, Anzeige als Grad/Minute/Sekunde[DMS (GMS)]	47	numerische Ableitung, nDeriv()	113-114
►Polar, Anzeige als Polarvektor [Polar]	125	numerische Ableitung, nDerivative()	112
►Rad, in Bogenmaß umwandeln	134	abrufen/zurückgeben	
►Rect, Anzeige als kartesischer Vektor	138	Variableninformationen, getVarInfo()	71
►Sphere, Anzeige als sphärischer Vektor[Sphere (Kugelkoordinaten)]	165	Abrufen/zurückgeben	
		Variableninformationen, getVarInfo()	74
⇒		abs(), Absolutwert	7
⇒, logische Implikation	208, 243	Absolutwert	
		Vorlage für	3-4
→		addieren, +	197
→, speichern	221	als kartesischen Vektor anzeigen, ►Rect	138
		Amortisationstabelle, amortTbl() ..	7, 16
↔		amortTbl(), Amortisationstabelle ..	7, 16
↔, logische doppelte Implikation[*]	209	and, Boolean operator	8
		and, Boolesches und	8
©		angle(), Winkel	9
©, Kommentar	222	ANOVA, einfache Varianzanalyse ...	10
		ANOVA2way, zweifache Varianzanalyse	10
°		Ans, letzte Antwort	13
°, Grad-Schreibweise	217	Antwort (letzte), Ans	13
°, Grad/Minute/Sekunde	218	Anz, Daten anzeigen	152
		Anzeige als	
		binär, ►Base2	17
		Dezimalwinkel, ►DD	39
		ganze Dezimalzahl, ►Base10	18
		Grad/Minute/Sekunde, ►DMS ..	47
		hexadezimal, ►Base16	19
		Polarvektor, ►Polar	125
		sphärischer Vektor, ►Sphere ...	165
		Zylindervektor, ►Cylind	38

Cycle, Zyklus	37	Eigenwert, eigVl()	50
		eigVc(), Eigenvektor	49
D		eigVl(), Eigenwert	50
d(), erste Ableitung	210	Einheitsmatrix, identity()	76
Daten anzeigen, Anz	152	Einheitsvektor, unitV()	186
Daten anzeigen, Disp	45	Einstellungen, hole aktuellen	72
dbd(), Tage zwischen Daten	38	Elemente in einer Liste bedingt zählen, countlf()	33
Define, definiere	40	Elemente in einer Liste zählen, zähle ()	32
Definiere	40	else if, Elseif	50
Definiere LibPriv (Define LibPriv) ...	41	else, Else	76
Definiere LibPub (Define LibPub) ...	41	Elseif, else if	50
Definiere, Define	40	end	
definieren		for, EndFor	59
öffentliche Funktion /		if, EndIf	76
öffentliches Programm	41	Schleife, EndLoop	100
private Funktion oder		while, EndWhile	190
Programm	41	end if, EndIf	76
deltaList()	42	end while, EndWhile	190
DelVar, Variable löschen	42	Ende	
delVoid(), ungültige Elemente		Funktion, EndFunc	64
entfernen	43	Ende der Schleife, EndLoop	100
det(), Matrixdeterminante	43	EndWhile, end while	190
Dezimal		Entfernen	
Anzeige als ganze Zahl, ►Base10	18	ungültige Elemente aus Liste ...	43
Winkelanzeige, ►DD	39	EOS (Equation Operating System) ..	245
diag(), Matrixdiagonale	44	Equation Operating System (EOS) ..	245
Diagonalform, ref()	139	Ergebnisse mit zwei Variablen,	
dim(), Dimension	44	TwoVar	184
Dimension, dim()	44	Ergebnisse, Statistik	166
DispAt	45	Ergebniswerte, Statistik	167
dividieren, /	200	Ersetzung durch „ “ Operator	220
dotP(), Skalarprodukt	48	erste Ableitung	
drehen, rotate()	146-147	Vorlage für	5
durchschnittliche Änderungsrate,		erweitern/verketten, augment() ...	15
avgRC()	15	euler(), Euler function	51
E		Exit, Abbruch	53
e Exponent		exp(), e hoch x	54
Vorlage für	2	Exponent, E	215
e hoch x, e^()	48, 54	Exponenten	
E, Exponent	215	Vorlage für	1
e^(), e hoch x	48	Exponentielle Regression, ExpReg ..	55
echter Bruch, propFrac	130	expr(), String in Ausdruck	55
eff(), Nominal- in Effektivsatz		ExpReg, exponentielle Regression ..	55
konvertieren	49	F	
Effektivsatz, eff()	49	factor(), Faktorisieren	56
Eigenvektor, eigVc()	49		

I	
identity(), Einheitsmatrix	76
if, if	76
If, if	76
ifFn()	77
imag(), Imaginärteil	78
Imaginärteil, imag()	78
in String, inString()	79
inString(), in String	79
int(), ganze Zahl	79
intDiv(), Ganzzahl teilen	79
Integral, \int	211
Interpolieren(), interpolieren	80
invF()	81
invNorm(), inverse kumulative Normalverteilung)	82
invt()	82
Invx ² ()	80
iPart(), Ganzzahliger Teil	82
irr(), interner Zinsfluss interner Zinsfluss, irr()	83
isPrime(), Primzahltest	83
isVoid(), Test auf Ungültigkeit	84
K	
kartesische x-Koordinate, P►Rx() ...	123
kartesische y-Koordinate, P►Ry() ...	124
Kehrwert, \wedge^{-1}	219
Ketten drehen, rotate()	146-147
kleiner als, <	206
Kleiner oder gleich, \leq	206
kleinstes gemeinsames Vielfaches, lcm	85
Kombinationen, nCr()	111
Kommentar, ©	222
komplex Konjugierte, conj()	26
konvertieren ►Rad	134
Korrelationsmatrix, corrMat()	27
Kosinus, cos()	28
Kotangens, cot()	31
Kreuzprodukt, crossP()	34
kubische Regression, CubicReg	36
kumulierte Summe, cumulativeSum()	37

kumulierteSumme(), kumulierte Summe	37
L	
Lbl, Marke	84
lcm, kleinstes gemeinsames Vielfaches	85
leere (ungültige) Elemente	241
left(), links	85
LibPriv	41
LibPub	41
libShortcut(), erstelle Tastaturbefehle für Bibliotheksobjekte	86
lineare Regression, LinRegAx	87
lineare Regression, LinRegBx	86
Lineare Regression, LinRegBx	89
links, left()	85
LinRegBx, lineare Regression	86
LinRegMx, lineare Regression	87
LinRegtIntervals, lineare Regression	89
LinRegtTest	90
linSolve()	92
list►mat(), Liste in Matrix	93
Liste in Matrix, list►mat()	93
Liste, Elemente bedingt zählen	33
Liste, Elemente zählen in	32
Listen Differenzen in einer Liste, Δ list() erweitern/verketten, augment() in absteigender Reihenfolge sortieren, SortD	92 15 164
in aufsteigender Reihenfolge sortieren, SortA	164
Kreuzprodukt, crossP()	34
kumulierte Summe, cumulativeSum()	37
leere Elemente in	241
Liste in Matrix, list►mat()	93
Matrix in Liste, mat►list()	101
Maximum, max()	101
Minimum, min()	105
neu, newList()	113
Produkt, product()	129
Skalarprodukt, dotP()	48
Summe, sum()	170
Summierung, sum()	171
Teil-String, mid()	104

In (), natürlicher Logarithmus	93	Eigenvektor, eigVc()	49
LnReg, logarithmische Regression ..	94	Eigenwert, eigVl()	50
Local, lokale Variable	95	Einheitsmatrix, identity()	76
Lock, Variable oder Variablengruppe sperrern	96	erweitern/verketteten, augment()	15
LöFehler, Fehler löschen	24	füllen, Fill	57
Logarithmen	93	kumulierte Summe, cumulativeSum()	37
Logarithmische Regression, LnReg ..	94	Liste in Matrix, list►mat()	93
Logarithmus Vorlage für	2	Matrix in Liste, mat►list()	101
logische doppelte Implikation, ⇔ ..	209	Matrixzeilenmultiplikation und - addition, mRowAdd() ..	107
logische Implikation, ⇒ ..	208, 243	Maximum, max()	101
Logistic, logistische Regression	97	Minimum, min()	105
LogisticD, logistische Regression	98	neu, newMat()	113
Logistische Regression, Logistic	97	Produkt, product()	129
Logistische Regression, LogisticD	98	Punkt-Addition, .+	202
lokal, Local	95	Punkt-Division, ./	203
lokale Variable, Local	95	Punkt-Multiplikation, .*	203
Loop, Schleife	100	Punkt-Potenz, .^	203
löschen Variable, DelVar	42	Punkt-Subtraktion, -.	202
Löschen	228	QR-Faktorisierung, QR	130
Fehler, LöFehler	24	Spaltendimension, colDim() ..	25
ungültige Elemente aus Liste	43	Spaltennorm, colNorm()	26
LU, untere/obere Matrixzerlegung ..	100	Summe, sum()	170
M		Summierung, sum()	171
Marke, Lbl	84	Transponierte, T	172
mat►list(), Matrix in Liste	101	untere/obere Matrixzerlegung, LU	100
Matrix Diagonalform, ref()	139	Untermatrix, subMat()	170, 172
reduzierte Diagonalform, rref() ..	150	Zeilenoperation, mRow()	107
Matrix (1 × 2) Vorlage für	4	max(), Maximum	101
Matrix (2 × 1) Vorlage für	4	Maximum, max()	101
Matrix (2 × 2) Vorlage für	4	mean(), Mittelwert	102
Matrix (m × n) Vorlage für	4	median(), Median	102
Matrix erstellen, constructMat()() ..	26	Median, median()	102
Matrix in Liste, mat►list()	101	MedMed, Mittellinienregression ..	103
Matrixzeilenaddition, rowAdd()	149	mid(), Teil-String	104
Matrixzeilentausch, rowSwap()	149	min(), Minimum	105
Matrizen Determinante, det()	43	Minimum, min()	105
Diagonale, diag()	44	Minuten-Schreibweise,	218
Dimension, dim()	44	mirr(), modifizierter interner Zinsfluss	106
		mit, 	220
		Mittellinienregression, MedMed ..	103
		Mittelwert, mean()	102
		mod(), Modulo	106
		Modi festlegen, setMode()	155
		Modifizierter interner Zinsfluss, mirr	106

()	106	nlichkeit)	117
Modulo, mod()	106	normPdf()	117
Moduseinstellungen, getMode()	72	(Wahrscheinlichkeitsdichte)	117
mRow(), Matrixzeilenoperation	107	nPr(), Permutationen	118
mRowAdd(),		npv(), Nettobarwert	119
Matrixzeilenmultiplikation		nSolve(), numerische Lösung	119
und -addition	107	numerisch	
Multipler linearer Regressions-t-Test	109	Ableitung, nDeriv()	113-114
multiplizieren, *	199	Ableitung, nDerivative()	112
MultReg (Mehrfachregression)	107	Integral, nInt()	114
MultRegIntervals()		Lösung, nSolve()	119
(Mehrfachregressionsinterv			
all)	108		
MultRegTests()	109		
N		O	
n-te Wurzel		Obergrenze, ceiling()	21, 34
Vorlage für	2	Objekte	
nand, Boolescher Operator	110	erstelle Tastaturbefehle für	
natürlicher Logarithmus, ln()	93	Bibliothek	86
nCr(), Kombinationen	111	oder (Boolesch), oder	121
nDerivative(), numerische Ableitung	112	oder, Boolescher Operator	121
Negation, Eingabe von negativen		OneVar, Statistik mit einer Variable	120
Zahlen	246	Operatoren	
Nettobarwert, npv()	119	Auswertungsreihenfolge	245
neu		ord(), numerischer Zeichencode	123
Liste, newList()	113		
Matrix, newMat()	113		
Neugrad-Schreibweise, g	216	P	
newList(), neue Liste	113	P►Rx(), kartesische x-Koordinate	123
newMat(), neue Matrix	113	P►Ry(), kartesische y-Koordinate	124
nfMax(), numerisches		Pdf()	61
Funktionsmaximum	113	Permutationen, nPr()	118
nfMin(), numerisches		piecewise() (Stückweise)	125
Funktionsminimum	114	poissCdf()	125
nicht, Boolescher Operator	117	poissPdf()	125
nInt(), numerisches Integral	114	polar	
nom(), Effektivzins in Nominalzins		Vektoranzeige, ►Polar	125
konvertieren	115	Polarkoordinate, R►Pr()	134
Nominalzinssatz, nom()	115	Polarkoordinate, R►Pθ()	134
nor, Boolescher Operator	115	polyEval(), Polynom auswerten	126
norm(), Frobeniusnorm	116	Polynom auswerten, polyEval()	126
Normalverteilung invertieren		Polynome	
(invNorm()	82	auswerten, polyEval()	126
Normalverteilungswahrscheinlichkeit,		PolyRoots()	127
normCdf()	116	Potenz von zehn, 10^()	219
normCdf()		Potenz, ^	201
(Normalverteilungswahrscheinlichkeit)	116	Potenzregression,	
		PowerReg	127, 141, 143, 175
		PowerReg, Potenzregression	127
		Prgm, Definiere Programm	128
		Primzahltest, isPrime()	83

prodSeq()	129	randNorm(), Zufallsnorm	136
product(), Produkt	129	randPoly(), Zufallspolynom	137
Produkt $\prod()$		randSamp()	137
Vorlage für	5	RandSeed, Zufallszahl	137
Produkt, $\prod()$	212	real(), reel	137
Produkt, product()	129	rechts, right()	80, 144-145
Programme		reduzierte Diagonalform, rref()	150
öffentliche Bibliothek definieren	41	reell, real()	137
Private Bibliothek definieren	41	ref(), Diagonalform	139
Programme und Programmieren		RefreshProbeVars	140
E/A-Bildschirm anzeigen, Anz	152	Regression vierter Ordnung,	
E/A-Bildschirm anzeigen, Zeige	45	QuartReg	133
Fehler löschen, LöFehler	24	Regressionen	
programmieren		exponentielle, ExpReg	55
Daten anzeigen, Disp	45	kubische, CubicReg	36
Definiere Programm, Prgm	128	lineare Regression, LinRegAx	87
Fehler übergeben, ÜgebFeh	124	lineare Regression, LinRegBx	86
Programmierung		Lineare Regression, LinRegBx	89
Daten anzeigen, Anz	152	logarithmische, LnReg	94
propFrac, echter Bruch	130	Logistic (Logistisch)	97
Prozent, %	204	logistische, Logistic	98
Punkt		Mittellinie, MedMed	103
Addition, .+	202	MultReg (Mehrfachregression)	107
Division, ./	203	Potenzregression,	
Multiplikation, .*	203	PowerReg	127, 141, 143, 175
Potenz, .^	203	quadratische, QuadReg	131
Subtraktion, .-	202	sinusförmige, SinReg	162
		vierter Ordnung, QuartReg	133
		remain(), Rest	141
		Request	141
		RequestStr	143
		Rest, remain()	141
		Return, Rückgabe	144
		right(), rechts	144
		right, right()	51, 188
		rk23(), Runge-Kutta-Funktion	145
		rotate(), drehen	146-147
		round(), runden	148
		rowAdd(), Matrixzeilenaddition	149
		rowDim(), Zeilendimension der	
		Matrix	149
		rowNorm(), Zeilennorm der Matrix	149
		rowSwap(), Matrixzeilentausch	149
		rref(), reduzierte Diagonalform	150
		Rückgabe, Return	144
		runden, round()	148
Q			
QR-Faktorisierung, QR	130		
QR,QR-Faktorisierung	130		
Quadratische Regression, QuadReg	131		
Quadratwurzel			
Vorlage für	1		
Quadratwurzel, $\sqrt{}$	211		
Quadratwurzel, $\#()$	165		
QuadReg, quadratische Regression	131		
QuartReg, Regression vierter			
Ordnung	133		
R			
r, Bogenmaß	216		
R \rightarrow Pr(), Polarkoordinate	134		
R \rightarrow P θ (), Polarkoordinate	134		
rand(), Zufallszahl	135		
randBin, Zufallszahl	135		
randInt(), ganzzahlige Zufallszahl	135		
randMat(), Zufallsmatrix	136		

S

Schleife, Loop	100	Mittelwert, mean()	102
Schreibweise Grad/Minute/Sekunde	218	Permutationen, nPr()	118
sec ⁻¹ (), Arkussekans	151	Standardabweichung, stdDev()	167-168, 186
sec(), Sekans	150	Statistik mit einer Variable, OneVar	120
sch ⁻¹ (), Arkussekans hyperbolicus ..	151	Varianz, variance()	187
sech(), Sekans hyperbolicus	151	Zufallsnorm, randNorm()	136
Sekunden-Schreibweise, "	218	Zufallszahl, RandSeed	137
seq(), Folge	153	Statistik mit einer Variable, OneVar	120
seqGen()	153	stdDevPop(), Populations- Standardabweichung	167
seqn()	154	stdDevSamp(), Stichproben- Standardabweichung	168
sequence, seq()	153-154	String	
setMode(), Modus festlegen	155	Dimension, dim()	44
shift(), verschieben	156	Länge	44
sign(), Zeichen	158	string(), Ausdruck in String	169
simult(), Gleichungssystem	159	Stringlänge	44
sin ⁻¹ (), Arkussinus	160	strings	
sin(), Sinus	160	right, right()	51, 188
sinh ⁻¹ (), Arkussinus hyperbolicus ..	162	Strings	
sinh(), Sinus hyperbolicus	161	Ausdruck in String, string()	169
SinReg, sinusförmige Regression	162	Format, format()	60
Sinus, sin()	160	Formatieren	60
Sinusförmige Regression, SinReg	162	in, inString	79
Skalar		links, left()	85
Produkt, dotP()	48	rechts, right()	80, 144-145
SortA, in aufsteigender Reihenfolge sortieren	164	String in Ausdruck, expr()	55
SortD, in absteigender Reihenfolge sortieren	164	Teil-String, mid()	104
sortieren		Umleitung, #	215
in absteigender Reihenfolge sortieren, SortD	164	verschieben, shift()	156
in aufsteigender Reihenfolge, SortA	164	Zeichencode, ord()	123
speichern		Zeichenstring, char()	22
Symbol, →	221	Stückweise definierte Funktion (2 Teile)	
Sprache		Vorlage für	2
Sprachinformation abrufen	71	Stückweise definierte Funktion (n Teile)	
sqrt(), Quadratwurzel	165	Vorlage für	3
Standardabweichung, stdDev()	167-168, 186	Student-t-	
stat.results	166	Wahrscheinlichkeitsdichte, tPdf()	178
stat.values	167	subMat(), Untermatrix	170, 172
Statistik		subtrahieren, -	198
Ergebnisse mit zwei Variablen, TwoVar	184	sum(), Summe	170
Fakultät, !	209	sumIf()	171
Kombinationen, nCr()	111	Summe \sum ()	
Median, median()	102	Vorlage für	5

poissPdf()	125	womit-Operator „ “	220
tCdf()	175	womit-Operator, Auswerungsreihenfolge	245
tPdf()	178		
χ^2 2way()	22		
χ^2 Cdf()	23	X	
χ^2 GOF()	23	x^2 , Quadrat	202
χ^2 Pdf()	24	XNOR	209
void, test for	84	xor, Boolesches exklusives oder	190
Vorlagen			
Absolutwert	3-4		
Bestimmtes Integral	6	Z	
Bruch	1	Zähle Tage zwischen Daten, dbd()	38
e Exponent	2	zähle(), Elemente in einer Liste	
erste Ableitung	5	zählen	32
Exponent	1	Zeichen	
Gleichungssystem (2 Gleichungen)	3	String, char()	22
Gleichungssystem (n Gleichungen)	3	Zeichencode, ord()	123
Logarithmus	2	Zeichen, sign()	158
Matrix (1 × 2)	4	Zeichenfolgen	
Matrix (2 × 1)	4	zum Erstellen von	
Matrix (2 × 2)	4	Variablenamen	
Matrix (m × n)	4	verwenden	246
n-te Wurzel	2	Zeichenstring, char()	22
Produkt $\prod()$	5	Zeichnen	229-231
Quadratwurzel	1	Zeige, Daten anzeigen	45
Stückweise definierte Funktion (2 Teile)	2	Zeilendimension der Matrix, rowDim ()	149
Stückweise definierte Funktion (n Teile)	3	Zeilenorm der Matrix, rowNorm()	149
Summe $\sum()$	5	Zeitwert des Geldes, Anzahl Zahlungen	182
zweite Ableitung	6	Zeitwert des Geldes, Barwert	183
		Zeitwert des Geldes, Endwert	182
W		Zeitwert des Geldes, Zahlungsbetrag	183
Wahrscheinlichkeit einer Student-t- Verteilung, tCdf()	175	Zeitwert des Geldes, Zinsen	182
Wahrscheinlichkeitsdichte, normPdf()	117	zInterval, z-Konfidenzintervall	191
Warncodes und -meldungen	260	zInterval_1Prop, z-Konfidenzintervall für eine Proportion	192
warnCodes(), Warning codes	188	zInterval_2Prop, z-Konfidenzintervall für zwei Proportionen	192
Warte-Befehl	188	zInterval_2Samp, z- Konfidenzintervall für zwei Stichproben	193
wenn, when()	189	zTest	194
when(), wenn	189	zTest_1Prop, z-Test für eine Proportion	195
while, While	190	zTest_2Prop, z-Test für zwei Proportionen	196
While, while	190	zTest_2Samp, z-Test für zwei	196
winkel, \angle	218		
Winkel, angle()	9		

Stichproben	
Zufallsmatrix, randMat()	136
Zufallsnorm, randNorm()	136
Zufallspolynom, randPoly()	137
Zufallsstichprobe	137
Zufallszahl, RandSeed	137
zuweisen, :=	222
Zwei-Stichproben F-Test	63
zweite Ableitung	
Vorlage für	6
Zyklus, Cycle	37

Δ

Δlist(), Listendifferenz	92
---------------------------------	----

X

χ^2 2way	22
χ^2 Cdf()	23
χ^2 GOF	23
χ^2 Pdf()	24