TI-Nspire™ CX
Referenzhandbuch

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

Wichtige Informationen

Sofern nicht ausdrücklich in der einem Programm beiliegenden Lizenz angegeben, übernimmt Texas Instruments für die Programme oder das Handbuchmaterial keinerlei Garantie, weder direkt noch indirekt. Dies umfasst auch jegliche indirekte Gewährleistung hinsichtlich der Marktgängigkeit oder der Eignung für einen bestimmten Zweck, ist jedoch nicht hierauf beschränkt und dieses Produkt wird lediglich „so wie es ist” zur Verfügung gestellt. In keinem Fall kann Texas Instruments für Schäden haftbar gemacht werden, die sich entweder in Verbindung mit dem Kauf bzw. Gebrauch dieses Produkts ergeben oder davon verursacht werden. Dies gilt für spezielle, begleitende und versehentliche Schäden sowie für Folgeschäden. Texas Instruments haftet maximal und ausschließlich mit dem in der Lizenz für das Programm genannten Betrag, unabhängig vom jeweiligen Fall. Des Weiteren haftet Texas Instruments nicht für Forderungen, die sich aus dem Gebrauch dieses Produkts durch eine andere Partei ergeben, welcher Art diese Forderungen auch immer sein mögen.

© 2026 Texas Instruments Incorporated

Die aktuellen Produkte können geringfügig von den Abbildungen abweichen.

Inhalt

Vorlagen für Ausdrücke **1**

Alphabetische Auflistung **7**

A 7
B 16
C 21
D 38
E 48
F 56
G 64
I 75
L 84
M 101
N 110
O 119
P 122
Q 130
R 133
S 149
T 171
U 185
V 185
W 187
X 189
Z 190

Sonderzeichen **196**

TI-Nspire™ CX II – Zeichenbefehle **223**

Grafikprogrammierung 223
Grafikbildschirm 223
Standardansicht und Einstellungen 224
Fehlermeldungen des Grafikbildschirms 225
Im Grafikmodus ungültige Befehle 225
C 227
D 228
F 232
G 234
P 235
F: 237
U 239

Leere (ungültige) Elemente **240**

Tastenkürzel zum Eingeben mathematischer Ausdrücke **242**

Auswertungsreihenfolge in EOS™ (Equation Operating System) **244**

TI-Nspire CX II – TI-Basic Programmierfunktionen **246**

Automatisches Einrücken im Programmierungseditor 246
Verbesserte Fehlermeldungen für TI-Basic 246

Konstanten und Werte **249**

Fehlercodes und -meldungen **250**

Warncodes und -meldungen **259**

Allgemeine Informationen **261**

Inhalt **262**

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**

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

Hinweis: Siehe auch **/ (Dividieren)**, Seite 199.

Beispiel:

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

Vorlage Exponent

[^] **Taste**

$\square^{\square}$

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 200.

Beispiel:

2^3 $\quad 8$

Vorlage Quadratwurzel

[ctrl] [x²] **Tasten**

$\sqrt{\square}$

Hinweis: Siehe auch **√() (Quadratwurzel)**, Seite 210.

Beispiel:

$\sqrt{4}$ $\quad 2$

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

Vorlage n-te Wurzel

ctrl ^ Tasten

$\sqrt[\square]{\square}$

Hinweis: Siehe auch **root()**, Seite 146.

Beispiel:

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

Vorlage e Exponent

e^x Tasten

$e^{\square}$

Potenz zur natürlichen Basis *e*

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

Example:

e^1	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:

$\log_4(2.)$	0.5

Vorlage Stückweise (2 Teile)

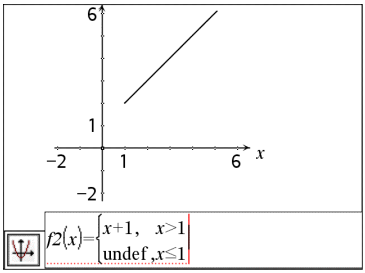
Katalog >

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

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 124.

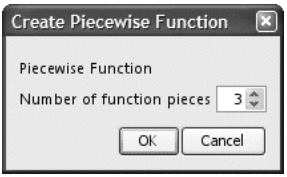
Beispiel:



Vorlage Stückweise (n Teile)

Katalog >

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



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

Beispiel:

Siehe Beispiel für die Vorlage Stückweise (2 Teile).

Vorlage System von 2 Gleichungen

Katalog >

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

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

Hinweis: Siehe auch **system()**, Seite 171.

Beispiel:

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

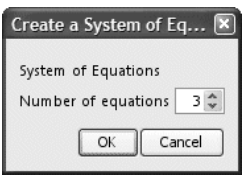
$$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$$

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

Vorlage System von n Gleichungen

Katalog >

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



Hinweis: Siehe auch **system()**, Seite 171.

Beispiel:

Siehe Beispiel für die Vorlage Gleichungssystem (2 Gleichungen).

Vorlage Absolutwert

Katalog >

$|\square|$ **Hinweis:** Siehe auch **abs()**, Seite 7.

Beispiel:

Vorlage Absolutwert

Katalog >

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

Vorlage dd°mm'ss.ss''

Katalog >

□°□'□"

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

Beispiel:

$$30°15'10" \qquad 0.528011$$

Vorlage Matrix (2 x 2)

Katalog >

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

Erzeugt eine 2 x 2 Matrix.

Beispiel:

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

Vorlage Matrix (1 x 2)

Katalog >

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

Beispiel:

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

Vorlage Matrix (2 x 1)

Katalog >

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

Beispiel:

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

Vorlage Matrix (m x n)

Katalog >

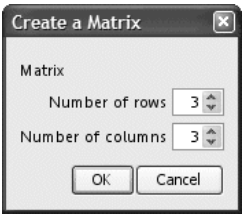
Die Vorlage wird angezeigt, nachdem Sie aufgefordert wurden, die Anzahl der Zeilen und Spalten anzugeben.

Beispiel:

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

Vorlage Matrix (m x n)

Katalog >



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

Vorlage Summe (Σ)

Katalog >

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

Hinweis: Siehe auch **Σ() (sumSeq)**, Seite 211.

Beispiel:

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

Vorlage Produkt (Π)

Katalog >

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

Hinweis: Siehe auch **Π() (prodSeq)**, Seite 211.

Beispiel:

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

Vorlage Erste Ableitung

Katalog >

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

Beispiel:

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

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 209.

Vorlage Zweite Ableitung

Katalog >

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

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

Hinweis: Siehe auch **d() (Ableitung)**, Seite 209.

Beispiel:

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

Vorlage Bestimmtes Integral

Katalog >

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

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

Hinweis: Siehe auch **nInt()**, Seite 114.

Beispiel:

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

Alphabetische Auflistung

Elemente, deren Namen nicht alphabetisch sind (wie +, !, und >) finden Sie am Ende dieses Abschnitts (Seite 196). 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*

abs(*Liste1***)⇒***Liste*

abs(*Matrix1***)⇒***Matrix*

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.

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

$|2-3\cdot i|$ 3.60555

amortTbl() — Katalog >

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

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

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

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* und *PmtAt* werden in der TVM-Argumentetabelle (Seite 182) beschrieben.

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

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

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

- 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 **ΣInt()** und **ΣPrn()**, Seite 212, und **bal()**, Seite 16, verwenden.

and (und) Katalog >

Boolescher Ausdr1 **and** *Boolescher Ausdr2⇒Boolescher Ausdruck*

Boolesche Liste1 **and** *Boolesche Liste2 ⇒Boolesche Liste*

Boolesche Matrix1 **and** *Boolesche Matrix2⇒Boolesche Matrix*

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

*Ganzzahl1***and***Ganzzahl2⇒Ganzzahl*

Im Hex-Modus:

0h7AC36 and 0h3D5F	0h2C16

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 and 0b100	0b100

and (und) Katalog >

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) Katalog >

angle(*Wert1***)⇒***Wert*

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

Im Grad-Modus:

angle(0+2·*i*)	90

Im Neugrad-Modus:

angle(0+3·*i*)	100

Im Bogenmaß-Modus:

angle(1+***i***)	0.785398
angle({1+2·***i***,3+0·***i***,0−4·***i***})	{1.10715,0.,⁻1.5708}

angle(*Liste1***)⇒***Liste*

angle(*Matrix1***)⇒***Matrix*

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 165)

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 165)

LevZei=0 für Block

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

Ausgaben: Block-Design

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

Ausgaben des SPALTENFAKTORS

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

Ausgaben des ZEILENFAKTORS

Ausgabevariable	Beschreibung
stat.Frow	F Statistik des Zeilenfaktors
stat.PValRow	Wahrscheinlichkeitswert des Zeilenfaktors
stat.dfRow	Freiheitsgrade des Zeilenfaktors
stat.SSRow	Summe der Fehlerquadrate des Zeilenfaktors
stat.MSRow	Mittlere Quadrate für Zeilenfaktor

INTERAKTIONS-Ausgaben

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

FEHLER-Ausgaben

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

Ans (Antwort)

ctrl (–) Taste

Ans⇒*Wert*

Gibt das Ergebnis des zuletzt ausgewerteten Ausdrucks zurück.

56	56
56+4	60
60+4	64

approx() (Approximieren)

Katalog >

approx(*Wert1***)**⇒*Zahl*

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

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

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

approx(*Liste1***)**⇒*Liste*

approx(*Matrix1***)**⇒*Matrix*

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

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

▸approxFraction()

Katalog >

Wert ▸approxFraction([*Tol*]**)**⇒*Wert*

Liste **▸approxFraction(**[*Tol*]**)**⇒*Liste*

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

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

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

$\frac{1}{2}+\frac{1}{3}+\tan(\pi)$	0.833333
0.83333333333333▸approxFraction(5.E-14)	$\frac{5}{6}$
{π,1.5}▸approxFraction(5.E-14)	$\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

approxRational() Katalog >

approxRational(*Wert*[, *Tol*]**)⇒***Wert*

approxRational(*Liste*[, *Tol*]**)⇒***Liste*

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

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

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

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

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

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

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

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

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

arccsch() Siehe $\text{csch}^{-1}()$, Seite 36.

arcsec() Siehe $\sec^{-1}()$, Seite 150.

arcsech() Siehe $\text{sech}^{-1}()$, Seite 150.

arcsin() Siehe $\sin^{-1}$(), Seite 159.

arcsinh() Siehe $\sinh^{-1}$(), Seite 161.

arctan() Siehe $\tan^{-1}$(), Seite 172.

arctanh() Siehe $\tanh^{-1}$(), Seite 173.

augment() (Erweitern) Katalog >

augment(*Liste1, Liste2***)⇒***Liste*

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

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

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*.

```
[1 2; 3 4] → m1                [1 2; 3 4]
[5; 6] → m2                    [5; 6]
augment(m1,m2)                 [1 2 5; 3 4 6]
```

avgRC() (Durchschnittliche Änderungsrate) Katalog >

avgRC(*Ausdr1***, *Var* [=*Wert*] [, *Schritt*])⇒***Ausdruck***

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

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

avgRC(*Matrix1***, *Var* [=*Wert*] [,**

```
x:=2                           2
avgRC(x^2-x+2,x)               3.001
avgRC(x^2-x+2,x,.1)            3.1
avgRC(x^2-x+2,x,3)             6
```

avgRC() (Durchschnittliche Änderungsrate) Katalog >

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() Katalog >

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

bal(*NPmt*,*AmortTabelle***)**⇒*Wert*

Amortisationsfunktion, die den Saldo nach einer angegebenen Zahlung berechnet.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* und *PmtAt* werden in der TVM-Argumentetabelle (Seite 182) beschrieben.

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

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* und *PmtAt* werden in der TVM-Argumentetabelle (Seite 182) beschrieben.

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

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

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

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

bal(4,*tbl*) 1674.27

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 212.

▸Base2

Ganzzahl1 **▸Base2**⇒*Ganzzahl*

256▸Base2	0b100000000
0h1F▸Base2	0b11111

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

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

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

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

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

Negative Zahlen werden als Binärkomplement angezeigt. Beispiel:

▸Base2 Katalog >

−1 wird angezeigt als

0hFFFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 Einsen) im Binärmodus

-2^{63} 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:

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

0h8000000000000000 im Hex-Modus

0b100...000 (63 Nullen) im Binärmodus

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

0h0 im Hex-Modus

0b0 im Binärmodus

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

0h7FFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 1's) im Binärmodus

▸Base10 Katalog >

Ganzzahl1 ▸**Base10**⇒*Ganzzahl*

0b10011▸Base10	19
0h1F▸Base10	31

▸Base10 Katalog >

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.

▸Base16 Katalog >

Ganzzahl1 ▸**Base16**⇒*Ganzzahl*

256▸Base16	0h100
0b111100001111▸Base16	0hF0F

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.

▶Base16 Katalog >

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() Katalog >

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≤X ≤*obereGrenze*) ⇒*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 ≤ *obereGrenze*) setzen Sie *untereGrenze*=0

binomPdf() Katalog >

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***)⇒***Wert*

Gibt die erste ganze Zahl zurück, die ≥ dem Argument ist.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

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

```
ceiling(.456)                                   1.
```

ceiling(*Liste1***)⇒***Liste*

ceiling(*Matrix1***)⇒***Matrix*

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

```
ceiling({-3.1,1,2.5})                           {-3.,1,3.}
ceiling([0  -3.2·i; 1.3  4])                    [0  -3.·i; 2.  4]
```

centralDiff() Katalog >

centralDiff(*Ausdr1***,***Var* [=*Wert*] [,*Schritt*])⇒***Ausdruck***

centralDiff(*Ausdr1***,***Var* [,*Schritt*])|***Var*=*Wert*⇒***Ausdruck***

centralDiff(*Ausdr1***,***Var* [=*Wert*] [,*Liste*])⇒***Liste***

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

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

Gibt die numerische Ableitung unter Verwendung des zentralen Differenzenquotienten zurück.

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

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

```
centralDiff(cos(x),x)|x=π/2                     -1.
```

centralDiff() Katalog >

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

Hinweis: Siehe auch.

char() (Zeichenstring) Katalog >

char(*Ganzzahl***)⇒***Zeichen*

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.

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

χ22way Katalog >

χ22way *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 165.)

Informationen zu den Auswirkungen leerer Elemente in einer Matrix finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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

χ^2Cdf()

Katalog >

χ^2Cdf
(*untereGrenze***,***obereGrenze***,***Freigrad***)⇒**
Zahl, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

chi2Cdf
(*untereGrenze***,***obereGrenze***,***Freiheitsgrad***)**
⇒*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($X \leq$ *obereGrenze*) setzen Sie *untereGrenze*= 0.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

χ^2GOF

Katalog >

χ^2GOF *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 165)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgabevariable	Beschreibung
stat.χ^2	Chi-Quadrat-Testgröße: sum((beobachtet - erwartet)^2/erwartet

Ausgabevariable	Beschreibung
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.CompList	Liste der Chi-Quadrat-Summanden in der Testgröße

χ^2Pdf() Katalog >

χ^2Pdf(*XWert***,***FreiGrad***)⇒***Zahl***,** wenn *Xwert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

chi2Pdf(*XWert***,***FreiGrad***)⇒***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 240).

ClearAZ (LöschAZ) Katalog >

ClearAZ

Löscht alle Variablen mit einem Zeichen im aktuellen Problembereich.

Wenn eine oder mehrere Variablen gesperrt sind, wird bei diesem Befehl eine Fehlermeldung angezeigt und es werden nur die nicht gesperrten Variablen gelöscht. Siehe **unLock**, Seite 185

$5 \rightarrow b$	5
b	5
ClearAZ	*Done*
b	"Error: Variable is not defined"

ClrErr (LöFehler) Katalog >

ClrErr

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

Ein Beispiel für **ClrErr** finden Sie als Beispiel 2 im Abschnitt zum Befehl **Versuche (Try)**, Seite 178.

ClrErr (LöFehler) Katalog >

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 123, und **Try**, Seite 178.

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

colAugment() (Spaltenerweiterung) Katalog >

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

Gibt eine neue Matrix zurück, die durch Anfügen von *Matrix2* an *Matrix1* erzeugt wurde. Die Matrizen müssen gleiche Spaltendimensionen haben, und *Matrix2* wird zeilenweise an *Matrix1* angefügt. Verändert weder *Matrix1* noch *Matrix2*.

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

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

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

colDim() (Spaltendimension) Katalog >

colDim(*Matrix***)⇒*Ausdruck***

Gibt die Anzahl der Spalten von *Matrix* zurück.

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

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

colNorm() (Spaltennorm)

Katalog >

colNorm(*Matrix***)⇒***Ausdruck*

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

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

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

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

conj() (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.

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

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

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*.

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

CopyVar Katalog >

CopyVar *Var1*, *Var2*

CopyVar *Var1*., *Var2*.

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

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

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*..

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

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.

corrMat() (Korrelationsmatrix) Katalog >

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

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

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 vorübergend aufzuheben.

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
$\cos(45°)$	0.707107

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 (λi) und Eigenvektoren (Vi) 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 Bogenmaß-Modus:

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

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

cos() (Kosinus) trig Taste

Dann ist A = X B X^{-1}und f(A) = X f(B) X^{-1}.
Beispiel: cos(A) = X cos(B) X^{-1}, wobei:

cos (B) =

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

Alle Berechnungen werden unter Verwendung von Fließkomma-Operationen ausgeführt.

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

$\cos^{-1}$(*Wert1***)⇒***Wert***

$\cos^{-1}$(*Liste1***)⇒***Liste***

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

$\cos^{-1}$(*Liste1***)** gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Kosinus 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 `arccos(...)` eintippen.

$\cos^{-1}$(*Quadratmatrix1***)⇒***Quadratmatrix***

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()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Grad-Modus:

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

$\cos^{-1}(\{0,0.2,0.5\})$
$\{1.5708,1.36944,1.0472\}$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

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

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

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

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

cosh() (Cosinus hyperbolicus) Katalog >

cosh(*Wert1***)⇒***Wert*

cosh(*Liste1***)⇒***Liste*

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

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

Im Grad-Modus:

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

cosh(*Quadratmatrix1***)⇒***Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

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

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

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

$\cosh^{-1}$(*Wert1***)⇒***Wert*

$\cosh^{-1}$(*Liste1***)⇒***Liste*

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

$\cosh^{-1}(\{1,2.1,3\})$ $\{0,1.37286,1.76275\}$

$\cosh^{-1}$(*Wert1***)** gibt den inversen Cosinus hyperbolicus des Arguments zurück.

$\cosh^{-1}$(*Liste1*) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Cosinus hyperbolicus zurück.

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

cosh⁻¹() (Arkuskosinus hyperbolicus) Katalog >

cosh⁻¹(*Quadratmatrix1***)⇒***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.

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

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

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

cot() (Kotangens) trig Taste

cot(*Wert1***) ⇒ *Wert***

cot(*Liste1***) ⇒ *Liste***

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

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

Im Grad-Modus:

$\cot(45)$ 1.

Im Neugrad-Modus:

$\cot(50)$ 1.

Im Bogenmaß-Modus:

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

cot⁻¹() (Arkuskotangens) trig Taste

cot⁻¹(*Wert1***)⇒*Wert***

cot⁻¹(*Liste1***)⇒*Liste***

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

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

Im Grad-Modus:

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

$\cot^{-1}()$ (Arkuskotangens) — trig Taste

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

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

coth() (Kotangens hyperbolicus) — Katalog >

coth(*Wert1***)⇒***Wert*

coth(*Liste1***)⇒***Liste*

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

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

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

coth⁻¹(*Wert1***)⇒***Wert*

coth⁻¹(*Liste1***)⇒***Liste*

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

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

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

count() (zähle) — Katalog >

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

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

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

$\text{count}(2,4,6)$	3
$\text{count}(\{2,4,6\})$	3
$\text{count}\left(2,\{4,6\},\begin{bmatrix}8 & 10\\ 12 & 14\end{bmatrix}\right)$	7

count() (zähle) Katalog >

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 240).

countIf() Katalog >

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 240).

Hinweis: Siehe auch **sumIf()**, Seite 170, 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***)⇒***Liste***

cPolyRoots(*KoeffListe***)⇒***Liste***

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

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

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

Hinweis: Siehe auch **polyRoots()**, Seite 126.

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

crossP() (Kreuzprodukt) Katalog >

crossP(*Liste1***, *Liste2***)⇒***Liste***

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

crossP(*Vektor1***, *Vektor2***)⇒***Vektor***

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.

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

csc() (Kosekans) trig Taste

csc(*Wert1***)⇒***Wert***

Im Grad-Modus:

csc() (Kosekans) trig Taste

csc(*Liste1***)⇒***Liste*

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

$\csc(45)$	1.41421

Im Neugrad-Modus:

$\csc(50)$	1.41421

Im Bogenmaß-Modus:

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

csc⁻¹() (Inverser Kosekans) trig Taste

csc⁻¹(*Wert1***)** ⇒ *Wert*

csc⁻¹(*Liste1***)** ⇒ *Liste*

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

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

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

Im Grad-Modus:

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

csch() (Kosekans hyperbolicus) Katalog >

csch(*Wert1***)** ⇒ *Wert*

csch(*Liste1***)** ⇒ *Liste*

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

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

csch⁻¹() (Inverser Kosekans hyperbolicus) Katalog >

csch⁻¹(*Wert1***)** ⇒ *Wert*

csch⁻¹(*Liste1***)** ⇒ *Liste*

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

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

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

CubicReg (Kubische Regression) Katalog >

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

Berechnet die kubische polynomiale Regressiony = a ·x3+b · x2+c ·x+dauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a ·x^3+b ·x^2+c ·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 *X List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten *Y List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

cumulativeSum() (kumulierteSumme)

Katalog >

cumulativeSum(*Liste1***)⇒***Liste***

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

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

cumulativeSum(*Matrix1***)⇒***Matrix***

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

Ein leeres (ungültiges) Element in *Liste1* oder *Matrix1* erzeugt ein ungültiges Element in der resultierenden Liste oder Matrix. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

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

Cycle (Zyklus)

Katalog >

Cycle (Zyklus)

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

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

Cycle (Zyklus) Katalog >

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

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

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

▸Cylind (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,∠θ, z] an.

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

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

dbd() Katalog >

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) Katalog >

Zahl ▸**DD**⇒*Wert*

Liste1 ▸**DD**⇒*Liste*

Matrix1 ▸**DD**⇒*Matrix*

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

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

Im Grad-Modus:

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

Im Neugrad-Modus:

1▸DD	$\frac{9}{10}$°

Im Bogenmaß-Modus:

(1.5)▸DD	85.9437°

▸Decimal (Dezimal) Katalog >

Wert1 ▸**Decimal**⇒*Wert*

Liste1 ▸**Decimal**⇒*Wert*

Matrix1 ▸**Decimal**⇒*Wert*

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

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

$\frac{1}{3}$▸Decimal	0.333333

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 $g(x,y)=2\cdot x-3\cdot y$	*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 *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 $g(x,y)=$Func	*Done*
If $x>y$ Then	
Return x	
Else	
Return y	
EndIf	
EndFunc	
$g(3,-7)$	3

Definie

Katalog >

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
                                Done
g(3,-7)
                 3 greater than -7
                                Done
```

Definiere LibPriv (Define LibPriv)

Katalog >

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)

Katalog >

Define LibPub *Var* **=** *Expression*

Define LibPub *Function*(*Param1*, *Param2*,

Definiere LibPub (Define LibPub) Katalog >

...) = *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 Katalog >

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

DelVar *Var*.

Löscht die angegebene Variable oder Variablengruppe im Speicher.

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 185.

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

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.

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

delVoid()

Katalog >

delVoid(*Liste1***)⇒***Liste*

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

Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

det() (Matrixdeterminante)

Katalog >

det(*Quadratmatrix***[,** *Toleranz***])⇒***Ausdruck*

Gibt die Determinante von *Quadratmatrix* zurück.

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

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

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

det() (Matrixdeterminante) Katalog >

berechnet:

5E-14 ·**max(dim(***Quadratmatrix***))** ·
rowNorm(*Quadratmatrix***)**

diag() (Matrixdiagonale) Katalog >

diag(*Liste***)⇒***Matrix*

diag(*Zeilenmatrix***)⇒***Matrix*

diag(*Spaltenmatrix***)⇒***Matrix*

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

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

diag(*Quadratmatrix***)⇒***Zeilenmatrix*

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

Quadratmatrix muss eine quadratische Matrix sein.

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

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

dim() (Dimension) Katalog >

dim(*Liste***)⇒***Ganzzahl*

Gibt die Dimension von *Liste* zurück.

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

dim(*Matrix***)⇒***Liste*

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

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

dim(*String***)⇒***Ganzzahl*

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

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

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

Disp (Zeige) Katalog >

Disp *AusdruckOderString1* [, *AusdruckOderString2*] ...

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

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

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

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

DispAt Katalog >

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

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

Die Zeilennummer kann als Term angegeben werden.

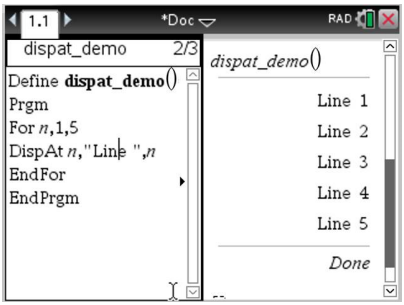
Beachten Sie, dass die Zeilennummer nicht für den gesamten Bildschirm gedacht ist, sondern für den Bereich unmittelbar nach dem Befehl/Programm.

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

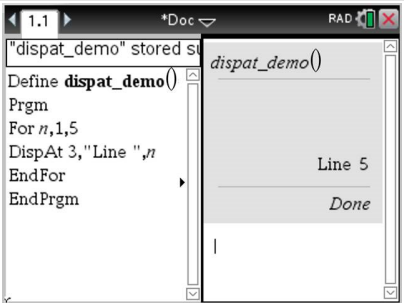
DispAtund Disp können im gleichen Programm verwendet werden.

```
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:

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

DispAt Katalog >

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) Katalog >

Zahl **▸DMS**

Liste **▸DMS**

Matrix **▸DMS**

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

Interpretiert den Parameter als Winkel und zeigt die entsprechenden GMS-Werte (engl. DMS) an (GGGGGG°MM'SS.ss''). Siehe °, ', '' (Seite 217) 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.

Im Grad-Modus:

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

dotP() (Skalarprodukt) Katalog >

dotP(*Liste1***, *Liste2*)⇒*Ausdruck***

Gibt das Skalarprodukt zweier Listen zurück.

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

dotP(*Vektor1***, *Vektor2*)⇒*Ausdruck***

Gibt das Skalarprodukt zweier Vektoren zurück.

Es müssen beide Zeilenvektoren oder beide Spaltenvektoren sein.

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

E

e^() [e^x] Taste

e^(*Wert1***)⇒*Wert***

Gibt **e** hoch *Wert1* zurück.

Hinweis: Siehe auch Vorlage **e Exponent**, Seite 2.

e^{1} 2.71828

e^{3^2} 8103.08

Hinweis: Das Drücken von [e^x] zum Anzeigen von e^(ist nicht das gleiche wie das Drücken von [E] 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***

Gibt **e** hoch jedes Element der *Liste1* zurück.

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

e^(*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()**.

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

e^()

e^x Taste

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

eff()

Katalog >

eff(*Nominalzinssatz, CpY***)⇒***Wert*

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 114.

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

eigVc() (Eigenvektor)

Katalog >

eigVc(*Quadratmatrix***)⇒***Matrix*

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

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

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

Im Komplex-Formatmodus "kartesisch":

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

$\text{eigVc}(m1)$

$$\begin{bmatrix} -0.800906 & 0.767947 & 0 \\ 0.484029 & 0.573804+0.052258\cdot i & 0.5738 \\ 0.352512 & 0.262687+0.096286\cdot i & 0.2626 \end{bmatrix}$$

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

eigVl() (Eigenwert) Katalog >

eigVl(*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":

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

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

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

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)=Func
        If x≤-5 Then
        Return 5
        ElseIf x>-5 and x<0 Then
        Return -x
        ElseIf x≥0 and x≠10 Then
        Return x
        ElseIf x=10 Then
        Return 3
        EndIf
        EndFunc
                                Done
```

EndFor Siehe For, Seite 59.

EndFunc Siehe Func, Seite 64.

EndIf

Siehe If, Seite 76.

EndLoop

Siehe Loop, Seite 99.

EndWhile

Siehe While, Seite 189.

EndPrgm

Siehe Prgm, Seite 128.

EndTry

Siehe Try, Seite 178.

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

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

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) und y(0)=10

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

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

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

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

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 **sign(***VarSchritt***)** = **sign (***VarMax-Var0***)** und Lösungen an *Var0*+i·*VarSchritt* für alle i=0,1,2,… zurückgegeben werden, sodass *Var0*+i·*VarSchritt* in [*var0*,*VarMax*] ist (möglicherweise gibt es keinen Lösungswert an *VarMax*).

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

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

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

eval () Hub-Menü

eval(*Expr***)** ⇒ *Zeichenfolge*

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

eval () Hub-Menü

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.

```
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
```

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

```
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"
```

Hinweis: Siehe auch **Get** (Seite 65), **GetStr** (Seite 73) und **Send** (Seite 151).

Exit (Abbruch) Katalog >

Exit (Abbruch)

Funktionslisting:

Exit (Abbruch) Katalog >

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

exp() (e hoch x) e^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 θ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

e^1 2.71828

e^{3^2} 8103.08

exp(*Liste1***)⇒***Liste*

Gibt **e** hoch jedes Element der *Liste1* zurück.

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

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.

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

expr() (String in Ausdruck) Katalog >

expr(*String***)⇒***Ausdruck*

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

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

ExpReg (Exponentielle Regression) Katalog >

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

Berechnet die exponentielle Regressiony = a ·(b)ˣauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a ·(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 *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten *Y List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

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.

So halten Sie eine Berechnung manuell an:

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

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

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.

FCdf() Katalog >

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

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

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

Für P($X \le$ *ObereGrenze*), *UntGrenze* =0 setzen.

Fill (Füllen) Katalog >

Fill *Zahl, MatrixVar*⇒*Matrix*

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

MatrixVar muss bereits vorhanden sein.

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

$$\text{Fill } 1.01, amatrix \qquad Done$$

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

Fill *Zahl, ListeVar*⇒*Liste*

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

ListeVar muss bereits vorhanden sein.

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

$$\text{Fill } 1.01, alist \qquad Done$$

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

FiveNumSummary Katalog >

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

Bietet eine gekürzte Version der Statistik mit 1 Variablen auf Liste *X*.
Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.MinX	Minimum der x-Werte
stat.Q_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte

floor() (Untergrenze) Katalog >

floor(*Wert1***)**⇒*Ganzzahl*

```
floor(-2.14)                    -3.
```

floor() (Untergrenze) Katalog >

Gibt die größte ganze Zahl zurück, die ≤ dem Argument ist. Diese Funktion ist identisch mit **int()**.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

floor(*Liste1***)⇒***Liste*

floor(*Matrix1***)⇒***Matrix*

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

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

$\text{floor}\left(\left\{\frac{3}{2},0,{}^{-}5.3\right\}\right)$	$\{1,0,{}^{-}6.\}$
$\text{floor}\left(\begin{bmatrix}1.2 & 3.4\\ 2.5 & 4.8\end{bmatrix}\right)$	$\begin{bmatrix}1. & 3.\\ 2. & 4.\end{bmatrix}$

For Katalog >

For *Var*, *Von*, *Bis* [, *Schritt*]
Block
EndFor

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

Var darf keine Systemvariable sein.

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

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

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

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

format() (Format) Katalog >

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(*Ausdr1***)⇒***Ausdruck*

fPart(*Liste1***)⇒***Liste*

fPart(*Matrix1***)⇒***Matrix*

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

fPart() (Funktionsteil) Katalog >

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 >

FPdf
(*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
(*Liste1***,***HäufGanzzahlListe***)**⇒*Liste*

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

Liste1 kann eine beliebige gültige Liste sein.

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

freqTable▸list() Katalog >

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 240).

frequency() (Häufigkeit) Katalog >

frequency(*Liste1*,*binsListe***)⇒***Liste*

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

Wenn *binsListe* {b(1), b(2), ..., b(n)} ist, sind die festgelegten Bereiche {**?**≤b(1), b (1)<**?**≤b(2),...,b(n-1)<**?**≤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, **?**≤b(1)), countIf(Liste, b(1)<**?**≤b(2)), ..., countIf(Liste, b(n-1)<**?**≤b(n)), countIf (Liste, b(n)>**?**)}.

Elemente von *Liste1*, die nicht "in einem bin platziert" werden können, werden ignoriert. Leere (ungültige) Elemente werden ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

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.

frequency() (Häufigkeit) Katalog >

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

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

FTest_2Samp (Zwei-Stichproben F-Test) Katalog >

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 165.)

Für H_a: σ1 > σ2 setzen Sie *Hypoth*>0

Für H_a: σ1 ≠ σ2 (Standard) setzen Sie *Hypoth* =0

Für H_a: σ1 < σ2 setzen Sie *Hypoth*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 240).

Ausgabevariable	Beschreibung
Statistik.F	Berechnete Û Statistik für die Datenfolge
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.dfNumer	Freiheitsgrade des Zählers = n1-1
stat.dfDenom	Freiheitsgrade des Nenners = n2-1
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*

Ausgabevariable	Beschreibung
stat.x1_bar stat.x2_bar	Stichprobenmittelwerte der Datenfolgen in *Liste 1* und *Liste 2*
stat.n1, stat.n2	Stichprobenumfang

Func

Katalog >

Func
Block
EndFunc

Vorlage zur Erstellung einer benutzerdefinierten Funktion.

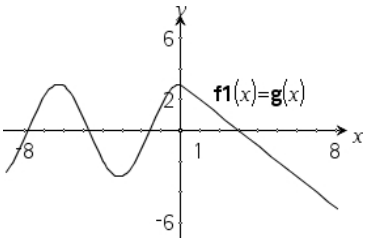
Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen ":" voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein. Die Funktion kann die Anweisung **Zurückgeben (Return)** verwenden, um ein bestimmtes Ergebnis zurückzugeben.

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*

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(18,33)                          3
```

gcd(*Liste1, Liste2***)⇒***Liste*

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

gcd() (Größter gemeinsamer Teiler) Katalog >

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

gcd(*Matrix1, Matrix2***)⇒***Matrix*

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

$$\gcd\left(\begin{bmatrix}2 & 4\\6 & 8\end{bmatrix},\begin{bmatrix}4 & 8\\12 & 16\end{bmatrix}\right) \qquad \begin{bmatrix}2 & 4\\6 & 8\end{bmatrix}$$

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≤X ≤*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 ≤ *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 *XWert*, die Anzahl der Einzelversuche, bis der erste Erfolg eingetreten ist, für die diskrete geometrische Verteilung mit der vorgegebenen Erfolgswahrscheinlichkeit p.

Get Hub-Menü

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

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

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.

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.

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

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

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

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

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

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

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

Get — Hub-Menü

Hinweis: Siehe auch **GetStr**, Seite 73 und **Send**, Seite 151.

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

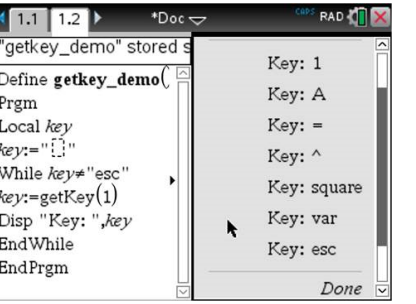
Beschreibung:getKey() – ermöglicht ein TI-Basic-Programm zum Holen von Tastatureingaben – Handheld, Desktop und Emulator auf Desktop.

Beispiel:

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

```
getKey()
```

Beispiel:



Handhabung von Tastenbetätigungen:

Handheld/Emulatortaste	Desktop	Rückgabewert
Esc	Esc	„Esc“
Touchpad – Oben klicken	–	„nach oben“
Ein	–	„Hauptmenü“

Handheld/Emulatortaste	Desktop	Rückgabewert
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“
Vorlagen	–	„Vorlage“
Katalog	–	„cat“
^	^	"^"
X^2	–	„Quadrat“
/ (Divisionstaste)	/	"/"
* (Multiplikationstaste)	*	"*"
e^x	–	„Ausdr“
10^x	–	„10power“
+	+	"+"
-	-	"-"
(	(	"("
)	)	")"

Handheld/Emulatortaste	Desktop	Rückgabewert
.	.	"."
(-)	–	„-“ (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
,	,	","
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*

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.

```
getLangInfo()                    "en"
```

getLangInfo() Katalog >

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 95, und**unLock**, Seite 185.

a:=65	65
Lock a	*Done*
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	*Done*
a:=75	75
DelVar a	*Done*

getMode() Katalog >

getMode(*ModusNameGanzzahl***)**⇒*Wert*

getMode(0)⇒*Liste*

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

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

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 154.

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*]

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

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

Hinweis: Siehe auch **Get**, Seite 65 und **Send**, Seite 151.

Zum Beispiel siehe **Get**.

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.

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

getVarInfo()⇒*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.

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.

getVarInfo()	"NONE"
Define x=5	*Done*
Lock x	*Done*
Define LibPriv $y=\{1,2,3\}$	*Done*
Define LibPub $z(x)=3\cdot x^2-x$	*Done*
getVarInfo()	$\begin{bmatrix} x & \text{"NUM"} & \text{"}\square\text{"} & 1 \\ y & \text{"LIST"} & \text{"LibPriv "} & 0 \\ z & \text{"FUNC"} & \text{"LibPub "} & 0 \end{bmatrix}$
getVarInfo(*tmp3*)	"Error: Argument must be a string"
getVarInfo("tmp3")	$\begin{bmatrix} volcyl2 & \text{"NONE"} & \text{"LibPub "} & 0 \end{bmatrix}$

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

Goto (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                    Done
          Local temp,i
          0→temp
          1→i
          Lbl top
          temp+i→temp
          If i<10 Then
          i+1→i
          Goto top
          EndIf
          Return temp
          EndFunc
g()                                55
```

▸Grad (Neugrad) Katalog >

Ausdr1 **▸Grad**⇒*Ausdruck*

Wandelt *Ausdr1* ins Winkelmaß Neugrad um.

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

Im Grad-Modus:

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

Im Bogenmaß-Modus:

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

I

identity() Katalog >

identity(*Ganze Zahl***)** ⇒ *Matrix*

Gibt die Einheitsmatrix mit der Dimension *Ganzzahl* zurück.

Ganzzahl muss eine positive ganze Zahl sein.

```
identity(4)
```

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

If *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                    Done
              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.

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

If

Katalog >

If *BooleanExpr1* **Then**
Block1
ElseIf *BooleanExpr2* **Then**
Block2
⋮
ElseIf *BooleanExprN* **Then**
BlockN
EndIf

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

```
Define g(x)=Func
        If x<-5 Then
        Return 5
        ElseIf x>-5 and x<0 Then
        Return -x
        ElseIf x≥0 and x≠10 Then
        Return x
        ElseIf x=10 Then
        Return 3
        EndIf
        EndFunc
                                Done
g(-4)                              4
g(10)                              3
```

ifFn()

Katalog >

ifFn(*BoolescherAusdruck***,***Wert_wenn_wahr* [**,***Wert_wenn_falsch* [**,***Wert_wenn_unbekannt*]]**)** ⇒ *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.
- 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

```
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

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.

```
ifFn({1,2,3}<2.5,4,{8,9,10})        {4,4,10}
```

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

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

ifFn() Katalog >

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_falsch ist nicht spezifiziert. Undef wird verwendet.

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

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

imag() Katalog >

imag(*Value1***)** ⇒ *Wert*

Gibt den Imaginärteil des Arguments zurück.

```
imag(1+2·i)                                2
```

imag(*List1***)** ⇒ *Liste*

Gibt eine Liste der Imaginärteile der Elemente zurück.

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

imag(*Matrix1***)** ⇒ *Matrix*

Gibt eine Matrix der Imaginärteile der Elemente zurück.

```
imag([1    2 ])                  [0  0]
     [i·3  i·4]                  [3  4]
```

Umleitung Siehe #(), Seite 214.

inString() Katalog >

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

Gibt die Position des Zeichens von *Quellstring* zurück, an der das erste Vorkommen von *Teilstring* beginnt.

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

inString() Katalog >

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*

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.

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

intDiv() Katalog >

intDiv(*Zahl1***,** *Zahl2***)** ⇒ *Ganzzahl*
intDiv(*Liste1***,** *Liste2***)** ⇒ *Liste*
intDiv(*Matrix1***,** *Matrix2***)** ⇒ *Matrix*

Gibt den mit Vorzeichen versehenen ganzzahligen Teil von (*Zahl1* ÷ *Zahl2*) zurück.

Für eine Liste oder Matrix wird für jedes Elementpaar der mit Vorzeichen versehene ganzzahlige Teil von (Argument1÷Argument2) zurückgegeben.

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

Interpolieren () Katalog >

Interpolieren(*xWert***,** *xListe***,** *yListe***,** *yStrListe***)** ⇒ *Liste*

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

Interpolieren ()

Katalog >

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.

```
rk:=rk23(-3·y+6·t+5,t,y,{0,10},5,1)
[0.      1.        2.        3.        4.        ▸
 5.  3.19499  5.00394  6.99957  9.00593  10
```

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ◀ und ▶, 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▸list(rk[1])
                {0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=mat▸list(rk[2])
 {5.,3.19499,5.00394,6.99957,9.00593,10.9978▸
yprimelist:=-3·y+6·t+5|y=ylist and t=xlist
 {-10.,1.41503,1.98819,2.00129,1.98221,2.006▸
interpolate(xvaluelist,xlist,ylist,yprimelist)
 {5.,2.67062,3.19499,4.02782,5.00394,6.00011▸
```

invχ2()

Katalog >

invχ2(*Fläche***,***FreiGrad***)**

invChi2(*Fläche***,***FreiGrad***)**

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

invF()

Katalog >

invF (*Fläche***,***FreiGradZähler***,***FreiGradNenner***)**

invF (*Fläche***,***FreiGradZähler***,***FreiGradNenner***)**

invF() Katalog >

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

invBinom() Katalog >

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

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

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

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***)⇒** *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.

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?

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

invBinomN() Katalog >

OutputForm=**1**, gibt Ergebnis als Matrix an.

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***)** ⇒ *Ganzzahl*
iPart(*Liste1***)** ⇒ *Liste*
iPart(*Matrix1***)** ⇒ *Matrix*

Gibt den ganzzahligen Teil des Arguments zurück.

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

Das Argument kann eine reelle oder eine komplexe Zahl sein.

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

irr() Katalog >

irr(*CF0*,*CFListe* [,*CFFreq*]**)** ⇒ *Wert*

Finanzfunktion, die den internen Zinsfluss einer Investition berechnet.

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

irr() Katalog >

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

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

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

Hinweis: Siehe auch **mirr()**, Seite 105.

isPrime() Katalog >

isPrime(*Zahl***)** ⇒ *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.

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

isPrime(5)	true
isPrime(6)	false

Funktion zum Auffinden der nächsten Primzahl nach einer angegebenen Zahl:

Define *nextprim*(*n*)=	Func	*Done*
	Loop	
	n+1→*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}

isVoid() Katalog >

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

Weitere Informationen zu ungültigen Elementen finden Sie auf Seite Seite 240.

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.

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→temp
        1→i
        Lbl top
        temp+i→temp
        If i<10 Then
        i+1→i
        Goto top
        EndIf
        Return temp
        EndFunc
g()                                55
```

lcm() (Kleinstes gemeinsames Vielfaches) Katalog >

lcm(*Zahl1***,** *Zahl2***)⇒***Ausdruck*

lcm(*Liste1***,** *Liste2***)⇒***Liste*

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.

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

lcm() (Kleinstes gemeinsames Vielfaches)

Katalog >

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

left() (Links)

Katalog >

left(*Quellstring*[**,** *Anz*]**)⇒***String*

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

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

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

left(*Liste1*[**,** *Anz*]**)⇒***Liste*

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

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

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

left(*Vergleich***)⇒***Ausdruck*

Gibt die linke Seite einer Gleichung oder Ungleichung zurück.

libShortcut()

Katalog >

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)

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}
```

libShortcut() Katalog >

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).

LinRegBx Katalog >

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

Berechnet die lineare Regressiony = a+b ·xauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a+b ·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 *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

LinRegMx

Katalog >

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

Berechnet die lineare Regression y = m ·x+b auf Liste *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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.

LinRegMx

Katalog >

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 240).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: m ·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 *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

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 165.)

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 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a+b ·x
stat.a, stat.b	Regressionskoeffizienten
stat.df	Freiheitsgrade
stat.r^2	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
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,	Vorhersageintervall für eine einzelne Beobachtung

Ausgabevariable	Beschreibung
stat.UpperPred]	
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat.$\hat{y}$	a + b ·XWert

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=α+βx. Er berechnet die Null-Hypothese H_0:β=0 (gleichwertig, ρ=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:β=ρ=0) untersucht wird.

Für H_a: β 0 und ρ 0 (Standard) setzen Sie $Hypoth$=0

Für H_a: β<0 und ρ<0 setzen Sie $Hypoth$<0

Für H_a: β>0 und ρ>0 setzen Sie $Hypoth$>0

Eine Zusammenfassung der Ergebnisse wird in der Variablen $stat.results$ gespeichert. (Seite 165.)

LinRegtTest (t-Test bei linearer Regression)

Katalog >

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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

linSolve()

Katalog >

linSolve(*SystemLinearerGl*, *Var1*, *Var2*, ...**)⇒***Liste*

linSolve(*LineareGl1* **and** *LineareGl2* **and** ..., *Var1*, *Var2*, ...**)⇒***Liste*

linSolve({*LineareGl1*, *LineareGl2*, ...**},** *Var1*, *Var2*, ...**) ⇒***Liste*

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

linSolve(*LineareGl1* **and** *LineareGl2* **and** ..., **{***Var1*, *Var2*, ...**})⇒***Liste*

linSolve({*LineareGl1*, *LineareGl2*, ...**},** **{***Var1*, *Var2*, ...**}) ⇒***Liste*

Liefert eine Liste mit Lösungen für die Variablen *Var1*, *Var2*, ...

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

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

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

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

linSolve() Katalog >

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*

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*.

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

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.

$\text{list}\blacktriangleright\text{mat}(\{1,2,3\})$ $\begin{bmatrix}1 & 2 & 3\end{bmatrix}$

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

ln() (Natürlicher Logarithmus) ctrl e^x Tasten

ln(*Wert1***)⇒***Wert*

ln(*Liste1***)⇒***Liste*

Gibt den natürlichen Logarithmus des Arguments zurück.

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

```
ln(2.)                    0.693147
```

Bei Komplex-Formatmodus reell:

```
ln({-3,1.2,5})
"Error: Non-real calculation"
```

Bei Komplex-Formatmodus kartesisch:

```
ln({-3,1.2,5})
{1.09861+3.14159·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()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

```
ln([1 5 3; 4 2 1; 6 -2 1])
[ 1.83145+1.73485·i   0.009193-1.49086
  0.448761-0.725533·i  1.06491+0.623491▸
  -0.266891-2.08316·i  1.12436+1.79018· ]
```

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

LnReg Katalog >

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

Berechnet die logarithmische Regression y = a+b ·ln(x) auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a+b ·ln(x)
stat.a, stat.b	Regressionskoeffizienten
stat.r^2	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten (ln(x), y)
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 *X-Liste*, die in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

Local (Lokale Variable) Katalog >

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

Deklariert die angegebenen Variablen *Variable* als lokale Variablen. Diese Variablen existieren nur während der Auswertung einer Funktion und werden gelöscht, wenn die Funktion beendet wird.

Hinweis: Lokale Variablen sparen Speicherplatz, da sie nur temporär existieren. Außerdem stören sie keine vorhandenen globalen Variablenwerte. Lokale Variablen müssen für **For**-Schleifen und für das temporäre Speichern von Werten in mehrzeiligen Funktionen verwendet werden, da Änderungen globaler Variablen in einer Funktion unzulässig sind.

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

```
Define rollcount()=Func
                   Local i
                   1→i
                   Loop
                   If randInt(1,6)=randInt(1,6)
                   Goto end
                   i+1→i
                   EndLoop
                   Lbl end
                   Return i
                   EndFunc
                                          Done
rollcount()                                16
rollcount()                                3
```

Lock Katalog >

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

Lock*Var***.**

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

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

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

Siehe **unLock, Seite 185, und getLockInfo(), Seite 71.**

```
a:=65                                      65
Lock a                                     Done
getLockInfo(a)                             1
a:=75                   "Error: Variable is locked."
DelVar a                "Error: Variable is locked."
Unlock a                                   Done
a:=75                                      75
DelVar a                                   Done
```

log() (Logarithmus) ctrl 10^x Tasten

log(*Wert1***[,***Wert2***])⇒***Wert*

log(*Liste1***[,***Wert2***])⇒***Liste*

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

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

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

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

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

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

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

Bei Komplex-Formatmodus reell:

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

"Error: Non−real calculation"

Bei Komplex-Formatmodus kartesisch:

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

$\{0.477121+1.36438\cdot i,0.079181,0.69897\}$

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

Gibt den Matrix-Logarithmus von *Zahl2* zur Basis *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Logarithmus jedes Elements zur Basis *Zahl2*. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\log_{10}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

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

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

Logistic Katalog >

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

Berechnet die logistische Regressiony = (c/(1+a ·e^-bx))auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

Logistic Katalog >

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 240).

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

LogisticD Katalog >

LogisticD *X*, *Y* [, [*Iterationen*], [*Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die logistische Regression y = (c/(1+a ·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 165.)

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 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: c/(1+a ·e^{-bx})+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 *X-Liste*, die in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit -Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit -Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

Loop (Schleife) Katalog >

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
                                          Done
rollcount()                                 16
rollcount()                                  3
```

LU *Matrix*, *lMatrix*, *uMatrix*, *pMatrix* [,*Tol*]

Berechnet die Doolittle LU-Zerlegung (LR-Zerlegung) einer reellen oder komplexen Matrix. Die untere (bzw. linke) Dreiecksmatrix ist in *lMatrix* gespeichert, die obere (bzw. rechte) Dreiecksmatrix in *uMatrix* und die Permutationsmatrix (in welcher der bei der Berechnung vorgenommene Zeilentausch dokumentiert ist) in *pMatrix*.

lMatrix · *uMatrix* = *pMatrix* · *Matrix*

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

- Wenn Sie ctrl enter verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
  5E-14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

Der **LU**-Faktorisierungsalgorithmus verwendet partielle Pivotisierung mit Zeilentausch.

$$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$$

LU *m1*,*lower*,*upper*,*perm* — *Done*

lower $\begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$

upper $\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$

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

mat▸list() (Matrix in Liste) Katalog >

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.

$\text{mat}\blacktriangleright\text{list}\left(\begin{bmatrix}1 & 2 & 3\end{bmatrix}\right)$ $\{1,2,3\}$

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

$\text{mat}\blacktriangleright\text{list}(m1)$ $\{1,2,3,4,5,6\}$

max() (Maximum) Katalog >

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.

$\max(2.3,1.4)$ 2.3

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

max(*Liste***)⇒***Ausdruck***

Gibt das größte Element von *Liste* zurück.

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

max(*Matrix1***)⇒***Matrix***

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

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

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

mean() (Mittelwert) Katalog >

mean(*Liste***[,** *Häufigkeitsliste***])⇒***Ausdruck*

Gibt den Mittelwert der Elemente in *Liste* zurück.

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

$\text{mean}(\{0.2,0,1,-0.3,0.4\})$ 0.26

$\text{mean}(\{1,2,3\},\{3,2,1\})$ $\frac{5}{3}$

mean(*Matrix1***[,** *Häufigkeitsmatrix***])⇒***Matrix*

Ergibt einen Zeilenvektor aus den Mittelwerten aller Spalten in *Matrix1*.

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

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

Im Vektorformat kartesisch:

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

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

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

median() (Median) Katalog >

median(*Liste***[,** *freqList***])⇒***Ausdruck*

Gibt den Medianwert der Elemente in *Liste* zurück.

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

$\text{median}(\{0.2,0,1,-0.3,0.4\})$ 0.2

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

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

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

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

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 240).

MedMed

Katalog > 📖

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

Berechnet die Median-Median-Liniey = (m ·x+b)auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.RegEqn	Median-Median-Linien-Gleichung: m ·x+b
stat.m, stat.b	Modellkoeffizienten
stat.Resid	Residuen von der Median-Median-Linie

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

mid() (Teil-String) Katalog >

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

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

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

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

min() (Minimum) Katalog >

min(*Wert1***, *Wert2*)⇒*Ausdruck***

min(*Liste1, Liste2*)⇒*Liste***

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

min(*Liste***)⇒*Ausdruck***

Gibt das kleinste Element von *Liste* zurück.

min({0,1,⁻7,1.3,0.5})	⁻7

min(*Matrix1***)⇒*Matrix***

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

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

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

mirr() Katalog >

mirr (*Finanzierungsrate***,***Reinvestitionsrate***, *CF0*,***CFListe***[,***CFFreq***])**

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.

list1:={6000,⁻8000,2000,⁻3000}	{6000,⁻8000,2000,⁻3000}
list2:={2,2,2,1}	{2,2,2,1}
mirr(4.65,12,5000,*list1*,*list2*)	13.41608607

mirr() Katalog >

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 82.

mod() (Modulo) Katalog >

mod(*Wert1***,** *Wert2***)⇒***Ausdruck*

mod(*Liste1***,** *Liste2***)⇒***Liste*

mod(*Matrix1***,** *Matrix2***)⇒***Matrix*

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

mod(x,0) = x

mod(x,y) = x − y floor(x/y)

Ist das zweite Argument ungleich Null, ist das Ergebnis in diesem Argument periodisch. Das Ergebnis ist entweder Null oder besitzt das gleiche Vorzeichen wie das zweite Argument.

Sind die Argumente zwei Listen bzw. zwei Matrizen, wird eine Liste bzw. Matrix zurückgegeben, die den Modulus jedes Elementpaars enthält.

Hinweis: Siehe auch **remain()**, Seite 140

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

mRow() (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.

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

mRowAdd() (Matrixzeilenaddition) Katalog >

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

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

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

Zahl × Zeile *Index1* + Zeile *Index2*

MultReg Katalog >

MultReg *Y***,** *X1*[**,***X2*[**,***X3***,…**[,***X10*]]]

Berechnet die lineare Mehrfachregression der Liste *Y* für die Listen *X1*, *X2*, …, *X10*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ ...
stat.b0, stat.b1, ...	Regressionskoeffizienten
stat.R^2	Multiples Bestimmtheitsmaß
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ ...
stat.Resid	Residuen von der Regression

MultRegIntervals Katalog >

MultRegIntervals *Y***,** *X1*[**,***X2*[**,***X3***,…** [**,***X10*]]]**,***XWertListe*[**,***KNiveau*]

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

MultRegIntervals

Katalog >

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ ...
stat.$\hat{y}$	Eine Punktschätzung: $\hat{y}$ = b0 + b1 · xl + ... für *XWertListe*
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
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 165.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgaben

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ ...
stat.F	Globale F-Testgröße
stat.PVal	Mit globaler F-Statistik verknüpfter P-Wert
stat.R^2	Multiples Bestimmtheitsmaß
stat.AdjR^2	Angepasster Koeffizient des multiplen Bestimmtheitsmaßes
stat.s	Standardabweichung des Fehlers
stat.DW	Durbin-Watson-Statistik; bestimmt, ob in dem Modell eine Autokorrelation erster Ordnung vorhanden ist
stat.dfReg	Regressions-Freiheitsgrade
stat.SSReg	Summe der Regressionsquadrate
stat.MSReg	Mittlere Regressionsstreuung
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittleres Fehlerquadrat
stat.bList	{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.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ . . .
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)

nand

ctrl = Tasten

BoolescherAusdr1 **nand** *BoolescherAusd2* 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* ≥ *Wert2* ≥ 0 ist **nCr()** die Anzahl der Möglichkeiten, *Wert1* Elemente aus *Wert2* Elementen auszuwählen (auch als Binomialkoeffizient bekannt).

$\mathrm{nCr}(z,3)\|z=5$	10
$\mathrm{nCr}(z,3)\|z=6$	20

nCr(*Wert***, 0)⇒1**

nCr(*Wert***,** *negGanzzahl***)⇒0**

nCr(*Wert***,** *posGanzzahl***)⇒** *Wert* · **(***Wert***−1)... (***Wert***−***posGanzzahl***+1)/** *posGanzzahl***!**

nCr(*Wert***,** *keineGanzzahl***)⇒***Ausdruck***!/ ((***Wert***−***keineGanzzahl***)! ·** *keineGanzzahl***!)**

nCr(*Liste1***,** *Liste2***)⇒***Liste*

Gibt eine Liste von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

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

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.

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

nDerivative() Katalog >

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

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

Gibt die numerische Ableitung zurück, berechnet durch automatische Ableitungsmethoden.

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

nDerivative() Katalog >

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.

Hierzu rechts ein Beispiel. Die erste Ableitung von x·(x^2+x)^(1/3) bei x=0 ist gleich 0. Nun ist allerdings die erste Ableitung des Unterausdrucks (x^2+x)^(1/3) bei x=0 nicht definiert. Dieser Wert wird gleichzeitig jedoch verwendet, um die Ableitung des Gesamtausdrucks zu berechnen. Daher meldet **nDerivative()** das Ergebnis als nicht definiert und zeigt eine Warnmeldung an.

Wenn Sie bei der Arbeit auf diese Einschränkung stoßen, prüfen Sie die Lösung grafisch. Ggf. können Sie es auch mit **centralDiff()** probieren.

$$\text{nDerivative}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x,1\right)\Big|x=0 \qquad \text{undef}$$

$$\text{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)\Big|x=0 \qquad 0.000033$$

newList() (Neue Liste) Katalog >

newList(*AnzElemente***)⇒***Liste*

Gibt eine Liste der Dimension *AnzElemente* zurück. Jedes Element ist Null.

$$\text{newList}(4) \qquad \{0,0,0,0\}$$

newMat() (Neue Matrix) Katalog >

newMat(*AnzZeil***,** *AnzSpalt***)⇒***Matrix*

Gibt eine Matrix der Dimension *AnzZeil* mal *AnzSpalt* zurück, wobei die Elemente Null sind.

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

nfMax() (Numerisches Funktionsmaximum) Katalog >

nfMax(*Ausdr***,** *Var***)⇒***Wert*

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

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

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

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Maximum von *Ausdr* auftritt.

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Invervall [*UntereGrenze*,*ObereGrenze*] für das lokale Maximum.

$\text{nfMax}\left(-x^2-2\cdot x-1,x\right)$	-1.
$\text{nfMax}\left(0.5\cdot x^3-x-2,x,-5,5\right)$	5.

nfMin() (Numerisches Funktionsminimum) Katalog >

nfMin(*Ausdr***,** *Var***)⇒***Wert*

nfMin(*Ausdr***,** *Var, UntereGrenze***)⇒***Wert*

nfMin(*Ausdr***,** *Var, UntereGrenze***,** *ObereGrenze***)⇒***Wert*

nfMin(*Ausdr, Var***) |** *UntereGrenze* ≤*Var*≤*ObereGrenze*⇒*Wert*

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Minimum von *Ausdr* auftritt.

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

nfMin() (Numerisches Funktionsminimum)

Katalog >

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Invervall [*UntereGrenze*,*ObereGrenze*] für das lokale Minimum.

nInt() (Numerisches Integral)

Katalog >

nInt(*Ausdr1, Var, Untere, Obere***)⇒***Ausdruck*

$\text{nInt}\left(e^{-x^2},x,-1,1\right)$ 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 ∫**(***Ausdr1, Var, Untere, Obere***)** zurück. Diese Näherung ist der gewichtete Durchschnitt von Stichprobenwerten des Integranden im Intervall *Untere*<*Var*<*Obere*.

Das Berechnungsziel sind sechs signifikante Stellen. Der angewendete Algorithmus beendet die Weiterberechnung, wenn das Ziel hinreichend erreicht ist oder wenn weitere Stichproben wahrscheinlich zu keiner sinnvollen Verbesserung führen.

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

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.

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

nom()

Katalog >

nom(*Effektivzins,CpY***)⇒***Wert*

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

nom() Katalog >

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 ctrl = Tasten

BoolescherAusd1 **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*⇒*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.

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}

nor

ctrl = Tasten

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).

norm()

Katalog >

norm(*Matrix***)⇒***Ausdruck*

norm(*Vektor***)⇒***Ausdruck*

Gibt die Frobeniusnorm zurück.

$\text{norm}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$	5.47723
$\text{norm}\left(\begin{bmatrix}1 & 2\end{bmatrix}\right)$	2.23607
$\text{norm}\left(\begin{bmatrix}1\\ 2\end{bmatrix}\right)$	2.23607

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog >

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

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

Für P(X ≤ *obereGrenze*) setzen Sie *untereGrenze* = −9E999.

normPdf() (Wahrscheinlichkeitsdichte)

Katalog >

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

Berechnet die Wahrscheinlichkeitsdichtefunktion für die Normalverteilung an einem bestimmten *XWert* für die vorgegebenen μ und σ.

not (nicht) Katalog >

not
BoolescherAusdr1⇒
BoolescherAusdruck

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

not (2≥3)	true
not 0hB0▸Base16	0hFFFFFFFFFFFFFFF4F
not not 2	2

not *Ganzzahl1⇒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).

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.

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O.

not 0h7AC36	0hFFFFFFFFFFF853C9

Im Bin-Modus:

0b100101▸Base10	37
not 0b100101	0b1111111111111111111111111111111▸
not 0b100101▸Base10	⁻38

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) Katalog >

nPr(*Wert1***, *Wert2*)⇒***Ausdruck*

Für ganzzahlige *Wert1* und *Wert2* mit *Wert1* ≥ *Wert2* ≥ 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([6 5; 4 3],[2 2; 2 2])	[30 20; 12 6]

nPr(*Wert***, 0)⇒1**

nPr(*Wert***, *negGanzzahl*) ⇒ 1/((***Wert***+1)·(***Wert***+2)…(***Wert***−*negGanzzahl*))**

nPr() (Permutationen) Katalog >

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.

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

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.

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

npv() Katalog >

npv(*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.

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

CFListe ist eine Liste der Cash-Flow-Beträge nach dem anfänglichen Cash-Flow *CF0*.

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

$list1:=\{6000,{}^{-}8000,2000,{}^{-}3000\}$ $\{6000,{}^{-}8000,2000,{}^{-}3000\}$

$list2:=\{2,2,2,1\}$ $\{2,2,2,1\}$

$\text{npv}(10,5000,list1,list2)$ 4769.91

nSolve() (Numerische Lösung) Katalog >

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*

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.

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

Hinweis: Existieren mehrere Lösungen, können Sie mit Hilfe einer Schätzung eine bestimmte Lösung suchen.

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.

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

O

OneVar (Eine Variable) Katalog >

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 165.)

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 240).

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

Ausgabevariable	Beschreibung
stat.Q_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.SSX	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert

or (oder) Katalog >

BoolescherAusd1 **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.

```
Define g(x)=Func                    Done
        If x≤0 or x≥5
        Goto end
        Return x·3
        Lbl end
        EndFunc
g(3)                                9
g(0)    A function did not return a value
```

Ganzzahl1 **or** *Ganzzahl2*⇒*Ganzzahl*

Im Hex-Modus:

```
0h7AC36 or 0h3D5F                   0h7BD7F
```

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

```
0b100101 or 0b100                   0b100101
```

or (oder) Katalog >

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

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

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

Hinweis: Siehe **xor.**

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

ord() (Numerischer Zeichencode) Katalog >

ord(*String***)⇒***Ganzzahl*

ord(*Liste1***)⇒***Liste*

Gibt den Zahlenwert (Code) des ersten Zeichens der Zeichenkette *String* zurück. Handelt es sich um eine Liste, wird der Code des ersten Zeichens jedes Listenelements zurückgegeben.

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

P

P▶Rx() (Kartesische x-Koordinate) Katalog >

P▶Rx(*rAusdr***, θ***Ausdr***)⇒***Ausdruck*

Im Bogenmaß-Modus:

P▸Rx() (Kartesische x-Koordinate) Katalog >

P▸Rx(*rListe***, θ***Liste***)⇒***Liste***

P▸Rx(*rMatrix***, θ***Matrix***)⇒***Matrix***

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

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

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

$P\blacktriangleright Rx(4,60°)$ 2.

$P\blacktriangleright Rx\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$ $\{-1.5,7.07107,1.3\}$

P▸Ry() (Kartesische y-Koordinate) Katalog >

P▸Ry(*rWert***, θ***Wert***)⇒***Wert***

P▸Ry(*rListe***, θ***Liste***)⇒***Liste***

P▸Ry(*rMatrix***, θ***Matrix***)⇒***Matrix***

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

Hinweis: Das θ-Argument wird gemäß deraktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert.

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

Im Bogenmaß-Modus:

$P\blacktriangleright Ry(4,60°)$ 3.4641

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

PassErr (ÜbgebFeh) Katalog >

PassErr

Übergibt einen Fehler an die nächste Stufe.

Wenn die Systemvariable *Fehlercode (errCode)* Null ist, tut **PassErr** nichts.

Ein Beispiel zu **PassErr** finden Sie im Beispiel 2 unter Befehl **Versuche (Try)**, Seite 178.

PassErr (ÜbgebFeh) Katalog >

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 178.

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* [**,** ...]]]]**)**

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 $p(x)=\begin{cases} x, & x>0 \\ \text{undef}, & x\le 0 \end{cases}$	*Done*
$p(1)$	1
$p(-1)$	undef

poissCdf() Katalog >

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

poissCdf(λ**,***obereGrenze***)**(für P(0≤X ≤*obereGrenze*)⇒*Zahl*, wenn *obereGrenze* eine Zahl ist, Liste, wenn *obereGrenze* eine Liste ist

poissCdf() Katalog >

Berechnet die kumulative Wahrscheinlichkeit für die diskrete Poisson-Verteilung mit dem vorgegebenen Mittelwert λ.

Für P(X ≤ *obereGrenze*) setzen Sie *untereGrenze* = 0

poissPdf() Katalog >

poissPdf(λ,*XWert***)⇒***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**

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

Zeigt *Vektor* in Polarform [r∠ θ] an. Der Vektor muss die Dimension 2 besitzen und kann eine Zeile oder eine Spalte sein.

Hinweis: ▸Polar ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen, und sie nimmt keine Aktualisierung von *ans* vor.

Hinweis: Siehe auch **▸Rect**, Seite 137.

$[1 \quad 3.]\blacktriangleright\text{Polar} \qquad [3.16228 \quad \angle 71.5651]$

komplexerWert **▸Polar**

Zeigt *komplexerVektor* in Polarform an.

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

Im Bogenmaß-Modus:

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad e^{0.927295\cdot i}\cdot 5$

$\left\{\left(4 \angle \frac{\pi}{3}\right)\right\}\blacktriangleright\text{Polar} \qquad e^{1.0472\cdot i}\cdot 4.$

Im Neugrad-Modus:

▸Polar Katalog >

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∠ θ) verwendet werden.

$(4\cdot i)$▸Polar $(4\angle\ 100.)$

Im Grad-Modus:

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

polyEval() (Polynom auswerten) Katalog >

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.

polyEval({1,2,3,4},2) 26

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

polyRoots() Katalog >

polyRoots(*Poly***,***Var***) ⇒*Liste***

polyRoots(*KoeffListe***) ⇒*Liste***

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: { }.

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·x+2·x+1

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

Hinweis: Siehe auch **cPolyRoots()**, Seite 34.

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

cPolyRoots(y^3+1,y)
{-1,0.5−0.866025·*i*,0.5+0.866025·*i*}

polyRoots(x^2+2·x+1,x) {-1,-1}

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

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

Berechnet die Potenzregressiony = (a · (x)^b)auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a ·(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 *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*

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

Prgm

Katalog >

Prgm
Block
EndPrgm

Vorlage zum Erstellen eines benutzerdefinierten Programms. Muss mit dem Befehl **Definiere (Define)**, **Definiere LibPub (Define LibPub)** oder **Definiere LibPriv (Define LibPriv)** verwendet werden.

Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen ":" voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein.

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

GCD berechnen und Zwischenergebnisse anzeigen.

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

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

prodSeq()

Siehe Π(), Seite 211.

Product (PI) (Produkt)

Siehe Π(), Seite 211.

product() (Produkt)

Katalog >

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

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

product() (Produkt) Katalog >

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.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

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

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.

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

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

propFrac() (Echter Bruch) Katalog >

Mit der Funktion **propFrac()** können Sie gemischte Brüche darstellen und die Addition und Subtraktion bei gemischten Brüchen demonstrieren.

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

Q

QR Katalog >

QR *Matrix*, *qMatrix*, *rMatrix*[, *Tol*]

Berechnet die Householdersche QR-Faktorisierung einer reellen oder komplexen Matrix. Die sich ergebenden Q- und R-Matrzen werden in den angegebenen *Matrix* gespeichert. Die Q-Matrix ist unitär. Bei der R-Matrix handelt es sich um eine obere Dreiecksmatrix.

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

- Wenn Sie [ctrl] [enter] verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
  5E⁻14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

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

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}$
QR *m1*,*qm*,*rm*	*Done*
qm	$\begin{bmatrix}0.123091&0.904534&0.408248\\0.492366&0.301511&-0.816497\\0.86164&-0.301511&0.408248\end{bmatrix}$
rm	$\begin{bmatrix}8.12404&9.60114&11.0782\\0.&0.904534&1.80907\\0.&0.&0.\end{bmatrix}$

QR

Katalog >

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

Katalog >

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

Berechnet die quadratische polynomiale Regressiony = a ·x2+b ·x+cauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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 240).

Ausgabevariable	Beschreibung

stat.RegEqn	Regressionsgleichung: a ·x^2+b ·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 *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf*, *Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

QuartReg

Katalog >

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

Berechnet die polynomiale Regression vierter Ordnungy = a ·x^4+b ·x^3+c · x^2+d ·x+eauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

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.

QuartReg Katalog >

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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

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 Paars (*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:

$R\blacktriangleright P\theta(2,2)$ 45.

Im Neugrad-Modus:

$R\blacktriangleright P\theta(2,2)$ 50.

Im Bogenmaß-Modus:

$R\blacktriangleright P\theta(3,2)$ 0.588003

$R\blacktriangleright P\theta\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}0. & 2.94771 & 0.643501\end{bmatrix}$

R▶Pr() Katalog >

R▶Pr (*xWert*, *yWert*) ⇒ *Wert*
R▶Pr (*xListe*, *yListe*) ⇒ *Liste*
R▶Pr (*xMatrix*, *yMatrix*) ⇒ *Matrix*

Gibt die äquivalente r-Koordinate des Paars *(x,y)* zurück.

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

Im Bogenmaß-Modus:

$R\blacktriangleright Pr(3,2)$	3.60555
$R\blacktriangleright Pr\left(\left[3\ \ -4\ \ 2\right],\left[0\ \ \frac{\pi}{4}\ \ 1.5\right]\right)$	$\left[3\ \ 4.07638\ \ \frac{5}{2}\right]$

▶Rad Katalog >

Wert1▶*Rad* ⇒ *Wert*

Wandelt das Argument ins Bogenmaß um.

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

Im Grad-Modus:

$(1.5)\blacktriangleright Rad$	$(0.02618)^r$

Im Neugrad-Modus:

$(1.5)\blacktriangleright Rad$	$(0.023562)^r$

rand() (Zufallszahl) Katalog >

rand() ⇒ *Ausdruck*
rand(*#Trials*) ⇒ *List*

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(*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.

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

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	*Done*
randMat(3,3)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

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

randNorm() (Zufallsnorm) Katalog >

randNorm(μ, σ**)** ⇒ *Ausdruck*
randNorm(μ, σ, *#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 [μ−3•σ, μ+3•σ].

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

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	⁻3.54356

randPoly() (Zufallspolynom) Katalog >

randPoly(*Var*, *Ordnung*)⇒ *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*

RandSeed 1147	*Done*
rand()	0.158206

RandSeed (Zufallszahl) Katalog >

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.

real() (Reell) Katalog >

real(*Value1***)** ⇒ *Wert*

Gibt den Realteil des Arguments zurück.

$\text{real}(2+3\cdot i)$ → 2

real(*List1***)** ⇒ *Liste*

Gibt für jedes Element den Realteil zurück.

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

real(*Matrix1***)** ⇒ *Matrix*

Gibt für jedes Element den Realteil zurück.

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

▶Rect Katalog >

Vektor ▶**Rect**

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

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.

$\left(\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix}\right)$▶Rect → $\begin{bmatrix} 1.06066 & 1.06066 & 2.59808 \end{bmatrix}$

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**

Im Bogenmaß-Modus:

▶Rect Katalog >

Zeigt *komplexerWert* in der kartesischen Form a+bi an. *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∠ θ) verwendet werden.

$$\left(4\cdot e^{\frac{\pi}{3}}\right)\blacktriangleright\text{Rect} \qquad 11.3986$$

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

Im Neugrad-Modus:

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

Im Grad-Modus:

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

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

ref() (Diagonalform) Katalog >

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 [ctrl] [enter] verwenden oder den Modus **Autom. oder Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
  5E−14 •max(dim(*Matrix1*)) •rowNorm (*Matrix1*)

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

ref() (Diagonalform) Katalog >

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:

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

Die Warnung erscheint, weil das verallgemeinerte Element $1/a$ für a=0 nicht zulässig wäre.

Sie können dieses Problem umgehen, indem Sie zuvor einen Wert in a speichern oder wie im folgenden Beispiel gezeigt eine Substitution mit dem womit-Operator „|“ vornehmen.

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

Hinweis: Siehe auch **rref()** page 149.

RefreshProbeVars Katalog >

RefreshProbeVars

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

StatusVar Value	Status
$statusVar$ =0	Normal (Programmausführung fortsetzen)
$statusVar$ =1	Die Applikation Vernier DataQuest™ befindet sich im Data Collection-Modus.

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
```

RefreshProbeVars Katalog >

StatusVar Value	Status
	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.

```
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) Katalog >

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:

remain(x,0) x
remain(x,y) x−y•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.

remain(7,0)	7
remain(7,3)	1
remain(⁻7,3)	⁻1
remain(7,⁻3)	1
remain(⁻7,⁻3)	⁻1
remain({12,⁻14,16},{9,7,⁻5})	{3,0,1}

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

Request Katalog >

Request *promptString*, *var*[, *FlagAnz*

Definieren Sie ein Programm:

[, *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.

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

```
Define request_demo()=Prgm
    Request “Radius: ”,r
    Disp “Fläche = “,pi*r²
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

```
request_demo()
```

Ergebnis nach Auswahl von **OK**:

```
Radius: 6/2
Fläche = 28.2743
```

Definieren Sie ein Programm:

```
Define polynomial()=Prgm
    Request "Polynom in x eingeben:",p
(x)
    Disp "Reelle Wurzeln:",polyRoots(p
(x),x)
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

```
polynomial()
```

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*

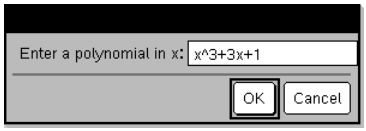
Anschließend kann das Programm die so definierte Funktion *Fkt*() nutzen. Die Meldung *EingabeString* sollte dem Benutzer die nötigen Informationen geben, damit dieser eine passende *Benutzerantwort* zur Vervollständigung der Funktionsdefinition eingeben kann.

Hinweis: Mit der Option Request Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen.

So halten Sie ein Programm an, das einen Befehl **Request** in einer Endlosschleife enthält:

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

Hinweis: Siehe auch **RequestStr**, page 142.



Ergebnis nach Eingabe von x^3+3x+1 und Auswahl von **OK**:

```
Reelle Wurzeln: {-0,322185}
```

RequestStr

Katalog >

RequestStr *promptString*, *var*[, *FlagAnz*]

Definieren Sie ein Programm:

RequestStr

Katalog >

Programmierbefehl: Verhält sich genauso wie die erste Syntax des Befehls **Request**, die Benutzerantwort wird jedoch immer als String interpretiert. Der Befehl **Request** interpretiert die Antwort hingegen als Ausdruck, es sei denn, der Benutzer setzt sie in Anführungszeichen ("").

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

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

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

Hinweis: Siehe auch **Request**, page 140.

```
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()
```

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

Katalog >

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.

```
Define factorial(nn)=
Func
Local answer,counter
1→answer
For counter,1,nn
answer·counter→answer
EndFor
Return answer
EndFunc
```

```
factorial(3)                    6
```

Return

Katalog >

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

right() (Rechts)

Katalog >

right(*Liste1* [, *Anz*]**)** ⇒ *Liste*

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

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

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

right(*Quellstring*[, *Anz*]**)** ⇒ *string*

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

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

```
right("Hello",2)                       "lo"
```

right(*Vergleich***)** ⇒ *Ausdruck*

Gibt die rechte Seite einer Gleichung oder Ungleichung zurück.

rk23 ()

Katalog >

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

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

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

Differentialgleichung:

y'=0.001*y*(100-y) und y(0)=10

```
rk23(0.001·y·(100−y),t,y,{0,100},10,1)
[ 0.     1.        2.        3.        4.
  10.    10.9367   11.9493   13.042    14.2 ▸
```

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

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 ▸
```

Verwendet die Runge-Kutta-Methode zum Lösen des Systems

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

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

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

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

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

Var ist die unabhängige Variable.

$ListeAbhVar$ ist eine Liste abhängiger Variablen.

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

$ListeAbhVar0$ ist eine Liste von Anfangswerten für abhängige Variablen.

Wenn $VarSchritt$ eine Zahl ungleich Null ergibt: Zeichen($VarSchritt$) = Zeichen ($VarMax$-$Var0$) und Lösungen werden an $Var0$+i*$VarSchritt$ für alle i=0,1,2,… zurückgegeben, sodass $Var0$+i*$VarSchritt$ in [$var0$,$VarMax$] ist (möglicherweise gibt es keinen Lösungswert an $VarMax$).

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 $y2(0)=5$

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

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

rk23 ()

Katalog >

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

diftol ist die Fehlertoleranz (standardmäßig 0.001).

root() (Wurzel)

Katalog >

root(*Wert***)** ⇒ *Wurzel*
root(*Wert1***, *Wert2***)** ⇒ *Wurzel*

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

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.

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

rotate() (Rotieren)

Katalog >

rotate(*Ganzzahl1***[,***#Rotationen***])**⇒ *Ganzzahl*

Rotiert die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ▶**Base2**, Seite 17.

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

Beispielsweise in einer Rechtsrotation:

Im Bin-Modus>

rotate(0b11111111111111111111111111111111)	
0b10000000000000000000000000000001▸	
rotate(256,1)	0b1000000000

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

Im Hex-Modus:

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

rotate() (Rotieren) Katalog >

Jedes Bit rotiert nach rechts.

0b00000000000000111101011000011010 1

Bit ganz rechts rotiert nach ganz links.

ergibt sich:

0b10000000000000011110101100001101 0

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

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

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

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

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

Im Dec-Modus:

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

rotate(*String1***[,***#Rotationen***]) ⇒** *String*

Gibt eine um *#Rotationen* Zeichen nach rechts oder links rotierte Kopie von *String1* zurück. Verändert *String1* 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)

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

round() (Runden) Katalog >

round(*Wert1***[, *Stellen*]) ⇒** *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(1.234567,3)	1.235

round() (Runden) Katalog >

round(*Liste1***[, *Stellen*]) ⇒ *Liste***

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

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

round(*Matrix1***[, *Stellen*]) ⇒ *Matrix***

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

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

rowAdd() (Zeilenaddition) Katalog >

rowAdd(*Matrix1***, *rIndex1*, *rIndex2*) ⇒ *Matrix***

Gibt eine Kopie von *Matrix1* zurück, in der die Zeile *rIndex2* durch die Summe der Zeilen *rIndex1* und *rIndex2* ersetzt ist.

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

rowDim() (Zeilendimension) Katalog >

rowDim(*Matrix***) ⇒ *Ausdruck***

Gibt die Anzahl der Zeilen von *Matrix* zurück.

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

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

$$\text{rowDim}(m1) \qquad 3$$

rowNorm() (Zeilennorm) Katalog >

rowNorm(*Matrix***) ⇒ *Ausdruck***

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

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

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

rowSwap() (Zeilentausch) Katalog >

rowSwap(*Matrix1***, *rIndex1*, *rIndex2*)** ⇒ *Matrix*

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

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

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

rref() (Reduzierte Diagonalform) Katalog >

rref(*Matrix1***[, *Tol*])** ⇒ *Matrix*

Gibt die reduzierte Diagonalform von *Matrix1* zurück.

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

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 [ctrl] [enter] verwenden oder den Modus **Autom. oder Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
  5E−14 •max(dim(*Matrix1*)) •rowNorm (*Matrix1*)

Hinweis: Siehe auch **ref()** page 138.

S

sec() (Sekans) [trig] Taste

sec(*Wert1***)** ⇒ *Wert*

Im Grad-Modus:

sec() (Sekans) trig Taste

sec(*Liste1***)** ⇒ *Liste*

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

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

sec(45)	1.41421
sec({1,2.3,4})	{1.00015,1.00081,1.00244}

$\sec^{-1}$() (Arkussekans) trig Taste

$\sec^{-1}$(*Wert1***)** ⇒ *Wert*

$\sec^{-1}$(*Liste1***)** ⇒ *Liste*

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

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

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

Im Grad-Modus:

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

sech() (Sekans hyperbolicus) Katalog >

sech(*Wert1***)** ⇒ *Wert*

sech(*Liste1***)** ⇒ *Liste*

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

sech(3)	0.099328
sech({1,2.3,4})	{0.648054,0.198522,0.036619}

sech^{-1}() (Arkussekans hyperbolicus) Katalog >

sech^{-1}(*Wert1***)** ⇒ *Wert*

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

sech⁻¹() (Arkussekans hyperbolicus) Katalog >

sech⁻¹ (*Liste1*) ⇒ *Liste*

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.

$\text{sech}^{-1}(1)$	0
$\text{sech}^{-1}(\{1,-2,2.1\})$	$\{0,2.0944\cdot i,8.E^{-}15+1.07448\cdot i\}$

Send 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 65), **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.

$\text{seq}(n^2,n,1,6)$ $\{1,4,9,16,25,36\}$

$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$ $\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ $\frac{1968329}{1270080}$

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie [ctrl] [enter].
Windows®: Drücken Sie **Strg+Eingabetaste**.
Macintosh®: Drücken ⌘+**Eingabetaste**.
iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie [≈] aus.

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

seqGen() Katalog >

seqGen(*Ausdr*, *Var*, *abhVar*, {*Var0*, *VarMax*}[, *ListeAnfTerme* [, *VarSchritt* [, *ObergrWert*]]]**)** ⇒*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 $u(n) = u(n\text{-}1)^2/2$ mit $u(1)$=**2** und *VarSchritt*=**1**.

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

Beispiel mit Var0=2:

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

System zweiter Folgen:

seqGen() Katalog >

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**.

$$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2(n-1)}{2}+u1(n-1)\right\},n,\{u1,u2\},\{1,5\},\left[\frac{_}{2}\right]\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Hinweis: Die Lücke (_) in der oben aufgeführten Anfangsterm-Matrix zeigt an, dass der Anfangsterm für u1(n) mit der expliziten Folge-Formel u1(n)=1/n berechnet wird.

seqn() Katalog >

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

Generiert eine Term-Liste für eine Folge *u*(*n*)=*Ausdr*(*u*, *n*) wie folgt: Erhöht *n* von 1 bis *nMax* um 1, wertet *u*(*n*) für die entsprechenden Werte von *n* mithilfe der Formel *Ausdr*(*u*, *n*) und *ListeAnfTerme* aus und gibt die Ergebnisse als Liste zurück.

seqn(*Ausdr***(***n* [**,** *nMax* [**,** *ObergrWert*]]**)⇒***Liste*

Generiert eine Term-Liste für eine nichtrekursive Folge *u*(*n*)=*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**.

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

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

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

setMode(*ModusNameGanzzahl***,** *GanzzahlFestlegen***)** ⇒*Ganzzahl*

setMode(*Liste***)** ⇒*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)** → *var* gespeichert haben, können Sie **setMode(***var***)** verwenden, um diese Einstellungen wiederherzustellen, bis die Funktion oder das Programm beendet wird. Siehe **getMode()**, Seite 71.

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 prog1()=Prgm                    Done
               Disp π
               setMode(1,16)
               Disp π
               EndPrgm
prog1()
                                    3.14159
                                    3.14
                                    Done
```

setMode

Katalog >

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)

Katalog >

shift(*Ganzzahl1* [**,***#Verschiebungen*]**)**⇒*Ganzzahl*

Im Bin-Modus:

```
shift(0b1111010110000110101)
                 0b1111010110000011010
shift(256,1)                0b1000000000
```

Im Hex-Modus:

Verschiebt die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ▶**Base2**, Seite 17.

Ist *#Verschiebungen* positiv, erfolgt die Verschiebung nach links. ist *#Verschiebungen* negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist ⁻1 (ein Bit nach rechts verschieben).

In einer Rechtsverschiebung wird das ganz rechts stehende Bit abgeschnitten und als ganz links stehendes Bit eine 0 oder 1 eingesetzt. Bei einer Linksverschiebung wird das Bit ganz links abgeschnitten und 0 als letztes Bit rechts eingesetzt.

Beispielsweise in einer Rechtsverschiebung:

Alle Bits werden nach rechts verschoben.

0b00000000000000111101011000011010

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

Es ergibt sich:

0b00000000000000011110101100001101 0

Das Ergebnis wird gemäß dem jeweiligen Basis-Modus angezeigt. Führende Nullen werden nicht angezeigt.

shift(0h78E)	0h3C7
shift(0h78E,⁻2)	0h1E3
shift(0h78E,2)	0h1E38

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

shift(*Liste1* [**,***#Verschiebungen*]**)⇒***Liste*

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

Im Dec-Modus:

shift({1,2,3,4})	{undef,1,2,3}
shift({1,2,3,4},⁻2)	{undef,undef,1,2}
shift({1,2,3,4},2)	{3,4,undef,undef}

shift() (Verschieben) Katalog >

Ist *#Verschiebungen* positiv, erfolgt die Verschiebung nach links. ist *#Verschiebungen* negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist ⁻1 (ein Element nach rechts verschieben).

Dadurch eingeführte neue Elemente am Anfang bzw. am Ende von *Liste* werden auf "undef" gesetzt.

shift(*String1* [**,***#Verschiebungen*]**)⇒***String*

Gibt eine um *#Verschiebungen* Zeichen nach rechts oder links verschobene Kopie von *Liste1* zurück. Verändert *String1* nicht.

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.

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

sign() (Zeichen) Katalog >

sign(*Wert1***)⇒***Wert*

sign(*Liste1***)⇒***Liste*

sign(*Matrix1***)⇒***Matrix*

Gibt für reelle und komplexe *Wert1* *Wert1* / **abs(***Wert1***)** zurück, wenn *Wert1* ≠ 0.

Gibt 1 zurück, wenn *Wert1* positiv ist.

Gibt ⁻1 zurück, wenn *Wert1* negativ ist.

sign(0) gibt ±1 zurück, wenn als Komplex-Formatmodus Reell eingestellt ist; anderenfalls gibt es sich selbst zurück.

sign(0) stellt im komplexen Bereich den Einheitskreis dar.

Gibt für jedes Element einer Liste bzw. Matrix das Vorzeichen zurück.

sign(-3.2)	-1
sign({2,3,4,-5})	{1,1,1,-1}

Bei Komplex-Formatmodus Reell:

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

simult(*KoeffMatrix***,** *KonstVektor***[,** *Tol***])⇒***Matrix*

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

Hinweis: Siehe auch **linSolve()**, Seite 91.

KoeffMatrix muss eine quadratische Matrix sein, die die Koeffizienten der Gleichung enthält.

KonstVektor muss die gleiche Zeilenanzahl (gleiche Dimension) besitzen wie *KoeffMatrix* und die Konstanten enthalten.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. 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 ·max(dim(*KoeffMatrix*))
  ·rowNorm(*KoeffMatrix*)

Auflösen nach x und y:

x + 2y = 1

3x + 4y = ⁻1

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

Die Lösung ist x=⁻3 und y=2.

Auflösen:

ax + by = 1

cx + dy = 2

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

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

simult(*KoeffMatrix***,** *KonstMatrix***[,** *Tol***])⇒***Matrix*

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

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.

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

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

sin() (Sinus) trig Taste

sin(*Wert1***)⇒***Wert*

sin(*Liste1***)⇒***Liste*

sin(*Wert1***)** gibt den Sinus des Arguments zurück.

sin(*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 °,G oder ʳ benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Im Grad-Modus:

$$\sin\left(\left(\frac{\pi}{4}\right)^{r}\right) \qquad 0.707107$$

$$\sin(45) \qquad 0.707107$$

$$\sin(\{0,60,90\}) \qquad \{0.,0.866025,1.\}$$

Im Neugrad-Modus:

$$\sin(50) \qquad 0.707107$$

Im Bogenmaß-Modus:

$$\sin\left(\frac{\pi}{4}\right) \qquad 0.707107$$

$$\sin(45°) \qquad 0.707107$$

sin(*Quadratmatrix1***)⇒***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 Bogenmaß-Modus:

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

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\-0.045492 & 0.949254 & -0.020274\\-0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

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

$\sin^{-1}$(*Wert1***)⇒***Wert*

Im Grad-Modus:

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

$\sin^{-1}$(*Liste1*)⇒*Liste*

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

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

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

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

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

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

$\sin^{-1}(\{0,0.2,0.5\})$ $\{0.,0.201358,0.523599\}$

$\sin^{-1}$(*Quadratmatrix1*)⇒*Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

$$\sin^{-1}\left(\begin{bmatrix}1 & 5\\ 4 & 2\end{bmatrix}\right)$$
$$\begin{bmatrix}-0.174533-0.12198\cdot i & 1.74533-2.35591\cdot i\\ 1.39626-1.88473\cdot i & 0.174533-0.593162\cdot i\end{bmatrix}$$

sinh() (Sinus hyperbolicus) Katalog >

sinh(*Wert1*)⇒*Wert*

sinh(*Liste1*)⇒*Liste*

sinh (*Wert1*) gibt den Sinus hyperbolicus des Arguments zurück.

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

$\sinh(1.2)$ 1.50946

$\sinh(\{0,1.2,3.\})$ $\{0,1.50946,10.0179\}$

sinh(*Quadratmatrix1*)⇒*Quadratmatrix*

Im Bogenmaß-Modus:

sinh() (Sinus hyperbolicus) Katalog >

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

$$\begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

$\sinh^{-1}$() (Arkussinus hyperbolicus) Katalog >

$\sinh^{-1}$(*Wert1***)⇒***Wert***

$\sinh^{-1}$(*Liste1***)⇒***Liste***

$\sinh^{-1}$(*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.

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,1.81845\}$

$\sinh^{-1}$ (*Quadratmatrix1***)⇒***Quadratmatrix***

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.

Im Bogenmaß-Modus:

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

$$\begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

SinReg Katalog >

SinReg *X*, *Y* [, [*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 165.)

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 240).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a ·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 *X-Liste*, die in der Regression mit den Beschränkungen für *Häuf, Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.YReg	Liste der Datenpunkte in der modifizierten *Y-Liste*, die schließlich in der Regression mit den Beschränkungen für *Häuf, Kategorieliste* und *Mit-Kategorien verwendet wurde*
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

SortA (In aufsteigender Reihenfolge sortieren)

Katalog >

SortA *Liste1*[, *Liste2*] [, *Liste3*] ...

SortA *Vektor1*[, *Vektor2*] [, *Vektor3*] ...

Sortiert die Elemente des ersten Arguments in aufsteigender Reihenfolge.

Bei Angabe von mehr als einem Argument werden die Elemente der zusätzlichen Argumente so sortiert, dass ihre neue Position mit der neuen Position der Elemente des ersten Arguments übereinstimmt.

Alle Argumente müssen Listen- oder Vektornamen sein. Alle Argumente müssen die gleiche Dimension besitzen.

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA *list1*	*Done*
list1	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA *list2*,*list1*	*Done*
list2	$\{1,2,3,4\}$
list1	$\{4,3,2,1\}$

SortD (In absteigender Reihenfolge sortieren)

Katalog >

SortD *Liste1*[, *Liste2*] [, *Liste3*] ...

SortD *Vektor1*[,*Vektor2*] [,*Vektor3*] ...

Identisch mit **SortA** mit dem Unterschied, dass **SortD** die Elemente in absteigender Reihenfolge sortiert.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

SortD (In absteigender Reihenfolge sortieren)

Katalog >

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

▸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 [ρ ∠θ ∠ϕ] an.

Vektor muss die Dimension 3 besitzen und kann ein Zeilen- oder ein Spaltenvektor sein.

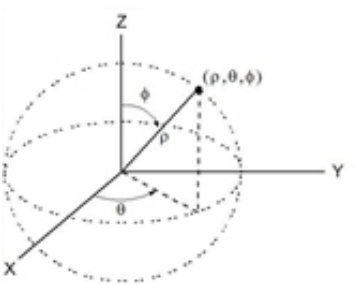
Hinweis: ▸Sphere ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen.

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie ctrl enter.
Windows®: Drücken Sie **Strg+Eingabetaste**.
Macintosh®: Drücken ⌘**+Eingabetaste**.
iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie ≈ aus.

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

$$\left(\left[2 \quad \angle\frac{\pi}{4} \quad 3\right]\right)\blacktriangleright\text{Sphere}$$
$$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$$



sqrt() (Quadratwurzel)

Katalog >

sqrt(*Wert1***)⇒***Wert*

sqrt(*Liste1***)⇒***Liste*

Gibt die Quadratwurzel des Arguments zurück.

$$\sqrt{4} \qquad 2$$
$$\sqrt{\{9,2,4\}} \qquad \{3,1.41421,2\}$$

sqrt() (Quadratwurzel) Katalog >

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 1.

stat.results Katalog >

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
"r²"	0.996109
"r"	0.998053
"Resid"	"{...}"
stat.values	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{⁻0.4,0.4,0.2,0.,⁻0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.x̄

stat.b9	stat.FBlock	Stat.$\hat{p}$	stat.Σx²	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}$Diff	stat.Σy²	stat.$\bar{x}$List
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat.$\bar{y}$
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat.$\hat{y}$
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat.$\hat{y}$List
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

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

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.

Siehe **stat.results**.

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:

stDevPop({1,2,5,⁻6,3,⁻2})	3.59398
stDevPop({1.3,2.5,⁻6.4},{3,2,5})	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 240).

stDevPop(*Matrix1***[,** *Häufigkeitsmatrix***])⇒***Matrix*

Ergibt einen Zeilenvektor der Populations-Standardabweichungen der Spalten in *Matrix1*.

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

Hinweis: *Matrix1* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

$$\text{stDevPop}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right)$$

$$\begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right)$$

$$\begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

stDevSamp() (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 240).

$$\text{stDevSamp}(\{1,2,5,-6,3,-2\}) \qquad 3.937$$

$$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.33345$$

stDevSamp() (Stichproben-Standardabweichung) Katalog >

stDevSamp(*Matrix1*[**,** *Häufigkeitsmatrix*]**)⇒***Matrix*

Ergibt einen Zeilenvektor der Stichproben-Standardabweichungen der Spalten in *Matrix1*.

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

Hinweis: *Matrix1* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

$$\text{stDevSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right)$$
$$\begin{bmatrix} 4. & 3.60555 & 2. \end{bmatrix}$$

$$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right)$$
$$\begin{bmatrix} 2.7005 & 5.44695 \end{bmatrix}$$

Stop (Stopp) Katalog >

Stop

Programmierbefehl: Beendet das Programm.

Stop ist in Funktionen nicht zulässig.

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

```
i:=0                                  0
Define prog1()=Prgm                   Done
               For i,1,10,1
               If i=5
               Stop
               EndFor
               EndPrgm
prog1()                               Done
i                                     5
```

Store (Speichern) Siehe → (speichern), Seite 220.

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(*Matrix1*[, *vonZei*] [, *vonSpl*] [, *bisZei*] [, *bisSpl*]**)** ⇒*Matrix*

Gibt die angegebene Untermatrix von *Matrix1* zurück.

Vorgaben: *vonZei*=1, *vonSpl*=1, *bisZei*=letzte Zeile, *bisSpl*=letzte Spalte.

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

$$\text{subMat}(m1,2,1,3,2) \qquad \begin{bmatrix}4&5\\7&8\end{bmatrix}$$

$$\text{subMat}(m1,2,2) \qquad \begin{bmatrix}5&6\\8&9\end{bmatrix}$$

Summe (Sigma) Siehe Σ(), Seite 211.

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 240).

$$\text{sum}(\{1,2,3,4,5\}) \qquad 15$$

$$\text{sum}(\{a,2\cdot a,3\cdot a\})$$

"Error: Variable is not defined"

$$\text{sum}(\text{seq}(n,n,1,10)) \qquad 55$$

$$\text{sum}(\{1,3,5,7,9\},3) \qquad 21$$

sum(*Matrix1*[, *Start*[, *Ende*]])⇒*Matrix*

Gibt einen Zeilenvektor zurück, der die Summen der Elemente aus den Spalten von *Matrix1* enthält.

Start und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *Matrix1* werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

$$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right) \qquad \begin{bmatrix}12&15&18\end{bmatrix}$$

$$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right) \qquad \begin{bmatrix}11&13&15\end{bmatrix}$$

sumIf(*Liste***,***Kriterien*[, *SummeListe*]**)⇒***Wert***

Gibt die kumulierte Summe aller Elemente in *Liste* zurück, die die angegebenen *Kriterien* erfüllen. Optional können Sie eine Alternativliste, *SummeListe*, angeben, an die die Elemente zum Kumulieren weitergegeben werden sollen.

Liste kann ein Ausdruck, eine Liste oder eine Matrix sein. *SummeListe* muss, sofern sie verwendet wird, dieselben Dimension(en) haben wie *Liste*.

Kriterien können sein:

- Ein Wert, ein Ausdruck oder eine Zeichenfolge. So kumuliert beispielsweise **34** nur solche Elemente in *Liste*, die vereinfacht den Wert 34 ergeben.
- Ein Boolescher Ausdruck, der das Sonderzeichen **?** als Platzhalter für jedes Element verwendet. Beispielsweise zählt **?<10** nur solche Elemente in *Liste* zusammen, die kleiner als 10 sind.

Wenn ein Element in *Liste* die *Kriterien* erfüllt, wird das Element zur Kumulationssumme hinzugerechnet. Wenn Sie *SummeListe* hinzufügen, wird stattdessen das entsprechende Element aus *SummeListe* zur Summe hinzugerechnet.

In der Lists & Spreadsheet Applikation können Sie anstelle von *Liste* und *SummeListe* auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

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

```
sumIf({1,2,e,3,π,4,5,6},2.5<?<4.5)
                          12.859874482
sumIf({1,2,3,4},2<?<5,{10,20,30,40})
                                   70
```

sumSeq() Siehe Σ(), Seite 211.

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 >

*Matrix1*T⇒*matrix*

Gibt die komplex konjugierte, transponierte Matrix von *Matrix1* zurück.

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

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

tan() (Tangens) trig Taste

tan(*Wert1***)**⇒*Wert*

tan(*Liste1***)**⇒*Liste*

tan(*Wert1***)** gibt den Tangens des Arguments zurück.

tan(*Liste1***)** gibt in Form einer Liste für jedes Element in *Liste1* den Tangens zurück.

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Im Grad-Modus:

$$\tan\left(\left(\frac{\pi}{4}\right)^r\right) \qquad 1.$$

$$\tan(45) \qquad 1.$$

$$\tan(\{0,60,90\}) \qquad \{0.,1.73205,\text{undef}\}$$

Im Neugrad-Modus:

$$\tan\left(\left(\frac{\pi}{4}\right)^r\right) \qquad 1.$$

$$\tan(50) \qquad 1.$$

$$\tan(\{0,50,100\}) \qquad \{0.,1.,\text{undef}\}$$

Im Bogenmaß-Modus:

tan() (Tangens)

trig **Taste**

$\tan\left(\frac{\pi}{4}\right)$ 1.

$\tan(45°)$ 1.

$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right)$ $\{0.,1.73205,0.,1.\}$

tan(*Quadratmatrix1***)⇒***Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

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

$$\begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

$\tan^{-1}$() (Arkustangens)

trig **Taste**

$\tan^{-1}$(*Wert1***)⇒***Wert***

$\tan^{-1}$(*Liste1***)⇒***Liste***

$\tan^{-1}$(*Wert1***)** gibt den Winkel zurück, dessen Tangens *Wert1* ist.

$\tan^{-1}$(*Liste1***)** gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Tangens zurück.

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

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

$\tan^{-1}$(*Quadratmatrix1***)⇒***Quadratmatrix***

Im Grad-Modus:

$\tan^{-1}(1)$ 45

Im Neugrad-Modus:

$\tan^{-1}(1)$ 50

Im Bogenmaß-Modus:

$\tan^{-1}(\{0,0.2,0.5\})$ $\{0,0.197396,0.463648\}$

Im Bogenmaß-Modus:

tan⁻¹() (Arkustangens) — trig 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()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

$$\begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

tanh() (Tangens hyperbolicus) — Katalog >

tanh(*Wert1***)⇒***Wert*

tanh(*Liste1***)⇒***Liste*

tanh(*Wert1***)** gibt den Tangens hyperbolicus des Arguments zurück.

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

$$\tanh(1.2) \qquad 0.833655$$

$$\tanh(\{0,1\}) \qquad \{0.,0.761594\}$$

tanh(*Quadratmatrix1***)⇒***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()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

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

$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

tanh⁻¹() (Arkustangens hyperbolicus) — Katalog >

tanh⁻¹(*Wert1***)⇒***Wert*

tanh⁻¹(*Liste1***)⇒***Liste*

tanh⁻¹(*Wert1***)** gibt den inversen Tangens hyperbolicus des Arguments zurück.

Im Komplex-Formatmodus “kartesisch”:

$$\tanh^{-1}(0) \qquad 0.$$

$$\tanh^{-1}(\{1,2.1,3\})$$

$$\{\text{undef},0.518046-1.5708\cdot i,0.346574-1.570$$

tanh⁻¹() (Arkustangens hyperbolicus) Katalog >

tanh⁻¹(*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.

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

tanh⁻¹(*Quadratmatrix1***)⇒***Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

$$\begin{bmatrix} -0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.9473 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901 \end{bmatrix}$$

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

tCdf() Katalog >

tCdf (*UntGrenze*,*ObGrenze*,*FreiGrad***)⇒***Zahl*, wenn *UntGrenze* und *ObGrenze* Zahlen sind, *Liste*, wenn *UntGrenze* und *ObGrenze* Listen sind

Berechnet für eine Student-*t*-Verteilung mit vorgegebenen Freiheitsgraden *FreiGrad* die Intervallwahrscheinlichkeit zwischen *UntGrenze* und *ObGrenze*.

Für P(X ≤ *obereGrenze*) setzen Sie *untereGrenze* = −9**E**999.

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.

Text Katalog >

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 140, oder**RequestStr**, Seite 142.

Hinweis: Sie können diesen Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

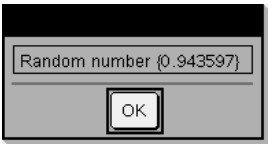
Schließen Sie in der Vorlage Prgm...EndPrgm jede Zeile mit ↵ ab anstatt mit enter. Auf der Computertastatur halten Sie **Alt** gedrückt und drücken die **Eingabetaste**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:=”Random number “ &
string(rand(i))
    Text strinfo
    EndFor
EndPrgm
```

Starten Sie das Programm:

```
text_demo()
```

Muster eines Dialogfelds:



Then Siehe If, Seite 76.

tInterval Katalog >

tInterval *Liste*[,*Häuf*[,*KNiv*]]

(Datenlisteneingabe)

tInterval $\bar{x}$,*sx*,*n*[,*KNiv*]

(Zusammenfassende statistische Eingabe)

tInterval

Katalog >

Berechnet das Konfidenzintervall *t*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 165.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 240).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat.σx	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert

tInterval_2Samp (Zwei-Stichproben-t-Konfidenzintervall)

Katalog >

tInterval_2Samp *Liste1*,*Liste2*[,*Häufigkeit1*[,*Häufigkeit2*[,*KStufe*[,*Verteilt*]]]]

(Datenlisteneingabe)

tInterval_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*KStufe*[,*Verteilt*]]

(Zusammenfassende statistische Eingabe)

Berechnet ein *t*-Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 165.)

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 240).

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.σx1, stat.σx2	Stichproben-Standardabweichungen für *Liste 1* und *Liste 2*
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn *Verteilt* = 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.

$$\text{trace}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right) \qquad 15$$

$$a:=12 \qquad 12$$

$$\text{trace}\left(\begin{bmatrix}a&0\\1&a\end{bmatrix}\right) \qquad 24$$

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 250).

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.

```
Define prog1()=Prgm
    Try
    z:=z+1
    Disp "z incremented."
    Else
    Disp "Sorry, z undefined."
    EndTry
    EndPrgm
                                    Done
z:=1:prog1()
                                z incremented.
                                    Done
DelVar z:prog1()
                              Sorry, z undefined.
                                    Done
```

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 123.

```
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"
```

Try (Versuche) Katalog >

```
    ClrErr
  Else
    PassErr
  EndIf
EndTry
EndPrgm
```

tTest Katalog >

tTest $\mu 0$,*Liste*[,*Häufigkeit*[,*Hypoth*]]

(Datenlisteneingabe)

tTest $\mu 0$,$\bar{x}$,*sx*,*n*,[*Hypoth*]

(Zusammenfassende statistische Eingabe)

Führt einen Hypothesen-Test für einen einzelnen, unbekannten Populationsmittelwert μ durch, wenn die Populations-Standardabweichung σ unbekannt ist. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 165.)

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 240).

Ausgabevariable	Beschreibung
stat.t	$(\bar{x} - \mu 0)$ / (stdev / 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 *Liste*
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 165.)

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 240).

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 *Liste 1* und *Liste 2*
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*
stat.n1, stat.n2	Stichprobenumfang
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn *Verteilt*=1.

tvmFV() Katalog >

tvmFV(*N***,***I***,***PV***,***Pmt***,[***PpY***],[***CpY***],[***PmtAt***])⇒***Wert***

Finanzfunktion, die den Geld-Endwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 182) beschrieben. Siehe auch **amortTbl()**, Seite 7.

```
tvmFV(120,5,0,-500,12,12)        77641.1
```

tvmI() Katalog >

tvmI(*N***,***PV***,***Pmt***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])⇒***Wert***

Finanzfunktion, die den jährlichen Zinssatz berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 182) beschrieben. Siehe auch **amortTbl()**, Seite 7.

```
tvmI(240,100000,-1000,0,12,12)        10.5241
```

tvmN() Katalog >

tvmN(*I***,***PV***,***Pmt***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])⇒***Wert***

Finanzfunktion, die die Anzahl der Zahlungsperioden berechnet.

```
tvmN(5,0,-500,77641,12,12)        120.
```

tvmN() Katalog >

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 182) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPmt() Katalog >

tvmPmt(*N***,***I***,***PV***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])⇒***Wert***

Finanzfunktion, die den Betrag der einzelnen Zahlungen berechnet.

```
tvmPmt(60,4,30000,0,12,12)        -552.496
```

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 182) beschrieben. Siehe auch **amortTbl()**, Seite 7.

tvmPV() Katalog >

tvmPV(*N***,***I***,***Pmt***,***FV***,[***PpY***],[***CpY***],[***PmtAt***])⇒***Wert***

Finanzfunktion, die den Barwert berechnet.

```
tvmPV(48,4,-500,30000,12,12)      -3426.7
```

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 182) 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 165.)

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 240).

Ausgabevariable	Beschreibung
stat.$\overline{x}$	Mittelwert der x-Werte
stat. x	Summe der x-Werte
stat. x2	Summe der x2-Werte
stat.sx	Stichproben-Standardabweichung von x
stat. x	Populations-Standardabweichung von x
stat.n	Anzahl der Datenpunkte
stat.$\overline{y}$	Mittelwert der y-Werte
stat. y	Summe der y-Werte
stat. y^2	Summe der y2-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_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.MinY	Minimum der y-Werte
stat.Q_1Y	1. Quartil von y
stat.MedY	Median von y
stat.Q_3Y	3. Quartil von y
stat.MaxY	Maximum der y-Werte
stat. $(x-)^2$	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert
stat. $(y-)^2$	Summe der Quadrate der Abweichungen der y-Werte vom Mittelwert

U

unitV() (Einheitsvektor) Katalog >

unitV(*Vektor1***)⇒***Vektor*

Gibt je nach der Form von *Vektor1* entweder einen Zeilen- oder einen Spalteneinheitsvektor zurück.

Vektor1 muss eine einzeilige oder eine einspaltige Matrix sein.

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

unitV([1 2 1])	[0.408248 0.816497 0.408248]
unitV([1; 2; 3])	[0.267261; 0.534522; 0.801784]

unLock Katalog >

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

unLock*Var***.**

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

Siehe **Lock**, Seite 95, und **getLockInfo()**, Seite 71.

a:=65	65
Lock *a*	*Done*
getLockInfo(*a*)	1
a:=75	"Error: Variable is locked."
DelVar *a*	"Error: Variable is locked."
Unlock *a*	*Done*
a:=75	75
DelVar *a*	*Done*

V

varPop() (Populationsvarianz) Katalog >

varPop(*Liste* [**,***Häufigkeitsliste*]**)⇒***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

varPop() (Populationsvarianz) Katalog >

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 240).

varSamp() (Stichproben-Varianz) Katalog >

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 240).

$$\text{varSamp}(\{1,2,5,^{-}6,3,^{-}2\}) \qquad \frac{31}{2}$$

$$\text{varSamp}(\{1,3,5\},\{4,6,2\}) \qquad \frac{68}{33}$$

varSamp(*Matrix1***[,** *Häufigkeitsmatrix***])⇒***Matrix*

Gibt einen Zeilenvektor zurück, der die Stichproben-Varianz jeder Spalte von *Matrix1* enthält.

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

Wenn ein Element in einer der Matrizen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Matrix wird ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 240).

Hinweis: *Matrix1* muss mindestens zwei Zeilen enthalten.

$$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right) \qquad \begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$$

$$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix},\begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right) \qquad \begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$$

Wait

Katalog >

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

Hinweis: 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}
```

warnCodes () Katalog >

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 259).

when() (Wenn) Katalog >

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.

$\text{when}(x<0,x+3)\|x=5$	undef

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.

$\text{when}(n>0,n\cdot factoral(n-1),1)\rightarrow factoral(n)$	*Done*
$factoral(3)$	6
$3!$	6

While

Katalog >

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→i
                        0→tempsum
                        While i≤n
                        tempsum+1/i→tempsum
                        i+1→i
                        EndWhile
                        Return tempsum
                        EndFunc
                                          Done
sum_of_recip(3)                           11/6
```

X

xor (Boolesches exklusives oder)

Katalog >

BoolescherAusd1 **xor** *BoolescherAusdr2*
ergibt *Boolescher Ausdruck*

BoolescheListe1 **xor** *BoolescheListe2*
ergibt *Boolesche Liste*

BoolescheMatrix1 **xor** *BoolescheMatrix2*
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.

```
true xor true                    false
5>3 xor 3>5                      true
```

Ganzzahl1 **xor** *Ganzzahl2* ⇒ *Ganzzahl*

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O

```
0h7AC36 xor 0h3D5F               0h79169
```

xor (Boolesches exklusives oder) Katalog >

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) Katalog >

zInterval σ,*Liste*[,*Häufigkeit*[,*KStufe*]]

(Datenlisteneingabe)

zInterval σ,$\bar{x}$,*n* [,*KStufe*]

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 165.)

zInterval (z-Konfidenzintervall) Katalog >

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.sx	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert
stat.σ	Bekannte Populations-Standardabweichung für Datenfolge *Liste*

zInterval_1Prop (z-Konfidenzintervall für eine Proportion) Katalog >

zInterval_1Prop *x*,*n* [,*KStufe*]

Berechnet ein *z*-Konfidenzinterval für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 165.)

x ist eine nicht negative Ganzzahl.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat.$\hat{p}$	Die berechnete Erfolgsproportion
stat.ME	Fehlertoleranz
stat.n	Anzahl der Stichproben in Datenfolge

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen) Katalog >

zInterval_2Prop *x1*,*n1*,*x2*,*n2*[,*KStufe*]

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 165.)

x1 und *x2* sind nicht negative Ganzzahlen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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 σ_1,σ_2 ,*Liste1*,*Liste2* [,*Häufigkeit1*[,*Häufigkeit2*,[*KStufe*]]]

(Datenlisteneingabe)

zInterval_2Samp σ_1,σ_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 165.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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.σx1, stat.σx2	Stichproben-Standardabweichungen für *Liste 1* und *Liste 2*
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.r1, stat.r2	Bekannte Populations-Standardabweichungen für Datenfolge *Liste 1* und *Liste 2*

zTest

Katalog >

zTest μ*0*,σ,*Liste*,[*Häufigkeit*[,*Hypoth*]]

(Datenlisteneingabe)

zTest μ*0*,σ,$\bar{x}$,*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 165.)

Getestet wird H_0: μ = μ0 in Bezug auf eine der folgenden Alternativen:

Für H_a: μ < μ0 setzen Sie *Hypoth*<0

Für H_a: μ ≠ μ0 (Standard) setzen Sie *Hypoth*=0

Für H_a: μ > μ0 setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

Ausgabevariable	Beschreibung
stat.z	($\bar{x}$ − μ0) / (σ / sqrt(n))

Ausgabevariable	Beschreibung
stat.P Value	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge in *Liste*
stat.sx	Stichproben-Standardabweichung der Datenfolge. Wird nur für *Daten*eingabe zurückgegeben.
stat.n	Stichprobenumfang

zTest_1Prop (z-Test für eine Proportion) Katalog >

zTest_1Prop *p0*,*x*,*n*[,*Hypoth*]

Berechnet einen *z*-Test für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 165.)

x ist eine nicht negative Ganzzahl.

Getestet wird H_0: $p = p0$ in Bezug auf eine der folgenden Alternativen:

Für H_a: $p > p0$ setzen Sie *Hypoth*>0

Für H_a: $p \neq p0$ *(Standard)* setzen Sie *Hypoth*=0

Für H_a: $p < p0$ setzen Sie *Hypoth*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 240).

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 165.)

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 240).

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 σ_1,σ_2 ,*Liste1*,*Liste2* [,*Häufigkeit1*[,*Häufigkeit2*[,*Hypoth*]]]

(Datenlisteneingabe)

zTest_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*Hypoth*]

(Zusammenfassende statistische Eingabe)

zTest_2Samp (z-Test für zwei Stichproben) Katalog >

Berechnet einen *z*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 165.)

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 240).

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 *Liste 1* und *Liste 2*
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*
stat.n1, stat.n2	Stichprobenumfang

Sonderzeichen

+ (addieren) ⊞ Taste

Wert1 + *Wert2*⇒*Wert*

Gibt die Summe der beiden Argumente zurück.

56	56
56+4	60
60+4	64
64+4	68
68+4	72

+ (addieren) [+] Taste

Liste1 + *Liste2*⇒*Liste*

Matrix1 + *Matrix2*⇒*Matrix*

Gibt eine Liste (bzw. eine Matrix) zurück, die die Summen der entsprechenden Elemente von *Liste1* und *Liste2* (oder *Matrix1* und *Matrix2*) enthält.

Die Argumente müssen die gleiche Dimension besitzen.

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\{22,3.14159,1.5708\}$
$\left\{10,5,\frac{\pi}{2}\right\}\rightarrow l2$	$\{10,5,1.5708\}$
$l1+l2$	$\{32,8.14159,3.14159\}$

Wert + *Liste1*⇒*Liste*

Liste1 + *Wert*⇒*Liste*

Gibt eine Liste zurück, die die Summen von *Wert* plus jedem Element der *Liste1* enthält.

$15+\{10,15,20\}$	$\{25,30,35\}$
$\{10,15,20\}+15$	$\{25,30,35\}$

Wert + *Matrix1*⇒*Matrix*

Matrix1 + *Wert*⇒*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.

$20+\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\3 & 24\end{bmatrix}$

−(subtrahieren) [−] Taste

Wert1 − *Wert2*⇒*Wert*

Gibt *Wert1* minus *Wert2* zurück.

$6-2$	4
$\pi-\frac{\pi}{6}$	2.61799

Liste1 − *Liste2*⇒*Liste*

Matrix1 − *Matrix2*⇒*Matrix*

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.

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,{}^{-}1.85841,0.\}$
$\begin{bmatrix}3 & 4\end{bmatrix}-\begin{bmatrix}1 & 2\end{bmatrix}$	$\begin{bmatrix}2 & 2\end{bmatrix}$

−(subtrahieren) — [−] Taste

Wert − Liste1⇒Liste

Liste1 − Wert⇒Liste

Subtrahiert jedes Element der *Liste1* von *Wert* oder subtrahiert *Wert* von jedem Element der *Liste1* und gibt eine Liste der Ergebnisse zurück.

$15-\{10,15,20\}$ → $\{5,0,-5\}$

$\{10,15,20\}-15$ → $\{-5,0,5\}$

Wert − Matrix1⇒Matrix

Matrix1 − Wert⇒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.

$20-\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → $\begin{bmatrix}19 & -2\\ -3 & 16\end{bmatrix}$

Hinweis: Verwenden Sie **.−** (Punkt Minus) zum Subtrahieren eines Ausdrucks von jedem Element.

·(multiplizieren) — [×] Taste

Wert1 •Wert2⇒Wert

Gibt das Produkt der beiden Argumente zurück.

$2\cdot 3.45$ → 6.9

Liste1•Liste2⇒Liste

Gibt eine Liste zurück, die die Produkte der entsprechenden Elemente aus *Liste1* und *Liste2* enthält.

Die Listen müssen die gleiche Dimension besitzen.

$\{1.,2,3\}\cdot\{4,5,6\}$ → $\{4,10,18\}$

Matrix1•Matrix2⇒Matrix

Gibt das Matrizenprodukt von *Matrix1* und *Matrix2* zurück.

Die Spaltenanzahl von *Matrix1* muss gleich die Zeilenanzahl von *Matrix2* sein.

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\cdot\begin{bmatrix}7 & 8\\ 7 & 8\\ 7 & 8\end{bmatrix}$ → $\begin{bmatrix}42 & 48\\ 105 & 120\end{bmatrix}$

·(multiplizieren) — [×] Taste

Wert•Liste1⇒Liste

Liste1•Wert⇒Liste

Gibt eine Liste zurück, die die Produkte von *Wert* und jedem Element der *Liste1* enthält.

$\pi\cdot\{4,5,6\}$ → $\{12.5664,15.708,18.8496\}$

Wert•Matrix1⇒Matrix

Matrix1•Wert⇒Matrix

Gibt eine Matrix zurück, die die Produkte von *Wert* und jedem Element der *Matrix1* enthält.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\cdot 0.01$ → $\begin{bmatrix}0.01 & 0.02\\0.03 & 0.04\end{bmatrix}$

$6\cdot\text{identity}(3)$ → $\begin{bmatrix}6 & 0 & 0\\0 & 6 & 0\\0 & 0 & 6\end{bmatrix}$

Hinweis: Verwenden Sie **.** ·(Punkt-Multiplikation) zum Multiplizieren eines Ausdrucks mit jedem Element.

/(dividieren) — [÷] Taste

Wert1 / Wert2⇒Wert

Gibt *Wert1* dividiert durch *Wert2* zurück.

Hinweis: Siehe auch **Vorlage Bruch**, Seite 1.

$\frac{2}{3.45}$ → 0.57971

Liste1 / Liste2⇒Liste

Gibt eine Liste der Elemente von *Liste1* dividiert durch *Liste2* zurück.

Die Listen müssen die gleiche Dimension besitzen.

$\frac{\{1.,2,3\}}{\{4,5,6\}}$ → $\left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$

Wert / Liste1 ⇒ Liste

Liste1 / Wert ⇒ Liste

Gibt eine Liste der Elemente von *Wert* dividiert durch *Liste1* oder*Liste1* dividiert durch *Wert* zurück.

$\frac{6}{\{3,6,\sqrt{6}\}}$ → $\{2,1,2.44949\}$

$\frac{\{7,9,2\}}{7\cdot 9\cdot 2}$ → $\left\{\frac{1}{18},\frac{1}{14},\frac{1}{63}\right\}$

Wert / *Matrix1⇒ Matrix*

Matrix1 / Wert ⇒ Matrix

Gibt eine Matrix zurück, die die Quotienten *Matrix1 / Wert* enthält.

$\frac{[7\ \ 9\ \ 2]}{7\cdot 9\cdot 2}$ → $\begin{bmatrix}\frac{1}{18} & \frac{1}{14} & \frac{1}{63}\end{bmatrix}$

/ (dividieren) ÷ Taste

Hinweis: Verwenden Sie **. /** (Punkt-Division) zum Dividieren eines Ausdrucks durch jedes Element.

^ (Potenz) ^ Taste

Wert1 **^** *Wert2* ⇒ *Wert*

Liste1 **^** *Liste2* ⇒ *Liste*

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.

4^2 → 16

$\{2,4,6\}^{\{1,2,3\}}$ → $\{2,16,216\}$

Wert **^** *Liste1* ⇒ *Liste*

Gibt *Wert* hoch den Elementen von *Liste1* zurück.

$\pi^{\{1,2,-3\}}$ → $\{3.14159,9.8696,0.032252\}$

Liste1 **^** *Wert* ⇒ *Liste*

Gibt die Elemente von *Liste1* hoch *Wert* zurück.

$\{1,2,3,4\}^{-2}$ → $\left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$

Quadratmatrix1 **^** *Ganzzahl* ⇒ *Matrix*

Gibt *Quadratmatrix1* hoch *Ganzzahl* zurück.

Quadratmatrix1 muss eine quadratische Matrix sein.

Ist *Ganzzahl* = ⁻1, wird die inverse Matrix berechnet.

Ist *Ganzzahl* < ⁻1, wird die inverse Matrix hoch der entsprechenden positiven Zahl berechnet.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^2$ → $\begin{bmatrix}7 & 10\\15 & 22\end{bmatrix}$

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-1}$ → $\begin{bmatrix}-2 & 1\\\frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-2}$ → $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\\frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x² (Quadrat)

x² 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 Matriz-Quadrat von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Quadrats jedes einzelnen Elements. Verwenden Sie .^2, um das Quadrat jedes einzelnen Elements zu berechnen.

4^2 → 16

$\{2,4,6\}^2$ → $\{4,16,36\}$

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2$ → $\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}.\hat{}\ 2$ → $\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$

.+ (Punkt-Addition)

. **+** Tasten

Matrix1 **.+** *Matrix2* ⇒ *Matrix*

Wert **.+** *Matrix1* ⇒ *Matrix*

Matrix1 **.+** *Matrix2* gibt eine Matrix zurück, die Summe jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Wert **.+** *Matrix1* gibt eine Matrix zurück, die die Summe von Zahl und jedem Element von *Matrix1* ist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$ → $\begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$

$5 .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$ → $\begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$

.− (Punkt-Subt.)

. **−** Tasten

Matrix1 **.−** *Matrix2* ⇒ *Matrix*

Wert **.−***Matrix1* ⇒ *Matrix*

Matrix1 **.−***Matrix2* gibt eine Matrix zurück, die die Differenz jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Wert **.−***Matrix1* gibt eine Matrix zurück, die die Differenz von *Wert* und jedem Element von *Matrix1* ist.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .- \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$ → $\begin{bmatrix} -9 & -18 \\ -27 & -36 \end{bmatrix}$

$5 .- \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$ → $\begin{bmatrix} -5 & -15 \\ -25 & -35 \end{bmatrix}$

.· (Punkt-Mult.)

[.] [×] Tasten

Matrix1 **.·** *Matrix2* ⇒ *Matrix*

Wert **.·***Matrix1* ⇒ *Matrix*

Matrix1 **.·** *Matrix2* gibt eine Matrix zurück, die das Produkt jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Wert **.·** *Matrix1* Gibt eine Matrix zurück, die die Produkte von *Wert* und jedem Element der *Matrix1* enthält.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} .\cdot \begin{bmatrix}10 & 20\\30 & 40\end{bmatrix} \qquad \begin{bmatrix}10 & 40\\90 & 160\end{bmatrix}$$

$$5 .\cdot \begin{bmatrix}10 & 20\\30 & 40\end{bmatrix} \qquad \begin{bmatrix}50 & 100\\150 & 200\end{bmatrix}$$

. / (Punkt-Division)

[.] [÷] Tasten

Matrix1 **. /** *Matrix2* ⇒ *Matrix*

Wert **. /** *Matrix1* ⇒ *Matrix*

Matrix1 **. /** *Matrix2* gibt eine Matrix zurück, die der Quotient jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Wert **. /** *Matrix1* gibt eine Matrix zurück, die der Quotient von *Wert* und jedem Element von *Matrix1* ist.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} ./ \left(\begin{bmatrix}10 & 20\\30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{10} & \frac{1}{10}\\\frac{1}{10} & \frac{1}{10}\end{bmatrix}$$

$$5 ./ \left(\begin{bmatrix}10 & 20\\30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{2} & \frac{1}{4}\\\frac{1}{6} & \frac{1}{8}\end{bmatrix}$$

.^ (Punkt-Potenz)

[.] [^] Tasten

Matrix1 **.^** *Matrix2* ⇒ *Matrix*

Wert **.^** *Matrix1* ⇒ *Matrix*

Matrix1 **.^** *Matrix2* gibt eine Matrix zurück, in der jedes Element aus *Matrix2* Exponent des entsprechenden Elements aus *Matrix1* ist.

Wert **.^** *Matrix1* gibt eine Matrix zurück, in der jedes Element aus *Matrix1* Exponent von *Wert* ist.

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} .^\wedge \begin{bmatrix}0 & 2\\3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 4\\27 & \frac{1}{4}\end{bmatrix}$$

$$5 .^\wedge \begin{bmatrix}0 & 2\\3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 25\\125 & \frac{1}{5}\end{bmatrix}$$

-(Negation) (−) Taste

-Wert1 ⇒ *Wert*

-Liste1 ⇒ *Liste*

-Matrix1 ⇒ *Matrix*

Gibt die Negation des Arguments zurück.

Bei einer Liste oder Matrix werden alle Elemente negiert zurückgegeben.

Ist das Argument eine binäre oder hexadezimale ganze Zahl, ergibt die Negation das Zweierkomplement.

```
-2.43                                  -2.43
-{-1,0.4,1.2E19}                {1.,-0.4,-1.2E19}
```

`Im Bin-Modus:`

Wichtig: Null, nicht Buchstabe O

```
-0b100101
0b11111111111111111111111111111111▸
```

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

% (Prozent) ctrl 📖 Tasten

Wert1 **%** ⇒ *Wert*

Liste1 **%** ⇒ *Liste*

Matrix1 **%** ⇒ *Matrix*

Ergibt $\frac{argument}{100}$

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 ctrl enter.
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*⇒*Boolescher Ausdruck*

Liste1 **=** *Liste2*⇒ *Boolesche Liste*

Matrix1 **=** *Matrix2*⇒ *Boolesche Matrix*

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung gleich *Ausdr2* ist.

Beispielfunktion mit den mathematischen Vergleichssymbolen: =, ≠, <, ≤, >, ≥

= (gleich)

[=] Taste

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung ungleich *Ausdr2* ist.

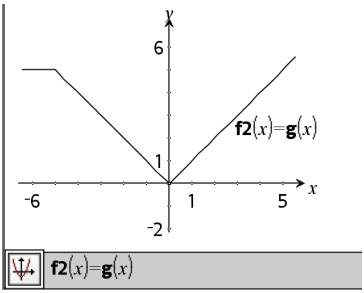
In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

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≤-5 Then
        Return 5
        ElseIf x>-5 and x<0 Then
        Return -x
        ElseIf x≥0 and x≠10 Then
        Return x
        ElseIf x=10 Then
        Return 3
        EndIf
        EndFunc
                                Done
```

Ergebnis der graphischen Darstellung g(x)



≠ (ungleich)

[ctrl] [=] Tasten

Ausdr1 ≠ Ausdr2 ⇒ Boolescher Ausdruck

Liste1 ≠ Liste2 ⇒ Boolesche Liste

Matrix1 ≠ Matrix2 ⇒ 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.

Siehe Beispiel bei “=” (gleich).

≠ (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* ⇒ *Boolescher Ausdruck*

Liste1 **<** *Liste2* ⇒ *Boolesche Liste*

Matrix1 **<** *Matrix2* ⇒ *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.

Siehe Beispiel bei “=” (gleich).

≤ (kleiner oder gleich)

ctrl = Tasten

Ausdr1 ≤ *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 ≤ *Liste2* ⇒ *Boolesche Liste*

Matrix1 ≤ *Matrix2* ⇒ *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.

Siehe Beispiel bei “=” (gleich).

≤ (kleiner 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 **<=**

> (größer als)

ctrl = Tasten

Ausdr1 **>** *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 **>** *Liste2* ⇒ *Boolesche Liste*

Matrix1 **>** *Matrix2* ⇒ *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.

Siehe Beispiel bei “=” (gleich).

≥ (größer oder gleich)

ctrl = Tasten

Ausdr1 ≥ *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 ≥ *Liste2* ⇒ *Boolesche Liste*

Matrix1 ≥ *Matrix2* ⇒ *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.

Siehe Beispiel bei “=” (gleich).

≥ (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

BoolescherAusd1 ⇒ *BoolescherAusdr2* ergibt *Boolescher Ausdruck*

BoolescheListe1 ⇒ *BoolescheListe2* ergibt *Boolesche Liste*

BoolescheMatrix1 ⇒ *BoolescheMatrix2* ergibt *Boolesche Matrix*

Ganzzahl1 ⇒ *Ganzzahl2* ergibt *Ganzzahl*

5>3 or 3>5	true
5>3 ⇒ 3>5	false
3 or 4	7
3 ⇒ 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} ⇒ {3,2,1}	{-1,-1,-3}

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)

ctrl = Tasten

BoolescherAusdr1 ⇔ *BoolescherAusdr2* ergibt *Boolescher Ausdruck*

BoolescheListe1 ⇔ *BoolescheLiset2* ergibt *Boolesche Liste*

BoolescheMatrix1 ⇔ *BoolescheMatrix2* ergibt *Boolesche Matrix*

Ganzzahl1 ⇔ *Ganzzahl2* ergibt *Ganzzahl*

5>3 xor 3>5	true
5>3 ⇔ 3>5	false
3 xor 4	7
3 ⇔ 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} ⇔ {3,2,1}	{-3,-1,-3}

Gibt die Negation einer **XOR** boleschen Operation auf beiden Argumenten zurück. Gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie **<=>** drücken

! (Fakultät)

?!▸ Taste

Wert1 **!** ⇒ *Wert*

Liste1 **!** ⇒ *Liste*

Matrix1 **!** ⇒ *Matrix*

5!	120
{{5,4,3}}!	{120,24,6}
$\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right)!$	$\begin{bmatrix}1 & 2\\6 & 24\end{bmatrix}$

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)

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

d(*Ausdr1***,** *Var***[,** *Ordnung***])⇒***Wert*

d(*Liste1***,** *Var***[,** *Ordnung***])⇒***Liste*

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.

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

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

$$x:=3:\frac{d}{dx}\left(\left\{x^2,x^3,x^4\right\}\right) \qquad \{6,27,108\}$$

$$\frac{d}{dx}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}}\right)|x=0 \qquad \text{undef}$$

$$\text{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)|x=0 \qquad 0.000033$$

d() (Ableitung) Katalog >

Hierzu rechts ein Beispiel. Die erste Ableitung von x·(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 >

∫(*Ausdr1***,** *Var***,** *Untere***,** *Obere***)** ⇒ *Wert*

$$\int_0^1 x^2\,dx \qquad 0.333333$$

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.

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) ctrl x² Tasten

√ (*Wert1***)**⇒*Wert*

√ (*Liste1***)**⇒*Liste*

$$\sqrt{4} \qquad 2$$

$$\sqrt{\{9,2,4\}} \qquad \{3,1.41421,2\}$$

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

√() (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.

Π() (ProdSeq) Katalog >

Π(*Ausdr1*, *Var*, *Von*, *Bis*)⇒*Ausdruck*

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

Wertet *Ausdr1* für jeden Wert von *Var* zwischen *Von* und *Bis* aus und gibt das Produkt der Ergebnisse zurück.

Hinweis: Siehe auch **Vorlage Produkt (Π)**, Seite 5.

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

$$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right) \qquad \left\{\frac{1}{120},120,32\right\}$$

Π(*Ausdr1*, *Var*, *Von*, *Von*−1)⇒1

Π(*Ausdr1*, *Var*, *Von*, *Bis*) ⇒1/Π (*Ausdr1*, *Var*, *Bis*+*1*, *Von*−*1*) if *Bis* < *Von*−*1*

$$\prod_{k=4}^{3}(k) \qquad 1$$

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.

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right) \qquad 6$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right) \qquad \frac{1}{4}$$

Σ() (SumSeq) Katalog >

Σ(*Ausdr1*, *Var*, *Von*, *Bis*)⇒*Ausdruck*

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

$$\sum_{n=1}^{5}\left(\frac{1}{n}\right) \qquad \frac{137}{60}$$

Σ() (SumSeq) Katalog >

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.

Σ(*Ausdr1***,** *Var***,** *Von***,** *Von*−*1***)⇒**0

Σ(*Ausdr1***,** *Var***,** *Von***,** *Bis***) ⇒**⁻**Σ(***Ausdr1***,** *Var***,** *Bis*+*1***,** *Von*−1**)** if *Bis* < *Von*−*1*

$$\sum_{k=4}^{3}(k) \qquad 0$$

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.

$$\sum_{k=4}^{1}(k) \qquad -5$$

$$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k) \qquad 4$$

ΣInt() Katalog >

ΣInt(*NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,[***Pmt***],** [*FV***],** [*PpY***],** [*CpY***],** [*PmtAt***],** [*WertRunden***])⇒***Wert*

ΣInt (*NPmt1***,***NPmt2***,***AmortTabelle***)⇒***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 182) 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.

ΣInt(1,3,12,4.75,20000,,,12,12) -218.11

tbl:=amortTbl(12,12,4.75,20000,,,12,12)

0	0.	0.	20000.
1	-79.17	-1630.69	18369.3
2	-72.71	-1637.15	16732.2
3	-66.23	-1643.63	15088.5
4	-59.73	-1650.13	13438.4
5	-53.19	-1656.67	11781.7
6	-46.64	-1663.22	10118.5
7	-40.05	-1669.81	8448.7
8	-33.44	-1676.42	6772.28
9	-26.81	-1683.05	5089.23
10	-20.14	-1689.72	3399.51
11	-13.46	-1696.4	1703.11
12	-6.74	-1703.12	-0.01

ΣInt(1,3,*tbl*) -218.11

ΣInt() Katalog >

- 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.

ΣInt(*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 ΣPrn() auf dieser und **Bal()**, Seite 16.

ΣPrn() Katalog >

ΣPrn(*NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,** [*Pmt*]**,** [*FV*]**,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**,** [*WertRunden*]**)⇒***Wert*

ΣPrn (*NPmt1***,***NPmt2***,***AmortTabelle***)⇒***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 182) 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.

ΣPrn(1,3,12,4.75,20000,,,12,12) -4911.47

tbl:=amortTbl(12,12,4.75,20000,,,12,12)

0	0.	0.	20000.
1	-79.17	-1630.69	18369.3
2	-72.71	-1637.15	16732.2
3	-66.23	-1643.63	15088.5
4	-59.73	-1650.13	13438.4
5	-53.19	-1656.67	11781.7
6	-46.64	-1663.22	10118.5
7	-40.05	-1669.81	8448.7
8	-33.44	-1676.42	6772.28
9	-26.81	-1683.05	5089.23
10	-20.14	-1689.72	3399.51
11	-13.46	-1696.4	1703.11
12	-6.74	-1703.12	-0.01

ΣPrn(1,3,*tbl*) -4911.47

ΣPrn() Katalog >

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

ΣPrn(*NPmt1,NPmt2,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 ΣInt() auf dieser und **Bal()**, Seite 16.

(Umleitung) ctrl Tasten

*varNameString*

Greift auf die Variable namens *VarNameString* zu. So können Sie innerhalb einer Funktion Variablen unter Verwendung von Strings erzeugen.

$xyz:=12$	12
$\#("x"\&"y"\&"z")$	12

Erzeugt oder greift auf die Variable xyz zu.

$10\rightarrow r$	10
$"r"\rightarrow s1$	"r"
$\#s1$	10

Gibt den Wert der Variable (r) zurück, dessen Name in Variable s1 gespeichert ist.

E (Wissenschaftliche Schreibweise) EE Taste

*Mantisse***E***Exponent*

Gibt eine Zahl in wissenschaftlicher Schreibweise ein. Die Zahl wird als *Mantisse* × $10^{Exponent}$ interpretiert.

Tipp: Wenn Sie eine Potenz von 10 eingeben möchten, ohne ein Dezimalwertergebnis zu verursachen, verwenden Sie 10^*Ganzzahl*.

23000.	23000.
2300000000.+4.1E15	4.1E15
$3\cdot 10^4$	30000

E (Wissenschaftliche Schreibweise) EE 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) 1 Taste

*Ausdr1*g⇒*Ausdruck*

*Ausdr1*g⇒*Ausdruck*

*Liste1*g⇒*Liste*

*Matrix1*g⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Bogenmaß-Modus einen Winkel in Neugrad anzugeben.

Im Winkelmodus Bogenmaß wird *Ausdr1* mit π/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.

Im Grad-, Neugrad- oder Bogenmaß-Modus:

$\cos(50^g)$	0.707107
$\cos(\{0,100^g,200^g\})$	$\{1,0.,-1.\}$

r (Bogenmaß) 1 Taste

*Wert1*r⇒*Wert*

*Liste1*r⇒*Liste*

*Matrix1*r⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Neugrad-Modus einen Winkel im Bogenmaß anzugeben.

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

$\cos\left(\frac{\pi}{4^r}\right)$	0.707107
$\cos\left(\left\{0^r,\left(\frac{\pi}{12}\right)^r,-(\pi)^r\right\}\right)$	$\{1,0.965926,-1.\}$

ʳ (Bogenmaß) 1 Taste

Im Winkelmodus Grad wird das Argument mit 180/π multipliziert.

Im Winkelmodus Bogenmaß wird das Argument unverändert zurückgegeben.

Im Neugrad-Modus wird das Argument mit 200/π multipliziert.

Tipp: Verwenden Sie ʳ in einer Funktionsdefinition, wenn Sie bei Ausführung der Funktion das Bogenmaß frei von der Winkelmoduseinstellung erzwingen möchten.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie `@r` eintippen.

° (Grad) 1 Taste

Wert1°⇒Wert

Liste1°⇒Liste

Matrix1°⇒Matrix

Diese Funktion gibt Ihnen die Möglichkeit, im Neugrad- oder Bogenmaß-Modus einen Winkel in Grad anzugeben.

Im Winkelmodus Bogenmaß wird das Argument mit π/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:

$\cos(45°)$ 0.707107

Im Winkelmodus Bogenmaß:

$\cos\left(\left\{0,\frac{\pi}{4},90°,30.12°\right\}\right)$

$\{1.,0.707107,0.,0.864976\}$

°, ', '' (Grad/Minute/Sekunde)

ctrl Tasten

dd°*mm*'*ss.ss*''⇒*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°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

∠ (Winkel)

ctrl Tasten

[*Radius*,∠θ_*Winkel*]⇒*Vektor*

(Eingabe polar)

[*Radius*,∠θ_*Winkel*,*Z_Koordinate*]⇒*Vektor*

(Eingabe zylindrisch)

[*Radius*,∠θ_*Winkel*,∠θ_*Winkel*]⇒*Vektor*

(Eingabe sphärisch)

Gibt Koordinaten als Vektor zurück, wobei die aktuelle Einstellung für Vektorformat gilt: kartesisch, zylindrisch oder sphärisch.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie **@<** eintippen.

(*Größe* ∠ *Winkel***)**⇒*komplexerWert*

Im Bogenmaß-Modus mit Vektorformat eingestellt auf:

kartesisch

$[5 \ \angle 60° \ \angle 45°]$

$[1.76777 \quad 3.06186 \quad 3.53553]$

zylindrisch

$[5 \ \angle 60° \ \angle 45°]$

$[3.53553 \quad \angle 1.0472 \quad 3.53553]$

sphärisch

$[5 \ \angle 60° \ \angle 45°]$

$[5. \quad \angle 1.0472 \quad \angle 0.785398]$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

∠ (Winkel) ctrl Tasten

(Eingabe polar)

Dient zur Eingabe eines komplexen Werts in polarer (r∠θ) Form. Der *Winkel* wird gemäß der aktuellen Winkelmoduseinstellung interpretiert.

$5+3\cdot i-\left(10\angle\frac{\pi}{4}\right)$ $\quad -2.07107-4.07107\cdot i$

$5+3\cdot i-\left(10\angle\frac{\pi}{4}\right)$ $\quad -2.07107-4.07107\cdot i$

_ (Unterstrich als leeres Element)

Siehe “Leere (ungültige) Elemente” , Seite 240.

10^() Katalog >

10^(*Wert1***)⇒***Wert*

10^(*Liste1***)⇒***Liste*

Gibt 10 hoch Argument zurück.

Bei einer Liste wird 10 hoch jedem Element von *Liste1* zurückgegeben.

$10^{1.5}$ $\quad 31.6228$

10^(*Quadratmatrix1***)⇒***Quadratmatrix*

Ergibt 10 hoch *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von 10 hoch jedem Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$$10^{\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix}}\quad \begin{bmatrix}1.14336\text{E}7&8.17155\text{E}6&6.67589\text{E}6\\9.95651\text{E}6&7.11587\text{E}6&5.81342\text{E}6\\7.65298\text{E}6&5.46952\text{E}6&4.46845\text{E}6\end{bmatrix}$$

^⁻¹(Kehrwert) Katalog >

Wert1 **^⁻¹⇒***Wert*

Liste1 **^⁻¹⇒***Liste*

Gibt den Kehrwert des Arguments zurück.

Bei einer Liste wird für jedes Element von *Liste1* der Kehrwert zurückgegeben.

$(3.1)^{-1}$ $\quad 0.322581$

^⁻¹(Kehrwert) Katalog >

Quadratmatrix1 **^⁻¹**⇒*Quadratmatrix*

Gibt die Inverse von *Qudratmatrix1* zurück.

Quadratmatrix1 muss eine nicht-singuläre quadratische Matrix sein.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

| (womit-Operator) ctrl Tasten

Ausdr **|** *BoolescherAusdr1* [**and***BoolescherAusdr2*]...

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.

$x+1|x=3 \qquad 4$

$x+55|x=\sin(55) \qquad 54.0002$

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.

$x^3-2\cdot x+7\rightarrow f(x) \qquad Done$

$f(x)|x=\sqrt{3} \qquad 8.73205$

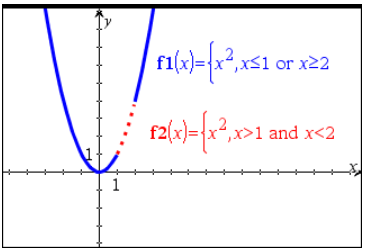
Intervallbeschränkungen werden in Form einer oder mehrerer mit logischen „**and**“ oder „**or**“ Operatoren verknüpfte Ungleichungen angegeben. Intervallbeschränkungen ermöglichen auch Vereinfachungen, die andernfalls ungültig oder nicht berechenbar wären.

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x) \qquad 0.$

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x)|x>0 \text{ and } x<5 \qquad 3.$

| (womit-Operator)

ctrl Tasten



Ausschließungen verwenden den relationalen Operator „ungleich“ (/= oder ≠), um einen bestimmten Wert bei der Operation auszuschließen.

→ (speichern)

ctrl var Taste

Wert → *Var*

Liste → *Var*

Matrix → *Var*

Expr → *Funktion*(*Param1*,...)

List → *Funktion*(*Param1*,...)

Matrix → *Funktion*(*Param1*,...)

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 =: myvar` ein.

$\frac{\pi}{4} \rightarrow myvar$	0.785398
$2\cdot\cos(x) \rightarrow y1(x)$	*Done*
$\{1,2,3,4\} \rightarrow lst5$	$\{1,2,3,4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → *str1*	"Hello"

:= (zuweisen)

ctrl [=] Tasten

Var := *Wert*

Var := *Liste*

Var := *Matrix*

Function(*Param1*,...) := *Ausdr*

Function(*Param1*,...) := *Liste*

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.

$myvar:=\frac{\pi}{4}$	.785398
$y1(x):=2\cdot\cos(x)$	*Done*
$lst5:=\{1,2,3,4\}$	$\{1,2,3,4\}$
$matg:=\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$	$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$
str1:="Hello"	"Hello"

© (Kommentar)

ctrl [📖] 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)=Func
        © Declare variables
        Local i,result
        result:=0
        For i,1,n,1 ©Loop n times
        result:=result+i^2
        EndFor
        Return result
        EndFunc
                                Done
g(3)                            14
```

0b, 0h

0 B Tasten, 0 H Tasten

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

Im Dec-Modus:

0b10+0hF+10	27

0b, 0h

0 B Tasten, 0 H Tasten

Kennzeichnet eine Dual- bzw. Hexadezimalzahl. Zur Eingabe einer Dual- oder Hexadezimalzahl muss unabhängig vom jeweiligen Basis-Modus das Präfix 0b bzw. 0h verwendet werden. Eine Zahl ohne Präfix wird als dezimal behandelt (Basis 10).

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

Im Bin-Modus:

0b10+0hF+10	0b11011

Im Hex-Modus:

0b10+0hF+10	0h1B

TI-Nspire™ CX II – Zeichenbefehle

Das vorliegende Dokument ergänzt das TI-Nspire™ Referenzhandbuch und das TI-Nspire™ CAS Referenzhandbuch. Alle TI-Nspire™ CX II Befehle werden in Version 5.1 des TI-Nspire™ Referenzhandbuchs und des TI-Nspire™ CAS Referenzhandbuchs ergänzt und mit ihnen veröffentlicht.

Grafikprogrammierung

In den TI-Nspire™ CX II Handhelds und TI-Nspire™ Desktop-Applikationen wurden für die Grafikprogrammierung Befehle hinzugefügt.

Die TI-Nspire™ CX II Handhelds wechseln in diesen Grafikmodus, wenn Grafikbefehle ausgeführt werden und wechseln nach Beendigung des Programms in den Kontext zurück, in dem das Programm ausgeführt wurde.

Auf dem Bildschirm wird bei Ausführung des Programms in der oberen Leiste „Wird ausgeführt…“ angezeigt. Bei Beendigung des Programms wird „Beendet” angezeigt. Durch Drücken einer beliebigen Taste verlässt das System den Grafikmodus.

- Der Wechsel zum Grafikmodus wird automatisch ausgelöst, wenn bei Ausführung des TI-Basic-Programms einer der Zeichenbefehle (Grafikbefehle) erkannt wird.
- Dieser Wechsel findet nur dann statt, wenn ein Programm in Calculator ausgeführt wird bzw. in Scratchpad in einem Dokument oder Taschenrechner.
- Der Wechsel vom Grafikmodus weg wird bei Programmbeendigung ausgeführt.
- Der Grafikmodus ist nur in der TI-Nspire™ CX II Handheld- und Desktop-TI-Nspire™ CX II Handheld-Ansicht verfügbar. Das bedeutet, dass dieser in der PC-Dokumentenansicht 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).

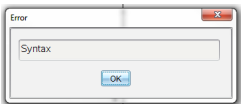
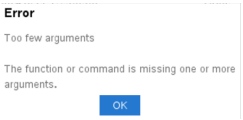
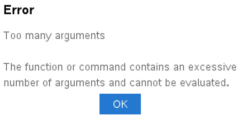
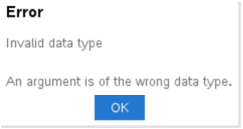
Grafikbildschirm	**Standard**
Höhe	212
Breite	318
Farbe	Weiß: 255,255,255

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)

Katalog > CXII

Clear *x, y, Breite, Höhe*

Löscht den gesamten Bildschirm, wenn keine Parameter angegeben wurden.

Werden *x, y, Breite* und *Höhe* angegeben, wird das durch die Parameter definierte Rechteck gelöscht.

```
Löschen
```

Löscht den gesamten Bildschirm

```
Clear 10,10,100,50
```

Löscht eine Rechtecksfläche mit der oberen linken Ecke in (10, 10), einer Breite 100 und einer Höhe 50

DrawArc

Katalog > CXII

DrawArc *x, y, Breite, Höhe, startAngle, arcAngle*

Zeichnet einen Bogen innerhalb eines definierten begrenzenden Rechtecks mit dem angegebenen Start- und Bogenwinkel.

x, y: obere linke Koordinate des begrenzenden Rechtecks

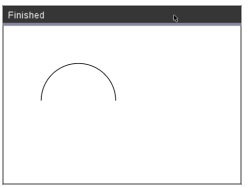
Breite, Höhe: Abmessungen des begrenzenden Rechtecks

Der „Bogenwinkel“ definiert die Ausbiegung des Bogens.

Diese Parameter können als Ausdrücke bereitgestellt werden, die eine Zahl ergeben, die dann auf die nächste Ganzzahl gerundet wird.

```
DrawArc 20,20,100,100,0,90
```

```
DrawArc 50,50,100,100,0,180
```



Siehe auch: FillArc

DrawCircle

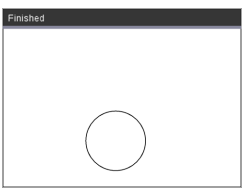
Katalog > CXII

DrawCircle *x, y, Radius*

x, y: Koordinate des Mittelpunkts

Radius: Radius des Kreises

```
DrawCircle 150,150,40
```



Siehe auch: FillCircle

DrawLine

Katalog > CXII

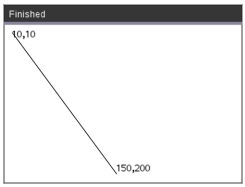
DrawLine *x1, y1, x2, y2*

Zeichnet eine Linie von *x1, y1, x2, y2* aus.

Ausdrücke, die eine Zahl ergeben, die dann auf die nächste Ganzzahl gerundet wird.

Bildschirmgrenzen: Wenn aufgrund der angegebenen Koordinaten ein Teil der Zeile außerhalb des Grafikbildschirms gezeichnet wird, dann wird dieser Teil der Linie abgeschnitten und keine Fehlermeldung angezeigt.

```
DrawLine 10,10,150,200
```



DrawPoly

Katalog > CXII

Es gibt zwei Varianten der Befehle:

DrawPoly *xlist, ylist*

oder

DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*

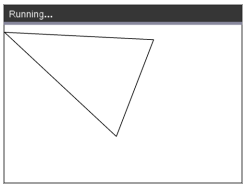
Hinweis: DrawPoly *xlist, ylist*
Form (Shape) verbindet *x1, y1* mit *x2, y2, x2, y2* mit *x3, y3* usw.

Hinweis: DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*
xn, yn wird **NICHT** automatisch mit *x1, y1* verbunden.

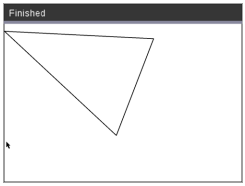
Ausdrücke, die eine Liste von realen Float-Variablen ergeben
xlist, ylist

Ausdrücke, die eine reale einzelne Float-Variable ergeben
x1, y1...xn, yn = Koordinaten für Polygoneckpunkte

```
xlist:={0,200,150,0}
ylist:={10,20,150,10}
DrawPoly xlist,ylist
```




```
DrawPoly 0,10,200,20,150,150,0,10
```



DrawPoly

Katalog > CXII

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

Katalog > CXII

DrawRect *x, y, Breite, Höhe*

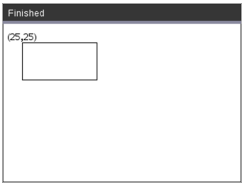
x, y: Obere linke Koordinate des Rechtecks

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

Katalog > CXII

DrawText *x, y, exprOrString1 [,exprOrString2]...*

x, y: Koordinaten der Textausgabe

Zeichnet den Text in *exprOrString* an der angegebenen *x*--*y*-Koordinatenposition.

Die Regeln für *exprOrString* sind die gleichen wie für **Disp** – **DrawText** kann mehrere Argumente akzeptieren.

```
DrawText 50,50,"Hallo Welt"
```

FillArc

Katalog > CXII

FillArc *x, y, Breite, Höhe startAngle, arcAngle*

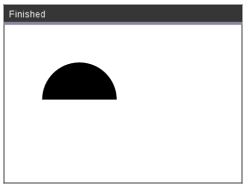
x, y: obere linke Koordinate des begrenzenden Rechtecks

Innerhalb des definierten begrenzenden Rechtecks mit den angegeben Start- und Bogenwinkeln einen Bogen zeichnen und füllen.

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem SetColor-Befehl eingestellt werden.

Der „Bogenwinkel“ definiert die Ausbiegung des Bogens.

```
FillArc 50,50,100,100,0,180
```



FillCircle

Katalog > CXII

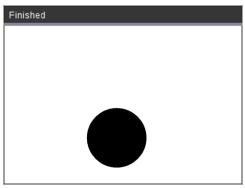
FillCircle *x, y, Radius*

x, y: Koordinate des Mittelpunkts

Einen Kreis mit angegebenen Mittelpunkt und Radius zeichnen und füllen.

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem SetColor-Befehl eingestellt werden.

```
FillCircle 150,150,40
```




```
Hier!
```

FillPoly

Katalog > CXII

FillPoly *xlist, ylist*

oder

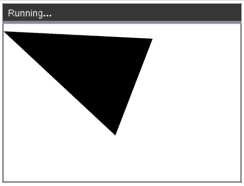
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

Hinweis: Linie und Farbe werden durch SetColor und SetPen festgelegt.

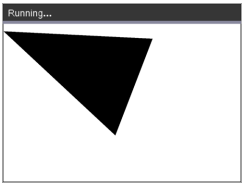
```
xlist:={0,200,150,0}
ylist:={10,20,150,10}
FillPoly xlist,ylist
```

FillPoly

Katalog > CXII




```
FillPoly 0,10,200,20,150,150,0,10
```



FillRect

Katalog > CXII

FillRect *x, y, Breite, Höhe*

x, y: Obere linke Koordinate des Rechtecks

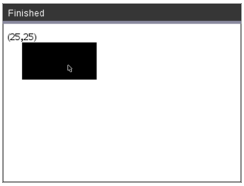
Breite, Höhe: Breite und Höhe des Rechtecks

An der durch (*x*,*y*) angegebenen Koordinate mit der oberen linken Ecke ein Rechteck zeichnen und füllen

Die Standardfüllfarbe ist Schwarz. Die Füllfarbe kann mit dem SetColor-Befehl eingestellt werden.

Hinweis: Linie und Farbe werden durch SetColor und SetPen festgelegt.

```
FillRect 25,25,100,50
```



getPlatform()

Katalog > CXII

getPlatform()

```
getPlatform()                    "dt"
```

Ergibt:
„dt“ auf Desktop-Softwareanwendungen
„hh“ auf TI-Nspire™ CX Handhelds
„ios“ auf TI-Nspire™ CX App für iPad®

PaintBuffer

Katalog > CXII

PaintBuffer

Farbengrafik-Puffer zum Bildschirm

Dieser Befehl wird in Verbindung mit UseBuffer verwendet, um die Geschwindigkeit der Darstellung auf dem Bildschirm zu erhöhen, wenn das Programm mehrere Grafikobjekte erzeugt.

```
UseBuffer
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
Radius:=randInt(10,50)
Wait 0,5
DrawCircle x,y,Radius
EndFor
PaintBuffer
```

Dieses Programm zeigt alle 10 Kreise gleichzeitig an.

Wird der Befehl „UseBuffer“ entfernt, wird jeder Kreis so angezeigt, wie er gezeichnet wird.

Siehe auch: UseBuffer

PlotXY *x, y, Form*

x, y: Koordinate zur Plot-Form

Form: eine Zahl zwischen 1 und 13, die die Form festlegt

1 – Gefüllter Kreis

2 – Leerer Kreis

3 – Gefülltes Quadrat

4 – Leeres Quadrat

5 – Kreuz

6 – Plus

7 – Dünn

8 – Mittelgroßer Punkt, ausgefüllt

9 – Mittelgroßer Punkt, unausgefüllt

10 – Großer Punkt, ausgefüllt

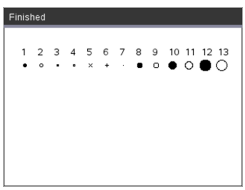
11 – Großer Punkt, unausgefüllt

12 – Größter Punkt, ausgefüllt

13 – Größter Punkt, unausgefüllt

```
PlotXY 100,100,1
```

```
For n,1,13
DrawText 1+22*n,40,n
PlotXY 5+22*n,50,n
EndFor
```



SetColor

Katalog > CXII

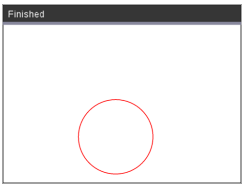
SetColor

Rot-Wert, Grün-Wert, Blau-Wert

Gültige Werte für Rot, Grün und Blau liegen zwischen 0 und 255.

Legt die Farbe für nachfolgende Draw-Befehle fest

```
SetColor 255,0,0
DrawCircle 150,150,100
```



SetPen

Katalog > CXII

SetPen

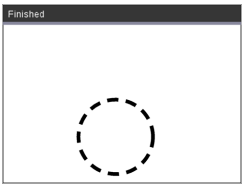
Dicke, Stil

Dicke: 1 <= 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.

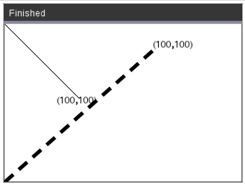
Befindet sich der Teil des gezeichneten Objekts außerhalb des Fensters, wird die Ausgabe zugeschnitten (nicht angezeigt) und keine Fehlermeldung angezeigt.

```
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 xmin größer oder gleich xmax oder ymin größer oder gleich ymax, wird eine Fehlermeldung angezeigt.



Objekte, die vor einem SetWindow-Befehl gezeichnet wurden, werden mit der neuen Konfiguration nicht neu gezeichnet.

Verwenden Sie zum Zurücksetzen der Fensterparameter auf die Standardeinstellungen:

SetWindow 0,0,0,0

UseBuffer

Katalog > CXII

UseBuffer

Zeichnet Grafik-Buffer außerhalb des Bildschirms anstatt auf den Bildschirm (zur Leistungssteigerung)

Dieser Befehl wird in Verbindung mit PaintBuffer verwendet, um die Geschwindigkeit der Darstellung auf dem Bildschirm zu erhöhen, wenn das Programm mehrere Grafikobjekte erzeugt.

Mit UseBuffer werden alle Grafiken erst nach Ausführung des nächsten PaintBuffer-Befehls angezeigt.

UseBuffer muss lediglich einmal im Programm aufgerufen werden, d. h. nicht bei jeder Verwendung von PaintBuffer ist ein entsprechender UseBuffer erforderlich.

```
UseBuffer
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
Radius:=randInt(10,50)
Wait 0,5
DrawCircle x,y,Radius
EndFor
PaintBuffer
```

Dieses Programm zeigt alle 10 Kreise gleichzeitig an.

Wird der Befehl „UseBuffer“ entfernt, wird jeder Kreis so angezeigt, wie er gezeichnet wird.

Siehe auch: PaintBuffer

Leere (ungültige) Elemente

Bei der Analyse von Daten der realen Welt liegt möglicherweise nicht immer ein vollständiger Datensatz vor. TI-Nspire™ lässt leere bzw. ungültige Datenelemente zu, sodass Sie mit den nahezu vollständigen Daten fortfahren können anstatt von vorn anfangen oder unvollständige Fälle verwerfen zu müssen.

Ein Beispiel für Daten mit leeren Elementen finden Sie im Kapitel Lists & Spreadsheet unter "*Tabellendaten grafisch darstellen*".

Mit der Funktion **delVoid()** können Sie leere Elemente aus einer Liste löschen. Die Funktion **isVoid()** sucht nach leeren Elementen. Einzelheiten finden Sie unter **delVoid()**, Seite 43, und **isVoid()**, Seite 83.

Hinweis: Um ein leeres Element manuell in einen mathematischen Ausdruck einzugeben, geben Sie "_" oder das Schlüsselwort **`void`** ein. Das Schlüsselwort **`void`** wird bei der Auswertung des Ausdrucks automatisch in das Symbol "_" konvertiert. Um "_" auf dem Handheld einzugeben, drücken Sie ctrl ␣.

Kalkulationen mit ungültigen Elementen

Bei der Mehrzahl aller Kalkulationen, die ein ungültiges Element enthalten, wird das Ergebnis ebenfalls ungültig sein. Sonderfälle sind nachstehend aufgeführt.

$\lvert_\rvert$	_
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

sum({2,_,3,5,6.6})	16.6
median({1,2,_,_,_,3})	2
cumulativeSum({1,2,_,4,5})	{1,3,_,7,12}
cumulativeSum([1 2; 3 _; 5 6])	[1 2; 4 _; 9 8]

Listenargumente, die ungültige Elemente enthalten

SortA und **SortD** verschieben alle ungültigen Elemente im ersten Argument nach unten.

{5,4,3,_,1} → *list1*	{5,4,3,_,1}
{5,4,3,2,1} → *list2*	{5,4,3,2,1}
SortA *list1*,*list2*	*Done*
list1	{1,3,4,5,_}
list2	{1,3,4,5,2}

{1,2,3,_,5} → *list1*	{1,2,3,_,5}
{1,2,3,4,5} → *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.}

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.}

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.}

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	**pi**
θ	**theta**
∞	**infinity**
$\leq$	**<=**
$\geq$	**>=**
$\neq$	**/=**
$\Rightarrow$ (logische Implikation)	**=>**
$\Leftrightarrow$ (logische doppelte Implikation, XNOR)	**<=>**
➞ (Operator speichern)	**=:**
| | (Absolutwert)	**abs(...)**
$\sqrt{}$()	**sqrt(...)**
Σ() (Vorlage Summe)	**sumSeq(...)**
Π() (Vorlage Produkt)	**prodSeq(...)**
$\sin^{-1}$(), $\cos^{-1}$(), ...	**arcsin(...)**, **arccos(...)**, ...
ΔListe()	**deltaList(...)**

Von der Computertastatur

Sonderzeichen:	Tastenkürzel:
i (imaginäre Konstante)	**@i**
e (natürlicher Logarithmus zur Basis e)	**@e**
E (wissenschaftliche Schreibweise)	**@E**
T (Transponierte)	**@t**

Sonderzeichen:	Tastenkürzel:
ʳ (Bogenmaß)	`@r`
° (Grad)	`@d`
ᵍ (Neugrad)	`@g`
∠ (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 (ʳ), 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→r und “r”→s1, dann #s1=10.

Postfix-Operatoren

Postfix-Operatoren sind Operatoren, die direkt nach einem Argument stehen, zum Beispiel 5!, 25% oder 60°15' 45". Argumente, auf die ein Postfix-Operator folgt, werden auf der vierten Prioritätsebene ausgewertet. Beispiel: Im Ausdruck 4^3! wird zuerst 3! ausgewertet. Das Ergebnis 6 wird dann als Exponent für 4 verwendet, und das Endergebnis ist 4096.

Potenz

Potenzen (^) und elementweise Potenzen (.^) werden von rechts nach links ausgewertet. Der Ausdruck 2^3^2 wird zum Beispiel wie 2^(3^2) ausgewertet, hat also das Ergebnis 512. Er unterscheidet sich damit vom Ausdruck (2^3)^2 mit dem Ergebnis 64.

Negation

Zum Eingeben einer negativen Zahl drücken Sie [(−)] und geben dann die Zahl ein. Postfix-Operatoren und Potenzen werden vor der Negation ausgewertet. Das Ergebnis von $^-x2$ ist zum Beispiel eine negative Zahl; $^-9^2 = {}^-81$. Um eine negative Zahl zu quadrieren, verwenden Sie Klammern: $(^-9)^2$, Ergebnis 81.

Einschränkung („|“)

Das Argument nach dem womit-Operator „|“ stellt eine Reihe von Einschränkungen dar, die beeinflussen, wie das Argument vor dem Operator ausgewertet wird.

TI-Nspire CX II – TI-Basic Programmierfunktionen

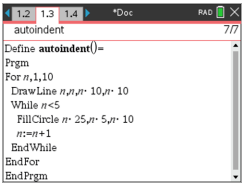
Automatisches Einrücken im Programmierungseditor

Der TI-Nspire™ Programmeditor rückt Anweisungen nun automatisch in einem Blockbefehl ein.

Blockbefehle sind If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry.

Der Editor stellt in einem Blockbefehl Programmbefehlen automatisch Leerstellen voran. Der Schließbefehl des Blocks wird am Öffnungsbefehl ausgerichtet.

Das unten stehende Beispiel zeigt das automatische Einrücken in Befehlen mit verschachtelten Blöcken.



Bei Codefragmenten, die kopiert und eingefügt werden, wird deren ursprüngliche Einrückung beibehalten.

Wird ein in einer früheren Version der Software erstelltes Programm geöffnet, wird die ursprüngliche Einrückung beibehalten.

Verbesserte Fehlermeldungen für TI-Basic

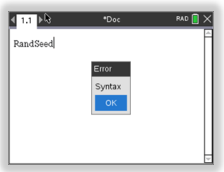
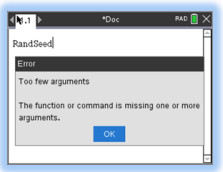
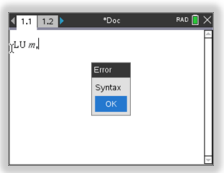
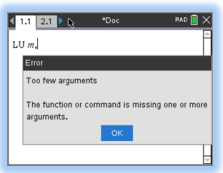
Fehler

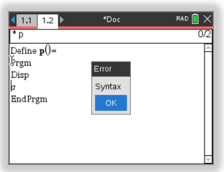
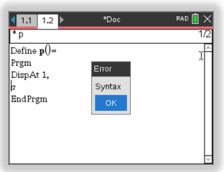
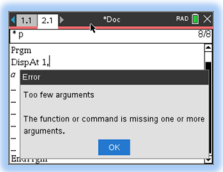
Fehlermeldungen	Neue Meldung
Fehler in der Bedingungsanweisung (**If/While**)	Eine bedingte Anweisung hat **RICHTIG** oder **FALSCH** nicht aufgeklärt. **HINWEIS:** Durch die Platzierung des Cursors auf die Linie mit dem Fehler muss nicht mehr angegeben werden, ob der Fehler ein „**If**“-Ausdruck oder ein „**While**“-Ausdruck ist.
EndIf fehlt	Erwartete **EndIf**, fand aber eine andere End-Anweisung
EndFor fehlt	Erwartete **EndFor**, fand aber eine andere End-Anweisung
EndWhile fehlt	Erwartete **EndWhile**, fand aber eine andere End-Anweisung

Fehlermeldungen	Neue Meldung
EndLoop fehlt	Erwartete **EndLoop**, fand aber eine andere End-Anweisung
EndTry fehlt	Erwartete **EndTry**, fand aber eine andere End-Anweisung
„**Then**“ nach **If** <condition> nicht angegeben	**If..Then** fehlt
„**Then**“ nach **ElseIf** <condition> nicht angegeben	**Then** fehlt in Block: **ElseIf**.
Wenn „**Then**“, „**Else**“ und „**ElseIf**“ außerhalb der Steuerblöcke gefunden wurden	**Else** ungültig außerhalb der Blöcke: **If..Then..EndIf** oder **Try..EndTry**
„**ElseIf**“ erscheint außerhalb des „**If..Then..EndIf**“-Blocks	**ElseIf** ungültig außerhalb des Blocks: **If..Then..EndIf**
„**Then**“ erscheint außerhalb des „**If....EndIf**“-Blocks	**Then** ungültig außerhalb des Blocks: **If..EndIf**

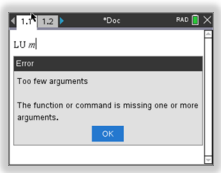
Syntaxfehler

Wenn Befehle, die ein oder mehrere Argumente erfordern, mit einer unvollständigen Argumentenliste aufgerufen werden, wird anstelle eines „**Syntax**“-Fehlers ein „**Zu wenige Argumente**“-Fehler ausgegeben.

Aktuelles Verhalten	Neues CX II-Verhalten
	
	

Aktuelles Verhalten	Neues CX II-Verhalten
	
	

Hinweis: Wenn auf eine unvollständige Argumentenliste kein Komma folgt, lautet die Fehlermeldung: „zu wenig Argumente“. Bei früheren Versionen war das genauso.



Konstanten und Werte

Die folgende Tabelle führt die Konstanten und ihre Werte auf, die verfügbar sind, wenn eine Einheitenumrechnung durchgeführt wird (NIST 2018). 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	8987551792,261 _m/_F
_Fc	Faraday-Konstante	96485,33212 _coul/_mol
_g	Erdbeschleunigung	9,80665 _m/_s^2
_Gc	Gravitationskonstante	6,6743E-11 _m^3/_kg/_s^2
_h	Plancksche Konstante	6,62607015E-34 _J _s
_k	Boltzmann-Konstante	1,380649E-23 _J/_°K
_μ0	Permeabilität des Vakuums	1,25663706127E-6 _N/_A^2
_μb	Bohr-Magneton	9,274009994E-24 _J _m^2/_Wb
_Me	Ruhemasse des Elektrons	9,10938371390E-31 _kg
_Mμ	Myonmasse	1,883531627E-28 _kg
_Mn	Ruhemasse des Neutrons	1,67492749804E-27 _kg
_Mp	Ruhemasse des Protons	1,67262192595E-27 _kg
_Na	Avogadro-Zahl	6,02214076E23 /_mol
_q	Elektronenladung	1,602176634E-19 _coul
_R	Druckkonstante für Wasserpotenzial	0,0831 L bars/mole K
_Rb	Bohr-Radius	5,29177210903E-11 _m
_Rc	Molare Gaskonstante	8,314462618 _J/_mol/_°K
_Rdb	Rydberg-Konstante	10973731,568160/_m
_Re	Elektronenradius	2,8179403262E-15 _m
_u	Atommasse	1,66053906893E-27 _kg
_Vm	Molvolumen	2,241396954E-2 _m^3/_mol
_ε0	Permittivität des Vakuums	8,8541878188E-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 178).

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: • nicht mit einer Ziffer beginnt • keine Leerzeichen oder Sonderzeichen enthält • keine unzulässigen Unterstriche oder Punkte enthält • die maximale Zeichenlänge nicht überschreitet Weitere Einzelheiten finden Sie im Abschnitt Calculator in der Dokumentation.
160	Argument muss Ausdruck sein
165	Batteriespannung zu niedrig zum Senden/Empfangen Setzten Sie vor dem Senden oder Empfangen neue Batterien ein.
170	Grenze

Fehlercode	Beschreibung
	Um das Suchintervall zu definieren, muss die untere Grenze kleiner sein als die obere Grenze.
180	Abbruch Die Taste [esc] oder [on] wurde gedrückt, während eine lange Berechnung oder ein Programm ausgeführt wurde.
190	Zirkuläre Definition Diese Meldung wird angezeigt, um zu verhindern, dass durch unendliches Ersetzen von Variablenwerten bei der Vereinfachung der Platz im Hauptspeicher nicht ausreicht. Dieser Fehler wird beispielsweise durch 'a+1->a' ausgelöst, wenn a eine nicht definierte Variable ist.
200	Zusammengesetzter Ausdruck ungültig Diese Fehlermeldung würde zum Beispiel durch 'solve(3x^2-4=0,x) | x<0 or x>5' ausgelöst werden, weil die Einschränkung durch “oder (or)” anstatt “und (and)” getrennt wird.
210	Ungültiger Datentyp Ein Argument weist einen falschen Datentyp auf.
220	Abhängiger Grenzwert
230	Dimension Ein Listen- oder Matrixindex ist ungültig. Wenn beispielsweise die Liste {1,2,3,4} in L1 gespeichert wird, ist L1[5] ein Dimensionsfehler, weil L1 nur vier Elemente enthält.
235	Dimensionsfehler. Nicht genügend Elemente in den Listen.
240	Dimensionsfehler Zwei oder mehr Argumente müssen die gleiche Dimension haben. So ist beispielsweise [1,2]+[1,2,3] ein Dimensionsfehler, weil die Matrizen eine unterschiedliche Anzahl von Elementen enthalten.
250	Division durch Null
260	Bereichsfehler Ein Argument muss in einem festgelegten Bereich sein. rand(0) ist zum Beispiel nicht gültig.
270	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 \var ist beispielsweise ungültig.
575	Polarkomplex ungültig
580	Programmaufruf ungültig Programme können nicht innerhalb von Funktionen oder Ausdrücken wie z.B. '1+p(x)' aufgerufen werden, wenn p ein Programm ist.
600	Tabelle ungültig
605	Verwendung der Einheiten ungültig
610	Variablenname in Lokal-Anweisung ungültig
620	Variablen- bzw. Funktionsname ungültig
630	Variablenverweis ungültig
640	Vektorsyntax ungültig
650	Kabelübertragung gestört Eine Übertragung zwischen zwei Geräten wurde nicht abgeschlossen. Überprüfen Sie, dass das Kabel an beiden Seiten fest angeschlossen ist.
665	Diagonalisierung der Matrix nicht möglich
670	Wenig Speicher 1. Löschen Sie Daten in diesem Dokument 2. Speichern und schließen Sie dieses Dokument Wenn 1 und 2 fehlschlagen, nehmen Sie die Batterien heraus und setzen Sie sie wieder ein
672	Ressourcenauslastung
673	Ressourcenauslastung
680	fehlt (
690	fehlt)
700	fehlt “
710	fehlt]

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	Variable ist nicht definiert Der Variablen wurde kein Wert zugewiesen. Verwenden Sie einen der folgenden Befehle: • sto → • **:=** • **Definiere** um Variablen Werte zuzuweisen.
965	Betriebssystem nicht lizensiert
970	Variable ist aktiv, daher keine Verweise oder Änderungen zulässig
980	Variable ist geschützt
990	Ungültiger Variablenname Stellen Sie sicher, dass der Name die maximale Zeichenlänge nicht überschreitet
1000	Fenstervariable nicht im Bereich
1010	Zoom
1020	Interner Fehler
1030	Verletzung des Zugriffsschutzes auf geschützten Speicher
1040	Nicht unterstützte Funktion. Für diese Funktion ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1045	Nicht unterstützter Operator. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1050	Nicht unterstütztes Merkmal. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1060	Das Eingabeargument muss numerisch sein. Nur Eingaben, die numerische Werte enthalten, sind zulässig.
1070	Argument der trig. Funktion ist zu groß für eine exakte Vereinfachung
1080	Keine Unterstützung von Antw (Ans). Diese Applikation unterstützt nicht Antw (Ans).
1090	Funktion ist nicht definiert. Verwenden Sie einen der folgenden Befehle: • **Definiere** • **:=** • sto → um eine Funktion zu definieren.

Fehlercode	Beschreibung
1100	Nicht-reelle Berechnung Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{(-1)}$ ungültig. Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).
1110	Ungültige Grenzen
1120	Keine Zeichenänderung
1130	Argument kann weder eine Liste noch eine Matrix sein
1140	Argumentfehler Das erste Argument muss ein Polynomausdruck im zweiten Argument sein. Wenn das zweite Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1150	Argumentfehler Die ersten zwei Argumente müssen Polynomausdrücke im dritten Argument sein. Wenn das dritte Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1160	Bibliotheks-Pfadname ungültig Ein Pfadname muss in der Form *xxx**yyy* angegeben werden, wobei: • Der *xxx* Teil kann 1 bis 16 Zeichen haben. • Der *yyy* Teil kann 1 bis 15 Zeichen haben. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1170	Verwendung des Bibliotheks-Pfadnamens ungültig • Ein Wert kann einem Pfadnamen nicht mit **Definiere (Define)**, **:=** oder sto → zugewiesen werden. • Ein Pfadname kann nicht als lokale Variable festgelegt oder als Parameter in einer Funktions- oder Programmdefinition verwendet werden.
1180	Bibliotheks-Variablenname ungültig. Vergewissern Sie sich, dass der Name: • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 15 Zeichen Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1190	Bibliotheks-Dokument nicht gefunden:

Fehlercode	Beschreibung
	• Vergewissern Sie sich, dass sich die Bibliothek im Ordner MyLib befindet. • Aktualisieren Sie die Bibliotheken. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1200	Bibliothaksvariable nicht gefunden: • 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. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1210	Unzulässiger Name für Bibliothekskurzform. Vergewissern Sie sich, dass der Name: • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 16 Zeichen • nicht reserviert ist Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation.
1220	Bereichsfehler: Die Funktionen tangentLine und normalLine unterstützen nur Funktionen mit reellen Werten.
1230	Bereichsfehler. Im Grad- und Neugradmodus werden die trigonometrischen Konversionsoperatoren nicht unterstützt.
1250	Argumentfehler System linearer Gleichungen verwenden. Beispiel für ein System zweier linearer Gleichungen mit den Variablen x und y: 3x+7y=5 2y-5x=-1
1260	Argumentfehler: Das erste Argument von nfMin oder nfMax muss ein Ausdruck in einer einzigen Variablen sein. Es darf keine andere Variable ohne Wert außer der interessierenden Variablen enthalten.
1270	Argumentfehler

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 187.

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(): • 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 −∞
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

!

!, Fakultät 208

"

", Sekunden-Schreibweise 217

#

#, Umleitung 214
#, Umleitungsoperator 245

%

%, Prozent 203

*****

*, multiplizieren 198

-

-, subtrahieren 197

.

.*, Punkt-Multiplikation 202
.-, Punkt-Subtraktion 201
./, Punkt-Division 202
.^, Punkt-Potenz 202
.+, Punkt-Addition 201

/

/, dividieren 199

:

:=, zuweisen 221

^

^, Potenz 200
$^{-1}$, Kehrwert 218

|

|, womit-Operator 219

'

', Minuten-Schreibweise 217

+

+, addieren 196

<

<, kleiner als 205

=

=, gleich 203
≠, ungleich[*] 204

>

>, größer als 206

∏

∏, Produkt 211

∑

∑( ), Summe 211
∑Int( ) 212
∑Prn( ) 213

√

√( ), Quadratwurzel 210

∠

∠ , winkel 217

∫

∫, Integral 210

≤

≤, kleiner oder gleich 205

≥

≥, größer oder gleich 206

►

►, in Neugrad umwandeln 75
►approxFraction( ) 13
►Base2, Binärdarstellung[Base2] 17
►Base10, Anzeige als ganze Dezimalzahl[Base10] 18
►Base16, Hexadezimaldarstellung [Base16] 19
►Cylind, Anzeige als Zylindervektor [Cylind (Zylindervektor)] ... 38
►DD, Anzeige als Dezimalwinkel[DD (Dezimalwinkel)] 39
►Decimal, Anzeige als Dezimalzahl [Dezimal] 39
►DMS, Anzeige als Grad/Minute/Sekunde[DMS (GMS)] 47
►Polar, Anzeige als Polarvektor [Polar] 125
►Rad, in Bogenmaß umwandeln 134
►Rect, Anzeige als kartesischer Vektor 137
►Sphere, Anzeige als sphärischer Vektor[Sphere (Kugelkoordinaten)] 164

⇒

⇒ , logische Implikation 207, 242

→

→, speichern 220

⇔

⇔, logische doppelte Implikation[*] 208

©

©, Kommentar 221

°

°, Grad/Minute/Sekunde 217
°, Grad-Schreibweise 216

#

0b, binäre Anzeige 221

#

0h, hexadezimale Anzeige 221

#

10^( ), Potenz von zehn 218

A

Abbruch, Exit 53
Ableitungen
erste Ableitung, d( ) 209
numerische Ableitung, nDeriv( ) 113
numerische Ableitung, nDerivative( ) 111
abrufen/zurückgeben Variableninformationen, getVarInfo( ) 70
Abrufen/zurückgeben Variableninformationen, getVarInfo( ) 74
abs( ), Absolutwert 7
Absolutwert
Vorlage für 3-4
addieren, + ... 196
als kartesischen Vektor anzeigen, ►Rect 137
Amortisationstabelle, amortTbl( ) .. 7, 16
amortTbl( ), Amortisationstabelle .. 7, 16
and, Boolean operator 8
and, Boolesches und 8
angle( ), Winkel 9
ANOVA, einfache Varianzanalyse ... 10
ANOVA2way, zweifache Varianzanalyse 10
Ans, letzte Antwort 13
Antwort (letzte), Ans 13
Anz, Daten anzeigen 151
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 ... 164
Zylindervektor, ►Cylind 38
Anzeige als kartesischer Vektor, ►Rect 137
Anzeige als sphärischer Vektor, ►Sphere 164
Anzeige als Zylindervektor, ►Cylind . 38
approx( ), approximieren 13
approximieren, approx( ) 13
approxRational( ) 14
arccos() 14
arccosh() 14
arccot() 14
arccoth() 14
arccsc() 14
arccsch() 14
arcsec() 14
arcsech() 14
arcsin() 15
arcsinh() 15
arctan() 15
arctanh() 15
Argumente in TVM-Funktionen 182
Arkuskosinus, $\cos^{-1}$( ) 29
Arkussinus, $\sin^{-1}$( ) 159
Arkustangens, $\tan^{-1}$( ) 172
augment( ), erweitern/verketten ... 15
Ausdrücke
String in Ausdruck, expr( ) 55
Ausschließung mit „|“ Operator 219
Auswertungsreihenfolge 244
avgRC( ), durchschnittliche Änderungsrate 15

B

Befehl Stopp 168
benutzerdefinierte Funktionen 40
benutzerdefinierte Funktionen und Programme 41
Bestimmtes Integral
Vorlage für 6
Bibliothek
erstelle Tastaturbefehle für Objekte 85
binär
Anzeige, 0b 221
Darstellung, ►Base2 17
binomCdf( ) 20, 81
binomPdf( ) 20
Bogenmaß, r 215
Boolean operators
and 8
Boolesch
und, and 8
Boolesche Operatoren
⇒ 207, 242
⇔ 208
nand 110
nicht 117
nor 115
oder 121
xor 189
Brüche
propFrac (Echter Bruch) 129
Vorlage für 1

C

Cdf( ) 57
ceiling( ), Obergrenze 21
centralDiff( ) 21
char( ), Zeichenstring 22
ClearAZ 24
colAugment 25
colDim( ), Spaltendimension der Matrix 25
colNorm( ), Spaltennorm der Matrix 26
conj( ), Komplex Konjugierte 26
constructMat( ), Matrix erstellen ... 26
corrMat( ), Korrelationsmatrix 27
cos( ), Kosinus 28
$\cos^{-1}$, Arkuskosinus 29
cosh( ), Cosinus hyperbolicus 30
$\cosh^{-1}$( ), Arkuskosinus hyperbolicus 30
cot( ), Kotangens 31
$\cot^{-1}$( ), Arkuskotangens 31
coth( ), Kotangens hyperbolicus 32
$\coth^{-1}$( ), Arkuskotangens hyperbolicus 32
countIf( ), Elemente in einer Liste bedingt zählen 33
cPolyRoots() 34
crossP( ), Kreuzprodukt 34
csc( ), Kosekans 34
$\csc^{-1}$( ), inverser Kosekans 35
csch( ), Kosekans hyperbolicus 35
csch^{-1}( ), inverser Kosekans 36

hyperbolicus .....
CubicReg, kubische Regression 36
Cycle, Zyklus 37

D

d( ), erste Ableitung 209
Daten anzeigen, Anz 151
Daten anzeigen, Disp 45
dbd( ), Tage zwischen Daten 38
Define, definiere 40
Definiere 40
Definiere LibPriv (Define LibPriv) ... 41
Definiere LibPub (Define LibPub) ... 41
Definiere, Define 40
definieren
öffentliche Funktion / öffentliches Programm 41
private Funktion oder Programm 41
deltaList() 42
DelVar, Variable löschen 42
delVoid( ), ungültige Elemente entfernen 43
det( ), Matrixdeterminante 43
Dezimal
Anzeige als ganze Zahl, ►Base10 18
Winkelanzeige, ►DD 39
diag( ), Matrixdiagonale 44
Diagonalform, ref( ) 138
dim( ), Dimension 44
Dimension, dim( ) 44
DispAt 45
dividieren, / 199
dotP( ), Skalarprodukt 48
drehen, rotate( ) 146
durchschnittliche Änderungsrate, avgRC( ) 15

E

e Exponent
Vorlage für 2
e hoch x, e^( ) 48, 54
E, Exponent 214
e^( ), e hoch x 48
echter Bruch, propFrac 129
eff( ), Nominal- in Effektivsatz konvertieren 49
Effektivsatz, eff( ) 49
Eigenvektor, eigVc( ) 49
Eigenwert, eigVl( ) 50
eigVc( ), Eigenvektor 49
eigVl( ), Eigenwert 50
Einheitsmatrix, identity( ) 75
Einheitsvektor, unitV( ) 185
Einstellungen, hole aktuellen 71
Elemente in einer Liste bedingt zählen, countIf( ) 33
Elemente in einer Liste zählen, zähle ( ) 32
else if, ElseIf 50
else, Else 76
ElseIf, else if 50
end
for, EndFor 59
if, EndIf 76
Schleife, EndLoop 99
while, EndWhile 189
end if, EndIf 76
end while, EndWhile 189
Ende
Funktion, EndFunc 64
Ende der Schleife, EndLoop 99
EndWhile, end while 189
Entfernen
ungültige Elemente aus Liste ... 43
EOS (Equation Operating System) .. 244
Equation Operating System (EOS) .. 244
Ergebnisse mit zwei Variablen, TwoVar 183
Ergebnisse, Statistik 165
Ergebniswerte, Statistik 166
Ersetzung durch „|“ Operator 219
erste Ableitung
Vorlage für 5
erweitern/verketten, augment( ) ... 15
euler( ), Euler function 51
Exit, Abbruch 53
exp( ), e hoch x 54
Exponent, E 214
Exponenten
Vorlage für 1
Exponentielle Regression, ExpReg .. 55
expr( ), String in Ausdruck 55
ExpReg, exponentielle Regression .. 55

F

factor( ), Faktorisiere 56
Faktorisiere, factor( ) 56
Fakultät, ! 208
Fehler übergeben, ÜbgebFeh 123
Fehler und Fehlerbehebung
  Fehler löschen, LöFehler 24
  Fehler übergeben, ÜbgebFeh .. 123
Fehlercodes und -meldungen 259
festlegen
  Modus, setMode( ) 154
Fill, Matrix füllen 57
Finanzfunktionen, tvmFV( ) 181
Finanzfunktionen, tvmI( ) 181
Finanzfunktionen, tvmN( ) 181
Finanzfunktionen, tvmPmt( ) 182
Finanzfunktionen, tvmPV( ) 182
FiveNumSummary 58
floor( ), Untergrenze 58
Folge, seq( ) 152
For 59
For, for 59
for, For 59
format( ), Formatstring 60
Formatstring, format( ) 60
fpart( ), Funktionsteil 60
freqTable( ) 61
Frobeniusnorm, norm( ) 116
Füllen 232-233
Func, Funktion 64
Func, Programmfunktion 64
Funktion beenden, EndFunc 64
Funktionen
  benutzerdefiniert 40
  Programmfunktion, Func 64
  Teil, fpart( ) 60
Funktionen und Variablen
  kopieren 27

G

g, Neugrad 215
ganze Zahl, int( ) 79
Ganzzahl teilen, intDiv( ) 79
ganzzahliger Teil, iPart( ) 82
gcd( ), größter gemeinsamer Teiler . 64
gehe zu, Goto 75
geomCdf( ) 65
geomPdf( ) 65
Get 65, 234
getDenom( ), Nenner
  holen/zurückgeben 67
getLangInfo( ), Sprachinformationen
  abrufen/zurückgeben 70
getLockInfo( ), testet den Gesperrt-
  Status einer Variablen oder
  Variablengruppe 71
getMode( ), getMode-Einstellungen 71
getNum( ), Zähler
  holen/zurückgeben 73
GetStr 73
getType( ), get type of variable 73
getVarInfo( ),
  Variableninformationen
  abrufen/zurückgeben 74
gleich, = ... 203
Gleichungssystem (2 Gleichungen)
  Vorlage für 3
Gleichungssystem (n Gleichungen)
  Vorlage für 3
Gleichungssystem, simult( ) 158
Goto, gehe zu 75
Grad-/Minuten-/Sekundenanzeige,
  ►DMS 47
Grad-Schreibweise, ° 216
größer als, > 206
Größer oder gleich, ≥ 206
größter gemeinsamer Teiler, gcd( ) . 64
Gruppen, Gesperrt-Status testen ... 71
Gruppen, sperren und entsperren .. 95, 185

H

Häufigkeit( ) 62
hexadezimal
  Anzeige, ►Base16 19
  Anzeige, 0h 221
holen/zurückgeben
  Nenner, getDenom( ) 67
  Zähler, getNum( ) 73
holeTast 67
Hyperbolisch
  Arkuskosinus, $\cosh^{-1}$( ) 30
  Arkussinus, $\sinh^{-1}$( ) 161
  Arkustangens, $\tanh^{-1}$( ) 173
  Cosinus, cosh( ) 30
  Sinus, sinh( ) 160
  Tangens, tanh( ) 173

I

identity(), Einheitsmatrix 75
If, if 76
if, If 76
ifFn( ) 77
imag( ), Imaginärteil 78
Imaginärteil, imag( ) 78
in String, inString( ) 78
inString( ), in String 78
int( ), ganze Zahl 79
intDiv( ), Ganzzahl teilen 79
Integral, ∫ 210
Interpolieren( ), interpolieren 79
invF( ) 80
invNorm( ), inverse kumulative
Normalverteilung) 82
invt( ) 82
Invχ^2( ) 80
iPart( ), Ganzzahliger Teil 82
irr( ), interner Zinsfluss
interner Zinsfluss, irr( ) 82
isPrime( ), Primzahltest 83
isVoid( ), Test auf Ungültigkeit 83

K

kartesische x-Koordinate, P►Rx( ) ... 122
kartesische y-Koordinate, P►Ry( ) ... 123
Kehrwert, ^⁻¹ 218
Ketten
drehen, rotate( ) 146
kleiner als, < 205
Kleiner oder gleich, ≤ 205
kleinstes gemeinsames Vielfaches,
lcm 84
Kombinationen, nCr( ) 111
Kommentar, © 221
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 84
leere (ungültige) Elemente 240
left( ), links 85
LibPriv 41
LibPub 41
libShortcut( ), erstelle
Tastaturbefehle für
Bibliotheksobjekte 85
lineare Regression, LinRegAx 87
lineare Regression, LinRegBx 86
Lineare Regression, LinRegBx 88
links, left( ) 85
LinRegBx, lineare Regression 86
LinRegMx, lineare Regression 87
LinRegtIntervals, lineare Regression 88
LinRegtTest 90
linSolve() 91
list►mat( ), Liste in Matrix 92
Liste in Matrix, list►mat( ) 92
Liste, Elemente bedingt zählen 33
Liste, Elemente zählen in 32
Listen
Differenzen in einer Liste, Δlist( ) 92
erweitern/verketten, augment( ) 15
in absteigender Reihenfolge
sortieren, SortD 163
in aufsteigender Reihenfolge
sortieren, SortA 163
Kreuzprodukt, crossP( ) 34
kumulierte Summe,
cumulativeSum( ) 37
leere Elemente in 240
Liste in Matrix, list►mat( ) 92
Matrix in Liste, mat►list( ) 101
Maximum, max( ) 101
Minimum, min( ) 105
neu, newList( ) 112
Produkt, product( ) 128
Skalarprodukt, dotP( ) 48
Summe, sum( ) 169
Summierung, sum( ) 170
Teil-String, mid( ) 104

ln( ), natürlicher Logarithmus 93
LnReg, logarithmische Regression .. 93
Local, lokale Variable 95
Lock, Variable oder Variablengruppe
sperren 95
LöFehler, Fehler löschen 24
Logarithmen 93
Logarithmische Regression, LnReg .. 93
Logarithmus
Vorlage für 2
logische doppelte Implikation, ⇔ .. 208
logische Implikation, ⇒ 207, 242
Logistic, logistische Regression 96
LogisticD, logistische Regression 97
Logistische Regression, Logistic 96
Logistische Regression, LogisticD ... 97
lokal, Local 95
lokale Variable, Local 95
Loop, Schleife 99
Löschen 227
Fehler, LöFehler 24
ungültige Elemente aus Liste ... 43
löschen
Variable, DelVar 42
LU, untere/obere Matrixzerlegung . 100

M

Marke, Lbl 84
mat►list( ), Matrix in Liste 101
Matrix
Diagonalform, ref( ) 138
reduzierte Diagonalform, rref( ) 149
Matrix (1 × 2)
Vorlage für 4
Matrix (2 × 1)
Vorlage für 4
Matrix (2 × 2)
Vorlage für 4
Matrix (m × n)
Vorlage für 4
Matrix erstellen, constructMat()( ) .. 26
Matrix in Liste, mat►list( ) 101
Matrixzeilenaddition, rowAdd( ) 148
Matrixzeilentausch, rowSwap( ) 149
Matrizen
Determinante, det( ) 43
Diagonale, diag( ) 44
Dimension, dim( ) 44
Eigenvektor, eigVc( ) 49
Eigenwert, eigVl( ) 50
Einheitsmatrix, identity( ) 75
erweitern/verketten, augment( ) 15
füllen, Fill 57
kumulierte Summe,
cumulativeSum( ) 37
Liste in Matrix, list►mat( ) 92
Matrix in Liste, mat►list( ) 101
Matrixzeilenmultiplikation und -
addition, mRowAdd( ) . 107
Maximum, max( ) 101
Minimum, min( ) 105
neu, newMat( ) 113
Produkt, product( ) 128
Punkt-Addition, .+ 201
Punkt-Division, ./ 202
Punkt-Multiplikation, .* 202
Punkt-Potenz, .^ 202
Punkt-Subtraktion, .- 201
QR-Faktorisierung, QR 130
Spaltendimension, colDim( ) ... 25
Spaltennorm, colNorm( ) 26
Summe, sum( ) 169
Summierung, sum( ) 170
Transponierte, T 171
untere/obere Matrixzerlegung,
LU 100
Untermatrix, subMat( ) 169, 171
Zeilenoperation, mRow( ) 106
max( ), Maximum 101
Maximum, max( ) 101
mean( ), Mittelwert 102
median( ), Median 102
Median, median( ) 102
MedMed, Mittellinienregression ... 103
mid( ), Teil-String 104
min( ), Minimum 105
Minimum, min( ) 105
Minuten-Schreibweise, ... 217
mirr( ), modifizierter interner
Zinsfluss 105
mit, | 219
Mittellinienregression, MedMed ... 103
Mittelwert, mean( ) 102
mod( ), Modulo 106
Modi
festlegen, setMode( ) 154
Modifizierter interner Zinsfluss, mirr 105

( ) .....
Modulo, mod( ) 106
Moduseinstellungen, getMode( ) ... 71
mRow( ), Matrixzeilenoperation 106
mRowAdd( ),
Matrixzeilenmultiplikation
und -addition 107
Multipler linearer Regressions-t-Test 108
multiplizieren, * 198
MultReg (Mehrfachregression) 107
MultRegIntervals( )
(Mehrfachregressionsinterv
all) 107
MultRegTests( ) 108

N

nand, Boolescher Operator 110
natürlicher Logarithmus, ln( ) 93
nCr( ), Kombinationen 111
nDerivative( ), numerische Ableitung 111
Negation, Eingabe von negativen
Zahlen 245
Nettobarwert, npv ( ) 118
neu
Liste, newList( ) 112
Matrix, newMat( ) 113
Neugrad-Schreibweise, g 215
newList( ), neue Liste 112
newMat( ), neue Matrix 113
nfMax( ), numerisches
Funktionsmaximum 113
nfMin( ), numerisches
Funktionsminimum 113
nicht, Boolescher Operator 117
nInt( ), numerisches Integral 114
nom), Effektivzins in Nominalzins
konvertieren 114
Nominalzinssatz, nom( ) 114
nor, Boolescher Operator 115
norm( ), Frobeniusnorm 116
Normalverteilung invertieren
(invNorm( ) 82
Normalverteilungswahrscheinlichkeit,
normCdf( ) 116
normCdf( )
(Normalverteilungswahrschei
nlichkeit) 116
normPdf( )
(Wahrscheinlichkeitsdichte) 116
nPr( ), Permutationen 117
npv( ), Nettobarwert 118
nSolve( ), numerische Lösung 119
n-te Wurzel
Vorlage für 2
numerisch
Ableitung, nDeriv( ) 113
Ableitung, nDerivative( ) 111
Integral, nInt( ) 114
Lösung, nSolve( ) 119

O

Obergrenze, ceiling( ) 21, 34
Objekte
erstelle Tastaturbefehle für
Bibliothek 85
oder (Boolesch), oder 121
oder, Boolescher Operator 121
OneVar, Statistik mit einer Variable 119
Operatoren
Auswertungsreihenfolge 244
ord( ), numerischer Zeichencode ... 122

P

P►Rx( ), kartesische x-Koordinate ... 122
P►Ry( ), kartesische y-Koordinate ... 123
Pdf( ) 61
Permutationen, nPr( ) 117
piecewise( ) (Stückweise) 124
poissCdf( ) 124
poissPdf( ) 125
polar
Vektoranzeige, ►Polar 125
Polarkoordinate, R►Pr( ) 134
Polarkoordinate, R►Pθ( ) 133
polyEval( ), Polynom auswerten 126
Polynom auswerten, polyEval( ) 126
Polynome
auswerten, polyEval( ) 126
PolyRoots() 126
Potenz von zehn, 10^( ) 218
Potenz, ^ 200
Potenzregression,
PowerReg ... 126-127, 140, 142, 174
PowerReg, Potenzregression 127
Prgm, Definiere Programm 128

Primzahltest, isPrime( ) 83
prodSeq() 128
product( ), Produkt 128
Produkt ∏( )
Vorlage für 5
Produkt, ∏( ) 211
Produkt, product( ) 128
Programme
öffentliche Bibliothek definieren 41
Private Bibliothek definieren ... 41
Programme und Programmieren
E/A-Bildschirm anzeigen, Anz .. 151
E/A-Bildschirm anzeigen, Zeige . 45
Fehler löschen, LöFehler 24
programmieren
Daten anzeigen, Disp 45
Definiere Programm, Prgm 128
Fehler übergeben, ÜbgebFeh .. 123
Programmierung
Daten anzeigen, Anz 151
propFrac, echter Bruch 129
Prozent, % 203
Punkt
Addition, .+ 201
Division, ./ 202
Multiplikation, .* 202
Potenz, .^ 202
Subtraktion, .- 201

Q

QR,QR-Faktorisierung 130
QR-Faktorisierung, QR 130
Quadratische Regression, QuadReg . 131
Quadratwurzel
Vorlage für 1
Quadratwurzel, √( ) 210
Quadratwurzel, ‡( ) 164
QuadReg, quadratische Regression . 131
QuartReg, Regression vierter
Ordnung 132

R

r, Bogenmaß 215
R▸Pr( ), Polarkoordinate 134
R▸Pθ( ), Polarkoordinate 133
rand ( ), Zufallszahl 134
randBin, Zufallszahl 134
randInt( ), ganzzahlige Zufallszahl ... 135
randMat( ), Zufallsmatrix 135
randNorm( ), Zufallsnorm 136
randPoly( ), Zufallspolynom 136
randSamp( ) 136
RandSeed, Zufallszahl 136
real( ), reel 137
rechts, right( ) 79, 144
reduzierte Diagonalform, rref( ) 149
reell, real( ) 137
ref( ), Diagonalform 138
RefreshProbeVars 139
Regression vierter Ordnung,
QuartReg 132
Regressionen
exponentielle, ExpReg 55
kubische, CubicReg 36
lineare Regression, LinRegAx ... 87
lineare Regression, LinRegBx ... 86
Lineare Regression, LinRegBx .. 88
logarithmische, LnReg 93
Logistic (Logistisch) 96
logistische, Logistic 97
Mittellinie, MedMed 103
MultReg (Mehrfachregression) . 107
Potenzregression,
PowerRe
g 126-127, 140, 142, 174
quadratische, QuadReg 131
sinusförmige, SinReg 161
vierter Ordnung, QuartReg 132
remain( ), Rest 140
Request 140
RequestStr 142
Rest, remain( ) 140
Return, Rückgabe 143
right( ), rechts 144
right, right( ) 51, 187
rk23( ), Runge-Kutta-Funktion 144
rotate( ), drehen 146
round( ), runden 147
rowAdd( ), Matrixzeilenaddition 148
rowDim( ), Zeilendimension der
Matrix 148
rowNorm( ), Zeilennorm der Matrix 148
rowSwap( ), Matrixzeilentausch 149
rref( ), reduzierte Diagonalform 149
Rückgabe, Return 143
runden, round( ) 147

S

Schleife, Loop 99
Schreibweise Grad/Minute/Sekunde 217
sec( ), Sekans 149
sec⁻¹( ), Arkussekans 150
sech( ), Sekans hyperbolicus 150
sech⁻¹( ), Arkussekans hyperbolicus . 150
Sekunden-Schreibweise, " 217
seq( ), Folge 152
seqGen( ) 152
seqn( ) 153
sequence, seq( ) 152-153
setMode( ), Modus festlegen 154
shift( ), verschieben 155
sign( ), Zeichen 157
simult( ), Gleichungssystem 158
sin( ), Sinus 159
sin⁻¹( ), Arkussinus 159
sinh( ), Sinus hyperbolicus 160
sinh⁻¹( ), Arkussinus hyperbolicus ... 161
SinReg, sinusförmige Regression 161
Sinus, sin( ) 159
Sinusförmige Regression, SinReg ... 161
Skalar
  Produkt, dotP( ) 48
SortA, in aufsteigender Reihenfolge
    sortieren 163
SortD, in absteigender Reihenfolge
    sortieren 163
sortieren
  in absteigender Reihenfolge
    sortieren, SortD 163
  in aufsteigender Reihenfolge,
    SortA 163
speichern
  Symbol, → 220
Sprache
  Sprachinformation abrufen 70
sqrt( ), Quadratwurzel 164
Standardabweichung, stdDev( ) 166-167, 185
stat.results 165
stat.values 166
Statistik
  Ergebnisse mit zwei Variablen,
    TwoVar 183
  Fakultät, ! 208
  Kombinationen, nCr( ) 111
  Median, median( ) 102
  Mittelwert, mean( ) 102
  Permutationen, nPr( ) 117
  Standardabweichung,
    stdDev( ) 166-167, 185
  Statistik mit einer Variable,
    OneVar 119
  Varianz, variance( ) 186
  Zufallsnorm, randNorm( ) 136
  Zufallszahl, RandSeed 136
Statistik mit einer Variable, OneVar 119
stdDevPop( ), Populations-
    Standardabweichung 166
stdDevSamp( ), Stichproben-
    Standardabweichung 167
String
  Dimension, dim( ) 44
  Länge 44
string( ), Ausdruck in String 168
Stringlänge 44
Strings
  Ausdruck in String, string( ) 168
  Format, format( ) 60
  Formatieren 60
  in, inString 78
  links, left( ) 85
  rechts, right( ) 79, 144
  String in Ausdruck, expr( ) 55
  Teil-String, mid( ) 104
  Umleitung, # 214
  verschieben, shift( ) 155
  Zeichencode, ord( ) 122
  Zeichenstring, char( ) 22
strings
  right, right( ) 51, 187
Stückweise definierte Funktion (2
    Teile)
  Vorlage für 2
Stückweise definierte Funktion (n
    Teile)
  Vorlage für 3
Student-t-
    Wahrscheinlichkeitsdichte,
    tPdf( ) 177
subMat( ), Untermatrix 169, 171
subtrahieren, - 197
sum( ), Summe 169
sumIf( ) 170
Summe ∑( )
  Vorlage für 5

Summe der Tilgungszahlungen 213
Summe der Zinszahlungen 212
Summe, ∑( ) 211
Summe, sum( ) 169
sumSeq() 171

T

t test, t-Test 179
T, Transponierte 171
Tage zwischen Daten, dbd( ) 38
tan( ), Tangens 171
$\tan^{-1}$( ), Arkustangens 172
Tangens, tan( ) 171
tanh( ), Tangens hyperbolicus 173
$\tanh^{-1}$( ), Arkustangens hyperbolicus 173
Tastenkürzel 242
Tastenkürzel, Tastatur 242
tCdf(), Wahrscheinlichkeit einer
Student t-Verteilung 174
Teil-String, mid( ) 104
Test auf Ungültigkeit, isVoid( ) 83
Test_2S, Zwei-Stichproben F-Test ... 63
Text, Befehl 174
tInterval, Konfidenzintervall t 175
tInterval_2Samp, ZweiStichproben-t-
Konfidenzintervall 176
trace( ) 177
Transponierte, T 171
Try, Befehl zur Fehlerbehandlung ... 178
tTest, t-Test 179
tTest_2Samp, Zwei-Stichproben-t-
Test 180
TVM-Argumente 182
tvmFV( ) 181
tvmI( ) 181
tvmN( ) 181
tvmPmt( ) 182
tvmPV( ) 182
TwoVar, Ergebnisse mit zwei
Variablen 183

U

ÜbgebFeh, Fehler übergeben 123
Umleitung, # 214
Umleitungsoperator (#) 245
umwandeln
►Grad (Neugrad) 75
ungleich, ≠ 204
ungültige Elemente 240
ungültige Elemente, entfernen 43
unitV( ), Einheitsvektor 185
unLock, Variable oder
Variablengruppe
entsperren 185
Untergrenze, floor( ) 58
Untermatrix, subMat( ) 169, 171

V

Variable
Name aus String erstellen 245
Variable oder Funktion kopieren,
CopyVar 27
Variablen
alle einbuchstabigen löschen ... 24
lokal, Local 95
löschen, DelVar 42
Variablen und Funktionen
kopieren 27
Variablen und Variablengruppen
entsperren 185
Variablen und Variablengruppen
sperren 95
Variablen, sperren und entsperren 71, 95, 185
Varianz, variance( ) 186
varPop( ) (Populationsvarianz) 185
varSamp( ), Stichproben-Varianz 186
Vektoren
Anzeige als Zylindervektor,
►Cylind 38
Einheit, unitV( ) 185
Kreuzprodukt, crossP( ) 34
Skalarprodukt, dotP( ) 48
verschieben, shift( ) 155
Verteilungsfunktionen
binomCdf( ) 20, 81
binomPdf( ) 20
invNorm( ) 82
invt( ) 82
Invχ^2( ) 80
normCdf( )
(Normalverteilungswahrschei
nlichkeit) 116
normPdf( )
(Wahrscheinlichkeitsdi
chte) 116
poissCdf( ) 124

poissPdf( ) 125
tCdf( ) 174
tPdf( ) 177
$\chi^2$2way( ) 22
χ^2Cdf( ) 23
χ^2GOF( ) 23
χ^2Pdf( ) 24
void, test for 83
Vorlagen
Absolutwert 3-4
Bestimmtes Integral 6
Bruch 1
e Exponent 2
erste Ableitung 5
Exponent 1
Gleichungssystem (2 Gleichungen) 3
Gleichungssystem (n Gleichungen) 3
Logarithmus 2
Matrix (1 × 2) 4
Matrix (2 × 1) 4
Matrix (2 × 2) 4
Matrix (m × n) 4
n-te Wurzel 2
Produkt ∏( ) 5
Quadratwurzel 1
Stückweise definierte Funktion (2 Teile) 2
Stückweise definierte Funktion (n Teile) 3
Summe ∑( ) 5
zweite Ableitung 6

W

Wahrscheinlichkeit einer Student-t-Verteilung, tCdf( ) 174
Wahrscheinlichkeitsdichte, normPdf( ) 116
Warncodes und -meldungen 259
warnCodes( ), Warning codes 187
Warte-Befehl 187
wenn, when( ) 188
when( ), wenn 188
While, while 189
while, While 189
winkel, ∠ 217
Winkel, angle( ) 9
womit-Operator „|“ 219
womit-Operator, Auswerungsreihenfolge ... 244

X

x^2, Quadrat 201
XNOR 208
xor, Boolesches exklusives oder 189

Z

Zähle Tage zwischen Daten, dbd( ) .. 38
zähle( ), Elemente in einer Liste zählen 32
Zeichen
String, char( ) 22
Zeichencode, ord( ) 122
Zeichen, sign( ) 157
Zeichenfolgen
zum Erstellen von Variablennamen verwenden 245
Zeichenstring, char( ) 22
Zeichnen 228-230
Zeige, Daten anzeigen 45
Zeilendimension der Matrix, rowDim( ) 148
Zeilennorm der Matrix, rowNorm( ) 148
Zeitwert des Geldes, Anzahl Zahlungen 181
Zeitwert des Geldes, Barwert 182
Zeitwert des Geldes, Endwert 181
Zeitwert des Geldes, Zahlungsbetrag 182
Zeitwert des Geldes, Zinsen 181
zInterval, z-Konfidenzintervall 190
zInterval_1Prop, z-Konfidenzintervall für eine Proportion 191
zInterval_2Prop, z-Konfidenzintervall für zwei Proportionen 191
zInterval_2Samp, z-Konfidenzintervall für zwei Stichproben 192
zTest 193
zTest_1Prop, z-Test für eine Proportion 194
zTest_2Prop, z-Test für zwei Proportionen 195
zTest_2Samp, z-Test für zwei 195

Stichproben .....
Zufallsmatrix, randMat( ) 135
Zufallsnorm, randNorm( ) 136
Zufallspolynom, randPoly( ) 136
Zufallsstichprobe 136
Zufallszahl, RandSeed 136
zuweisen, := 221
Zwei-Stichproben F-Test 63
zweite Ableitung
Vorlage für 6
Zyklus, Cycle 37

Δ

Δlist( ), Listendifferenz 92

X

$\chi^2$2way 22
χ^2Cdf( ) 23
χ^2GOF 23
χ^2Pdf( ) 24