Programmieren in Python für den Grafiktaschenrechner TI-84 Plus CE-T *Python Edition*

Version 84CE Bundle 5.6.0.

Verwenden Sie die Online-Hilfe unter education.ti.com/eguide, um mehr über die TI-Technologie zu erfahren.

Wichtige Informationen

Sofern in der jeweiligen Programmlizenz nicht ausdrücklich anders aufgeführt, 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 dadurch verursacht werden, dies gilt für spezielle, begleitende und versehentliche Schäden sowie für Folgeschäden. Texas Instruments haftet maximal und ausschließlich in der Programmlizenz festgelegten Höhe, unabhängig vom jeweiligen Fall. Weiterhin haftet Texas Instruments nicht für Forderungen einer anderen Partei, die sich aus dem Gebrauch dieses Produkts ergeben, welcher Art diese Forderungen auch immer sein mögen.

„Python“ und die Python-Logos sind Warenzeichen oder eingetragene Warenzeichen der Python Software Foundation, die von Texas Instruments Incorporated mit Erlaubnis der Foundation verwendet werden.

Inhalt

Neuerungen

Neuerungen in der Python Programmier-App v5.5.0

TI-84 Plus CE-T *Python Edition*

Python-Programmierung

TI-84 Plus CE-T *Python Edition*

- Unterstützt Python-Programmierung mit der Python-App aus dem 84CE-Bundle v5.6.0. Aktualisieren Sie unter education.ti.com/84cetupdate auf die neueste Version.
- Bei geladener Python-App können Sie über 2nd apps oder prgm auf die Python-App zugreifen.

Hinweis: Welche Umgebung bietet Ihr CE-Taschenrechner für TI-Python?

- TI-84 Plus CE-T *Python Edition* mit 84CE-Bundle v5.6.0 oder höher

---

Übertragen von Python-Programmen

Beim Übertragen von Python-Programmen von einer Nicht-TI-Plattform auf eine TI-Plattform ODER von einem TI-Produkt auf ein anderes:

- Python-Programme, die Kernsprachenfunktionen und Standardbibliotheken (math, random usw.) verwenden, können ohne Änderungen importiert oder exportiert werden.
  Hinweis: Die Begrenzung der Listenlänge beträgt 100 Elemente.
- Programme, die plattformspezifische Bibliotheken – matplotlib (für PC), ti_plotlib/ti_system/ti_hub/usw. für TI-Plattformen verwenden, müssen bearbeitet werden, bevor sie auf einer anderen Plattform ausgeführt werden können.

Dies kann sogar zwischen TI-Plattformen erforderlich sein.

Neue Funktionen und TI-Python-Module

- Unterstützung komplexer Zahlentypen wie a+bj.
  - Siehe Menü „[Fns…] Types“ in Editor oder Shell.
- Zeitmodul
- TI-Module
  - ti_system
    - Rufen Sie eine Betriebssystemliste und eine Betriebssystem-Regressionsgleichung in einem Python-Programm auf. Erstellen Sie Listen in einem Python-Programm und speichern Sie diese in Betriebssystem-Listenvariablen. Die Begrenzung der Listenlänge beträgt 100 Elemente.
  - ti_plotlib
    - Führen Sie Python-Programme zur Darstellung von Statistik- und Funktionsdiagrammen aus.
  - ti_hub

- Erstellen Sie Python-Programme für den TI-Innovator™ Hub.

– ti_rover
  - Verwenden Sie die Python-Programmierung, um den TI-Innovator™ Rover zu steuern.

Erstellen Sie mit Vorlagen neue Programmtypen.

Wenn Ihr Programm notwendige Import-Anweisungen für Module erfordert, verwenden Sie beim Erstellen eines neuen Programms die Registerkarte „Types“. Die notwendigen Programmzeilen werden im Editor in Ihr neues Programm eingefügt. Dies ist besonders hilfreich bei MINT-Aktivitäten! Die Methodenvorlage „Plotting“ unterstützt erste Erfahrungen mit dem Schreiben eines Programms mit ti_plotlib.

Argument-Hilfsprogramme und Tipps im Menübildschirm

Eine Argumenthilfe hilft Ihnen bei der Auswahl des richtigen Arguments aus einem Menü, wenn Methoden String-Argumente enthalten. Keine manuelle Eingabe! Sie müssen nicht die korrekte Zeichenfolge nachschlagen!

Menübildschirm-Tipps sind bei Argumentbereichen, Voreinstellungen oder Tastendruck-Hinweisen verfügbar.

Aktualisierungen zur Tastatur der Python-App

math zeigt alle verfügbaren Module weiterhin an.

2nd [*i*] (über [.]) zeigt ein imaginäres j für eine komplexe Zahl a+bj in Python an.

Siehe auch: Tastatur

Software-Informationen

TI Connect™ CE

Connectivity-Unterstützung und *.py <> PY AppVar-Konvertierung für den TI-84 Plus CE-T *Python Edition*.

TI-SmartView™ CE-T

TI-84 Plus CE-T *Python Edition* Emulator unterstützt Python App v5.5.0

Beispielprogramme HELLO, GRAPH und LINREGR werden bei der Installation und beim Zurücksetzen geladen.

Der Datenimport-Assistent konvertiert für den CE-Emulator entsprechend formatierte *.csv-Dateien in Taschenrechnerlisten. Diese Funktion ist hilfreich, wenn Sie das Modul ti_system und externe Daten für die Python-Programmierung verwenden.

- Wenn Dezimalzahlen in der *.csv-Datei mit Verwendung eines Kommas dargestellt werden, wird die Datei nicht vom Datenimport-Assistenten konvertiert. Bitte überprüfen Sie die Zahlenformatierung Ihres Computer-Betriebssystems und konvertieren Sie das *.csv-Format, um die Dezimalpunktdarstellung zu verwenden. Beispiel: Die CE-Taschenrechnerliste und der Matrixeditor verwenden das Zahlenformat 12.34 und nicht 12,34.

Hinweis: Um Programme für TI-Innovator™ Hub oder TI-Innovator™ Rover auszuführen, senden Sie die Programme über TI Connect™ CE an den Taschenrechner. Beenden Sie die Python-App, bevor Sie über den Emulator-Explorer Dateien an den Computer und dann auf den Rechner übertragen. TI-Innovator™ Hub- und TI-Innovator™ Rover-Programme laufen nicht unter TI-SmartView™ CE-T.

Weitere Informationen zu den neuen und aktualisierten Funktionen finden Sie unter education.ti.com/84cetupdate.

Python-App

Informationen zur Verwendung, Navigation und Ausführung der Python-App finden Sie unter folgenden Punkten.

- Verwenden der Python-App
- Navigation in der Python-App
- Beispielaktivität

Verwenden der Python-App

Die Python-App ist für den TI-84 Plus CE-T *Python Edition* erhältlich. Die Informationen in diesem eGuide sind für die Verwendung mit dem TI-84 Plus CE-T *Python Edition* mit dem neuesten CE-Bundle bestimmt.

Wenn Sie die Python-Anwendung zum ersten Mal auf Ihrem TI-84 Plus CE-T *Python Edition* ausführen, weist Sie die Anwendung möglicherweise an, auf das neueste CE-Bundle für die neueste Python-App zu aktualisieren.
Gehen Sie auf education.ti.com/84cetupdate, um Ihren TI-84 Plus CE-T *Python Edition* zu aktualisieren.

Die Python-App bietet einen Dateimanager, einen Editor zum Erstellen von Programmen und eine Shell, um die Programme auszuführen und mit dem Python-Interpreter zu interagieren. Python-Programme, die als Python AppVars gespeichert oder erstellt werden, werden aus dem RAM ausgeführt. Python AppVars werden über den Speicherverwaltungsbildschirm des Betriebssystems archiviert, um die Speicherverwaltung von Python-Dateien zu unterstützen.

Hinweis: Wenn es sich bei Ihrem Taschenrechner um den TI-84 Plus CE-T handelt education.ti.com/84cetupdate die neuesten Informationen für Ihren CE.

Navigation in der Python-App

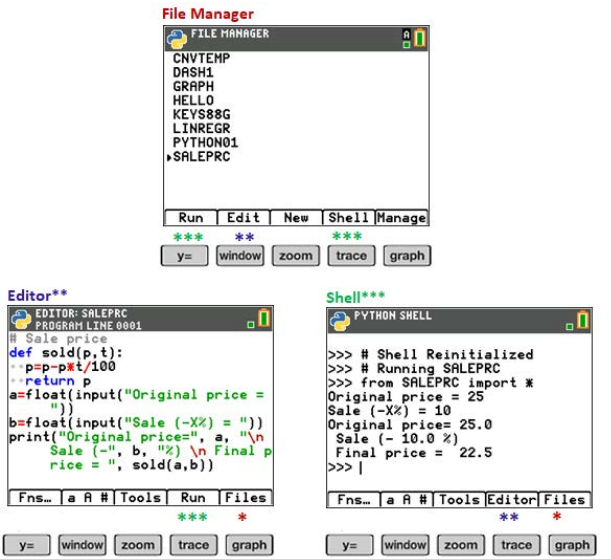
Verwenden Sie in der App die Schnelltasten auf dem Bildschirm, um zwischen den Arbeitsbereichen in der Python-App zu navigieren. In der Abbildung zeigen die Beschriftungen der Verknüpfungs-Registerkarten Folgendes an:

* Navigation zum Dateimanager [Files]

** Navigation zum Editor [Edit] oder [Editor]

*** Navigation zur Shell [Shell]

Auf die Verknüpfungs-Registerkarten im Bildschirm greifen Sie über die Grafik-Tastenzeile direkt unter dem Bildschirm zu. Siehe auch Tastatur. Die Menüs Editor>Tools und Shell>Tools enthalten ebenfalls Navigationsaktionen.



Beispielaktivität

Nutzen Sie die Beispielaktivität, um sich mit den Arbeitsbereichen in der Python-App vertraut zu machen.

- Erstellen Sie ein neues Programm über den Dateimanager.
- Schreiben Sie das Programm im Editor.
- Führen Sie das Programm in der Shell der Python-App aus.

Weitere Informationen zur Python-Programmierung auf Ihrem CE finden Sie in den Ressourcen für den
TI-84 Plus CE-T *Python Edition*.

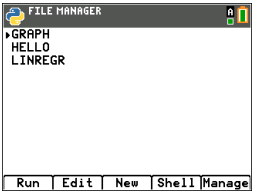
Erste Schritte:

- Führen Sie die Python-App aus.

Note: Actual screens may vary slightly from provided images.

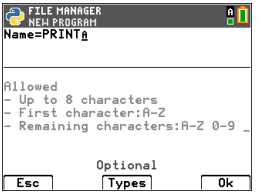
Geben Sie über den Dateimanager einen neuen Programmnamen ein.

- Drücken Sie zoom ([New]), um ein neues Programm zu erstellen.



Eingabe des neuen Dateinamens

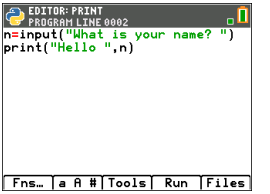
- Das Beispielprogramm wird „PRINT“ genannt. Geben Sie den Programmnamen ein und drücken Sie graph ([Ok]).
- Beachten Sie, dass sich der Cursor im ALPHA-Sperrmodus befindet. Geben Sie immer einen Programmnamen nach den auf dem Bildschirm vorgegebenen Vorgaben ein.



Tipp: Wenn sich der Cursor nicht im ALPHA-Sperrmodus befindet, drücken Sie 2nd alpha alpha für Großbuchstaben.

Geben Sie das Programm wie gezeigt ein.

Tipp: Die App ermöglicht eine schnelle Eingabe! Achten Sie während der Eingabe Ihres Programms immer auf den Cursor-Zustand!



Alphabetische Zeichen auf der Tastatur	alpha schaltet den Status des Einfüge-Cursors in Editor und

	Shell um. _ nicht alphabetisch a Kleinbuchstabe A Großbuchstabe
Wo ist das Gleichheitszeichen?	Drücken Sie sto→, wenn der Cursor auf _ steht. 
Wo befinden sich input() print()	[Fns...] I/O 1:print() 2:input()
Wo befindet sich das doppelte Anführungszeichen?	alpha ["] 
Wo befinden sich (und)?	Benutzen Sie die Tastatur, wenn der Cursor auf _ steht. 

Try-It! [a A #] und 2nd [catalog] sind bei Bedarf ebenfalls Helfer für die schnelle Eingabe!

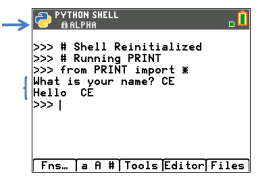
Führen Sie das Programm PRINT aus

- Drücken Sie im Editor trace ([Run]), um Ihr Programm in der Shell auszuführen.
- Geben Sie Ihren Namen bei der Eingabeaufforderung „What is your name?“ ein.
- In der Ausgabe wird „HELLO“ mit Ihrem Namen angezeigt.

Shell-Cursor-Statusanzeige.

Geben Sie Ihren Namen ein.

Ausgabe von PRINT wird angezeigt.



Hinweis: An der Shell-Eingabeaufforderung >>> können Sie einen Befehl ausführen, wie z. B. 2+3. Wenn Sie eine Methode aus math, random oder anderen verfügbaren Modulen verwenden, müssen Sie wie in jeder Python-Codierungsumgebung zuerst eine Modul-Importanweisung ausführen.

Einrichten einer Python-Sitzung mit Ihren Programmen

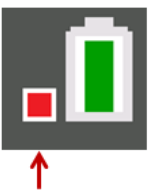
Wenn die Python-Anwendung gestartet wird, wird die CE-Verbindung für Ihre aktuelle Python-Sitzung mit der TI-Python-Umgebung synchronisiert. Sie sehen die Liste Ihrer Programme im RAM und in den dynamischen Modulen, während diese sich mit der Python-Umgebung synchronisieren.

Wenn die Python-Sitzung eingerichtet ist, signalisiert die Statusleiste durch eine grüne quadratische Anzeige neben dem Batteriesymbol, dass die Python-Sitzung einsatzbereit ist. Falls die Anzeige rot ist, warten Sie, bis sie wieder auf grün wechselt, nachdem die Python-Umgebung wieder verfügbar ist.

Möglicherweise sehen Sie eine Aktualisierung der Python-Distribution, wenn Sie die Python-App zusammen mit der Programmsynchronisierung nach der letzten Aktualisierung Ihres
TI-84 Plus CE-T *Python Edition* über education.ti.com/84cetupdate starten.

Trennen und erneutes Verbinden der Python-App

Wenn die Python-App ausgeführt wird, signalisiert eine Anzeige in der Statusleiste, ob Python einsatzbereit ist. Bis die Verbindung hergestellt ist, reagiert die CE-Tastatur möglicherweise nicht. Es empfiehlt sich, während Ihrer Python-Sitzung auf die Verbindungsanzeige in der Statusleiste zu achten.



Python nicht bereit

Python bereit

Bildschirmaufnahmen

Unter Verwendung von TI Connect™ CE unter education.ti.com/84cetupdate sind Bildschirmaufnahmen von jedem Python-App-Bildschirm erlaubt.

Python-Arbeitsbereiche

Die Python-App enthält drei Arbeitsbereiche für Ihre Python-Programmentwicklung.

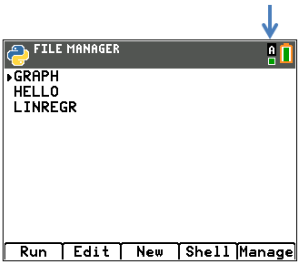
- Dateimanager
- Editor
- Shell

Python-Dateimanager

Der Dateimanager listet die im RAM Ihres Taschenrechners verfügbaren Python AppVars auf. Sie können Programme erstellen, bearbeiten und ausführen sowie zur Shell navigieren.

Drücken Sie im Alpha-Status einen beliebigen Buchstaben auf der Tastatur, um zu Programmen zu springen, die mit diesem Buchstaben beginnen.

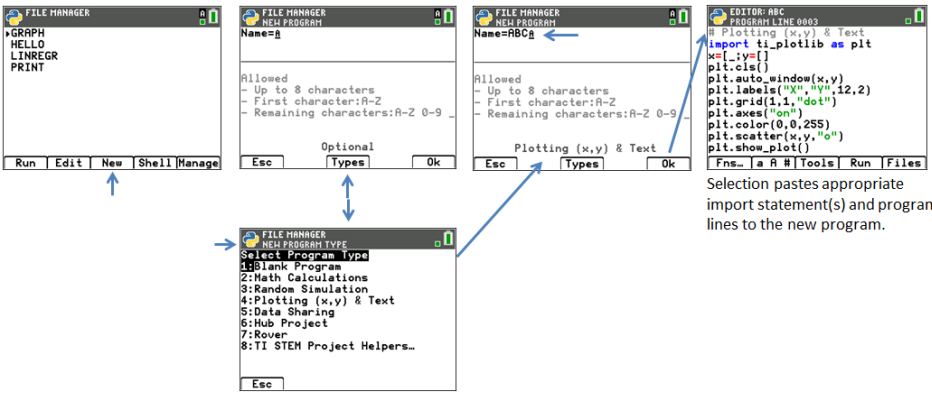
Drücken Sie bei Bedarf alpha, wenn die Anzeige **A** nicht in der Statusleiste angezeigt wird.



Tastenkürzel und Menüs im Dateimanager		
Menüs	**Taste**	**Beschreibung**
[Run]	y=	Wählen Sie mit ▲ oder ▼ ein Programm aus. Wählen Sie dann [Run], um das Programm auszuführen.
[Edit]	window	Wählen Sie mit ▲ oder ▼ ein Programm aus. Wählen Sie dann [Edit], um das Programm im Editor anzuzeigen und zu bearbeiten.
[New]	zoom	Wählen Sie [New], um einen neuen Programmnamen einzugeben und zum Editor zu wechseln, um Ihr neues Programm einzugeben. Wählen Sie im Bildschirm [New] die Option [Types] (drücken Sie hierzu [zoom]), um einen Programmtyp auszuwählen. Durch die Auswahl eines Programmtyps wird eine Vorlage von Import-Anweisungen und häufig verwendeten Funktionen und Methoden für diese Aktivität in Ihr neues Programm eingefügt.
[Shell]	trace	Wählen Sie [Shell], um die Shell-Eingabeaufforderung anzuzeigen (Python-Interpreter). Die Shell wird im aktuellen Zustand angezeigt.
[Manage]	graph	Wählen Sie [Manage], um: • die Versionsnummer anzuzeigen.

Tastenkürzel und Menüs im Dateimanager		
Menüs	**Taste**	**Beschreibung**
		• ein ausgewähltes Programm zu kopieren, zu löschen oder umzubenennen. • den Info-Bildschirm anzuzeigen. • die App zu beenden. Verwenden Sie hierzu auch 2nd [quit]

Erstellen eines neuen Programms mit den Programmtyp-Vorlagen

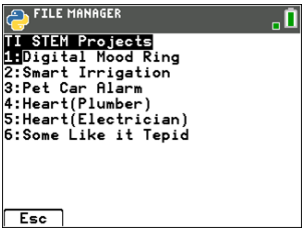


Selection pastes appropriate import statement(s) and program lines to the new program.

- Select [Types] to display Select Program Type menu.
- Import(s) displays on status bar.

Erstellen eines neuen STEM Activity-Programms mit Vorlagen

Wenn die AppVar TISTEMEN im Archiv geladen ist, wird der Menüpunkt „TI STEM Project Helpers…“ im Menü „Select Program Type“ aufgeführt. Wählen Sie die gewünschte STEM Activity-Vorlage, die Sie bei der Erstellung des neuen MINT-Programms unterstützt.



Python-Editor

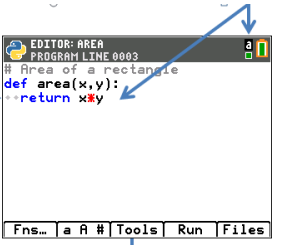
Der Python-Editor wird in einem ausgewählten Programm im Dateimanager oder in der Shell angezeigt. Der Editor zeigt Schlüsselwörter, Operatoren, Kommentare, Strings und Einzüge in Farbe an. Schnelles Einfügen gängiger Python-Schlüsselwörter und -Funktionen ist ebenso möglich wie die direkte Tastatureingabe und die Eingabe von [a A #]-Zeichen. Beim Einfügen eines Codeblocks wie z. B. if.. elif.. else bietet der Editor eine automatische Einrückung, die beim Schreiben des Programms nach Bedarf geändert werden kann.

Der Cursor befindet sich immer im Einfügemodus. Verwenden Sie 2nd und alpha, um den Cursor umzuschalten. Die Cursor-Status sind numerisch, a und A. [del] funktioniert wie eine Rücktaste und löscht ein Zeichen.

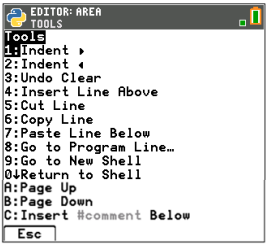
Position des Cursors in der Programmzeile.

Automatische Einrückung bei Codeblocks.

Graue Punkte als visueller Indikator für eingerückte Zeilen.



Nützliche Werkzeuge für die Bearbeitung und das Arbeiten in der Shell. Vollständige Beschreibung folgt.



Tastenkürzel und Menüs im Python-Editor		
Menüs	**Taste**	**Beschreibung**
[Fns…]	y=	Wählen Sie [Fns…], um auf Menüs mit häufig verwendeten Funktionen, Schlüsselwörtern und Operatoren zuzugreifen. Sie können so auch auf ausgewählte Inhalte der Mathematik- und

<table>
<tr><th colspan="3">Tastenkürzel und Menüs im Python-Editor</th></tr>
<tr><th>Menüs</th><th>Taste</th><th>Beschreibung</th></tr>
<tr><td></td><td></td><td>Zufallsmodule zugreifen.
Hinweis: 2nd [catalog] ist auch für das schnelle Einfügen hilfreich.</td></tr>
<tr><td>[a A #]</td><td>window</td><td>Wählen Sie [a A #], um auf eine Zeichenpalette als alternative Möglichkeit zur Eingabe vieler Zeichen zuzugreifen.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Wählen Sie [Tools], um auf Funktionen zuzugreifen, die Sie bei der Bearbeitung oder Interaktion mit der Shell unterstützen.
<table>
<tr><td>1: Indent ▶</td><td>Rückt die Programmzeile nach rechts ein, der Cursor wird auf das erste Zeichen der Zeile gesetzt.</td></tr>
<tr><td>2: Indent ◀</td><td>Verkleinert die Einrückung der Programmzeile nach links. Der Cursor wird auf das erste Zeichen der Zeile gesetzt.</td></tr>
<tr><td>3: Undo Clear</td><td>Fügt die letzte gelöschte Zeile in eine neue Zeile unterhalb der Programmzeile ein, die den Cursor enthält. Der Cursor wird am Ende der eingefügten Zeile angezeigt.</td></tr>
<tr><td>4: Insert Line Above</td><td>Fügt eine Zeile oberhalb der Programmzeile ein, in der sich der Cursor befindet. Ggf. wird die Zeile eingerückt und der Einzug durch Punkte angezeigt.</td></tr>
<tr><td>5: Cut Line</td><td>Die aktuelle Programmzeile, in der sich der Cursor befindet, wird ausgeschnitten. Der Cursor wird auf der Programmzeile angezeigt, die sich unter der ausgeschnittenen Zeile befindet.</td></tr>
<tr><td>6: Copy Line</td><td>Kopiert die aktuelle Programmzeile, in der sich der Cursor befindet. Eine kopierte Programmzeile kann an der Shell-Eingabeaufforderung eingefügt werden. Siehe nachstehenden Abschnitt „Shell“.</td></tr>
<tr><td>7: Paste Line Below</td><td>Fügt die letzte gespeicherte Programmzeile in die Zeile unter der Cursorposition ein.</td></tr>
<tr><td>8: Go to</td><td>Zeigt den Cursor am Anfang der</td></tr>
</table></td></tr>
</table>

Tastenkürzel und Menüs im Python-Editor

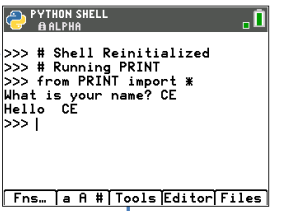
Menüs	Taste	Beschreibung	
		Program Line...	angegebenen Programmzeile an.
		9: Go to New Shell	Zeigt die neu initialisierte Shell an.
		0: Return to Shell	Zeigt die Shell im aktuellen Zustand an.
		A: Page up	Zeigt 11 Programmzeilen über der aktuellen Cursorposition an, sofern verfügbar.
		B: Page Down	Zeigt 11 Programmzeilen unter der aktuellen Cursorposition an, sofern verfügbar.
		C: Insert #comment Below	Fügt # in eine neue Zeile unterhalb der Cursorposition ein.
[Run]	trace	Wählen Sie [Run], um Ihr Programm auszuführen.	
[Files]	graph	Wählen Sie [Files], um den Dateimanager anzuzeigen.	

Python-Shell

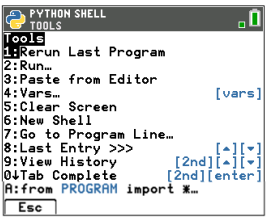
Die Python-Shell ist die Konsole, auf der Sie mit dem Python-Interpreter interagieren oder Ihre Python-Programme ausführen können. Schnelles Einfügen gängiger Python-Schlüsselwörter und -Funktionen ist ebenso möglich wie die direkte Tastatureingabe und die Eingabe von [a A #]-Zeichen. Die Shell-Eingabeaufforderung kann zum Testen einer aus dem Editor eingefügten Codezeile verwendet werden. Es können auch mehrere Codezeilen eingegeben und an einer Shell-Eingabeaufforderung >>> ausgeführt werden.

Cursorstatus-Indikator der Shell.

Die Shell wird reinitialisiert, wenn ein neues Programm ausgeführt wird.



Nützliche Werkzeuge für das Arbeiten in der Shell.
Siehe nachstehende Informationen.



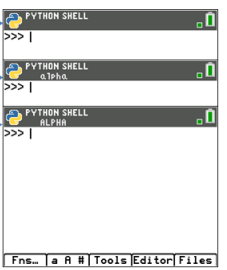
Shell-Cursorstatus

nicht alphabetisch

[2nd] [alpha], wenn ein Umschalten erforderlich ist

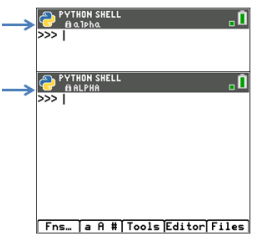
[alpha] alphabetisch

[alpha] erneut ALPHA



[2nd] [alpha] alphabetisch feststellen

erneut [alpha] ALPHA feststellen



<table>
<tr><th colspan="3">Tastenkürzel und Menüs in der Python-Shell</th></tr>
<tr><th>Menüs</th><th>Taste</th><th>Beschreibung</th></tr>
<tr><td>[Fns…]</td><td>y=</td><td>Wählen Sie [Fns...], um auf Menüs mit häufig verwendeten Funktionen, Schlüsselwörtern und Operatoren zuzugreifen. Sie können so auch auf ausgewählte Inhalte der Mathematik- und Zufallsmodule zugreifen.
Hinweis: 2nd [catalog] ist auch für das schnelle Einfügen hilfreich.</td></tr>
<tr><td>[a A #]</td><td>window</td><td>Wählen Sie [a A #], um auf eine Zeichenpalette als alternative Möglichkeit zur Eingabe vieler Zeichen zuzugreifen.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Wählen Sie [Tools], um die folgenden Menüpunkte anzuzeigen:
<table>
<tr><td>1: Rerun last program</td><td>Führt das letzte Programm, das in der Shell ausgeführt wurde, erneut aus.</td></tr>
<tr><td>2: Run…</td><td>Zeigt eine Liste der zur Ausführung in der Shell verfügbaren Python-Programme an.</td></tr>
<tr><td>3: Paste from Editor</td><td>Fügt die letzte kopierte Programmzeile aus dem Editor in die Shell-Eingabeaufforderung ein.</td></tr>
<tr><td>4: Vars…</td><td>Zeigt die Variablen des zuletzt ausgeführten Programms an. Zeigt keine programmdefinierten Variablen aus einem importierten Programm an.</td></tr>
<tr><td>5: Clear Screen</td><td>Löscht den Shell-Bildschirm. Eine neue Shell wird nicht reinitialisiert.</td></tr>
<tr><td>6: New Shell</td><td>Reinitialisiert eine neue Shell.</td></tr>
<tr><td>7: Go to Program Line…</td><td>Zeigt den Editor von der Shell aus mit dem Cursor auf der angegebenen Programmzeile an.</td></tr>
<tr><td>8: Last Entry>>> ▲▼</td><td>Zeigt während einer Shell-Sitzung die letzten Einträge an der Shell-Eingabeaufforderung an (bis zu 8).</td></tr>
<tr><td>9: View History 2nd ▲ 2nd ▼</td><td>Führen Sie auf dem Shell-Bildschirm einen Bildlauf durch, um die letzten während einer Shell-Sitzung ausgegebenen Zeilen anzuzeigen (bis zu 60).</td></tr>
</table></td></tr>
</table>

Tastenkürzel und Menüs in der Python-Shell			
Menüs	**Taste**	**Beschreibung**	
		0: Tab Complete 2nd [enter]	Zeigt die Namen der Variablen und Funktionen an, auf die in der aktuellen Shell-Sitzung zugegriffen werden kann. Wenn ein Buchstabe einer verfügbaren Variable oder Funktion eingegeben wird, drücken Sie 2nd [enter], um den Namen automatisch auszufüllen, sofern in der aktuellen Shell-Sitzung eine Übereinstimmung verfügbar ist.
		A: from PROGRAM import *…	Bei der erstmaligen Ausführung in einer Shell-Sitzung wird PROGRAM ausgeführt, und Variablen sind nur unter Verwendung von „Tab Complete“ sichtbar. Wenn das Programm in derselben Shell-Sitzung erneut ausgeführt wird, wird die Ausführung nicht ausgeführt. Dieser Befehl kann auch über 2nd [catalog] eingefügt werden.
[Editor]	trace	Wählen Sie [Editor], um den Editor mit den letzten Programmen im Editor anzuzeigen. Wenn der Editor leer ist, können Sie den Dateimanager anzeigen.	
[Files]	graph	Wählen Sie [Files], um den Dateimanager anzuzeigen.	

Hinweis:

- Um ein laufendes Python-Programm zu unterbrechen, z. B. wenn sich ein Programm in einer Endlosschleife befindet, drücken Sie on. Alternativ können Sie **[Tools]** (zoom) > **6:New Shell** drücken, um ein laufendes Programm anzuhalten.
- Wenn Sie das Modul ti_plotlib verwenden, um den Plotting-Bereich in der Shell zu zeichnen, drücken Sie clear, um den Plot zu löschen und zur Shell-Eingabeaufforderung zurückzukehren.

Ausführungsfehler: mit Shell >Tools zur Programmzeile gehen

Die TI-Python-Umgebung zeigt Python-Fehlermeldungen in der Shell an, wenn der Code ausgeführt wird. Wenn bei der Ausführung eines Programms ein Fehler angezeigt wird, wird eine Programmzeilennummer angezeigt. Verwenden Sie **Shell>Tools 7:Go to Program Line…** Geben Sie die Zeilennummer ein und drücken Sie **[OK]**. Der Cursor wird im Editor auf dem ersten Zeichen der entsprechenden Programmzeile angezeigt. Die Programmzeilennummer wird in der zweiten Zeile der Statusleiste im Editor angezeigt.

Eingaben – Tastatur, Katalog, Zeichentabelle und Menüs

Tipps für die schnelle Eingabe

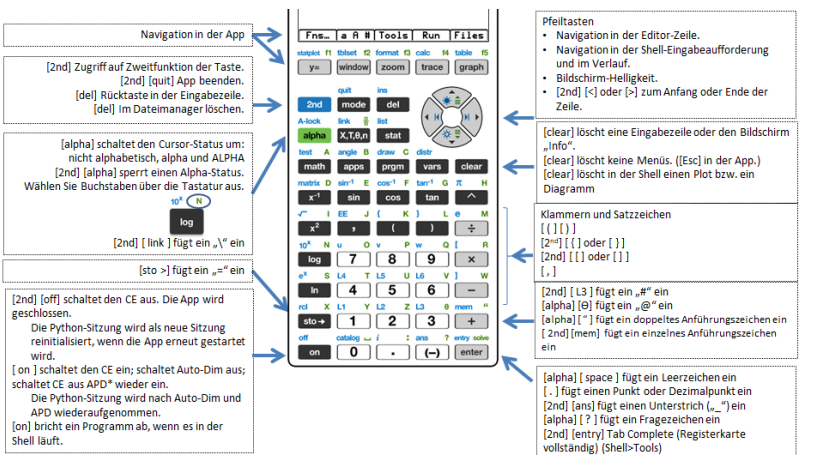
Verwenden der Tastatur, des Katalogs und der Menüs [a A #] und Fns…

Wenn Sie Code im Editor oder in der Shell eingeben, verwenden Sie die folgenden Eingabemethoden, um diesen schnell in die Bearbeitungszeile einzufügen.

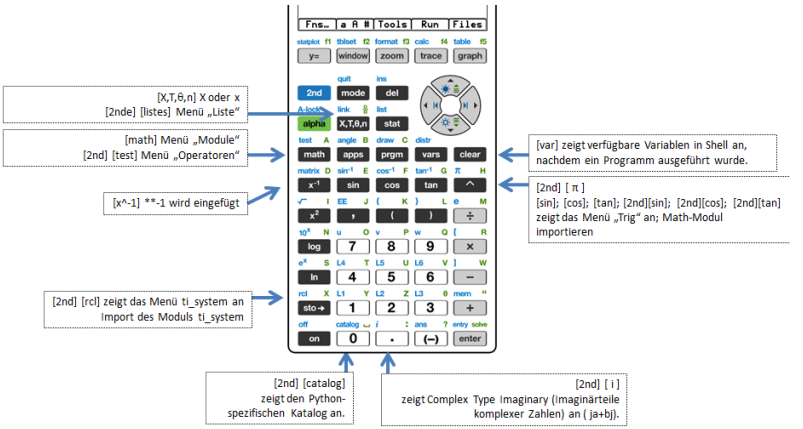
Tastatur

Wenn die Python-Anwendung läuft, ist die Tastatur so gestaltet, dass die entsprechenden Python-Operationen eingefügt oder Menüs geöffnet werden können, die für die einfache Eingabe von Funktionen, Schlüsselwörtern, Methoden, Operatoren usw. konzipiert sind. Wenn Sie 2nd und alpha drücken, erhalten Sie wie im Betriebssystem Zugriff auf die zweite und dritte Funktion einer Taste.

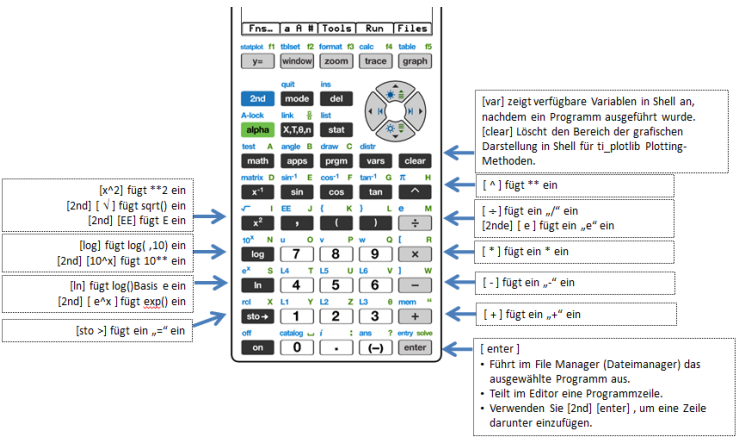
Navigation, Bearbeiten und Sonderzeichen in der Python-App nach Tastaturzeilen



Spezifische Tastenbetätigungen für Menüs und Funktionen in der Python-App nach Tastaturzeilen



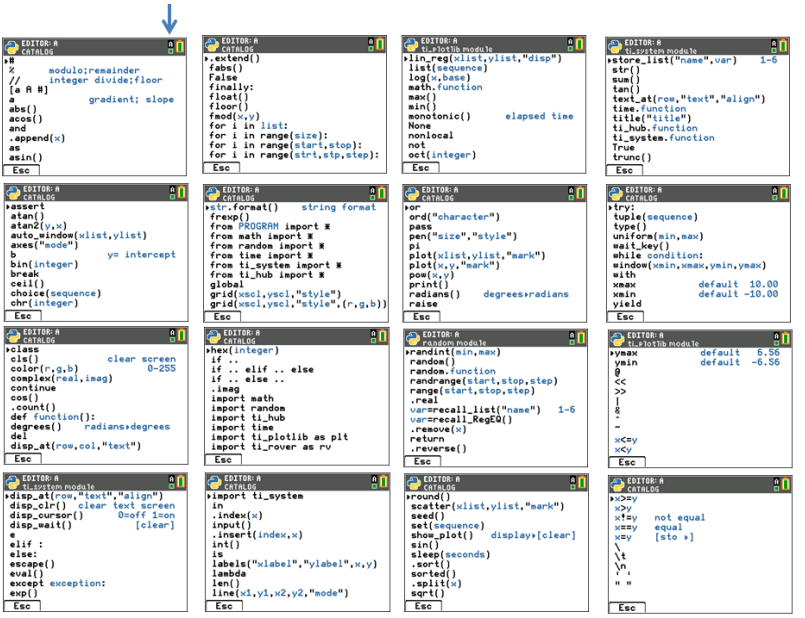
Spezifische Tastenbetätigungen für Menüs und Funktionen in der Python-App nach Tastaturzeilen (Forts.)



Katalog

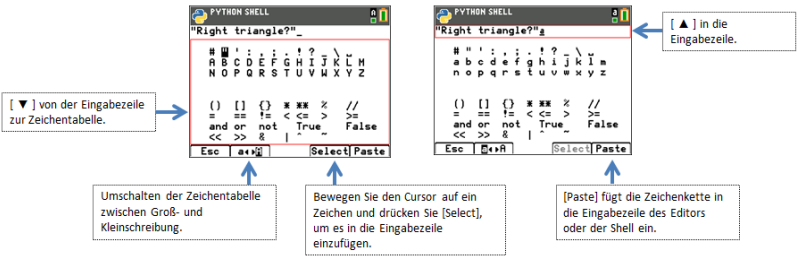
Wenn die Python-App ausgeführt wird, kann über 2nd [catalog] eine Liste häufig verwendeter Trennzeichen, Schlüsselwörter, Funktionen und Operatoren zum schnellen Einfügen in eine Eingabezeile angezeigt werden.
2nd [catalog] ist nur im Editor und in der Shell verfügbar. Eine ausführlichere Beschreibung der einzelnen Katalogartikel finden Sie im Referenz-Leitfaden. Um vom Anfang des Katalog-Menüs zum Ende zu gelangen und dann weiter nach oben zu navigieren, drücken Sie ▲.

Wählen Sie im Katalog alpha und eine Buchstabentaste, um die Liste ab diesem Anfangsbuchstaben anzuzeigen.



[a A #] Zeichentabelle

Die [a A #]-Verknüpfungs-Registerkarte zu einer Zeichenpalette ist eine praktische Funktion zur Eingabe von Zeichenketten im Editor oder in der Shell.



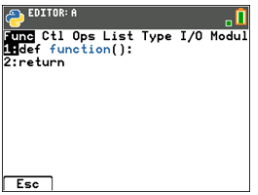
Hinweis: Wenn sich der Cursor-Fokus in der [a A #]-Eingabezeile befindet, sind ausgewählte Tasten der Tastatur nicht verfügbar. Wenn sich der Fokus in der Zeichentabelle befindet, ist das Tastenfeld eingeschränkt.

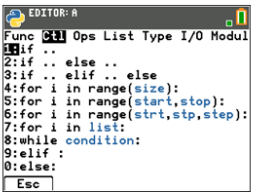
[Fns…] Menüs

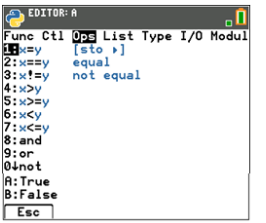
Die Verknüpfungs-Registerkarte [Fns…] zeigt Menüs mit häufig verwendeten Python-Funktionen, Schlüsselwörtern und Operatoren an. Die Menüs bieten auch Zugang zu den ausgewählten Funktionen und Konstanten aus den Modulen math und random. Anstatt dass Sie Zeichen für Zeichen über die Tastatur eingeben, bieten diese Menüs eine schnelle Möglichkeit zum Einfügen in Editor oder Shell. Drücken Sie [Fns…], wenn Sie sich im Editor oder in der Shell befinden. Alternative Eingabemethoden finden Sie auch unter Katalog und Tastatur.

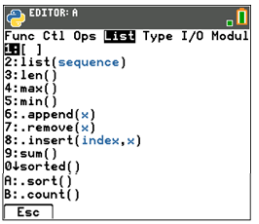
Untermenüs von Funktionen und Modulen

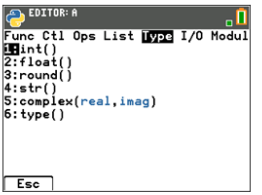
Integriert, Operatoren und Schlüsselwörter

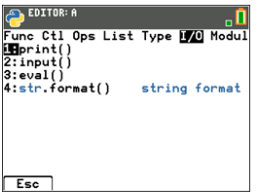










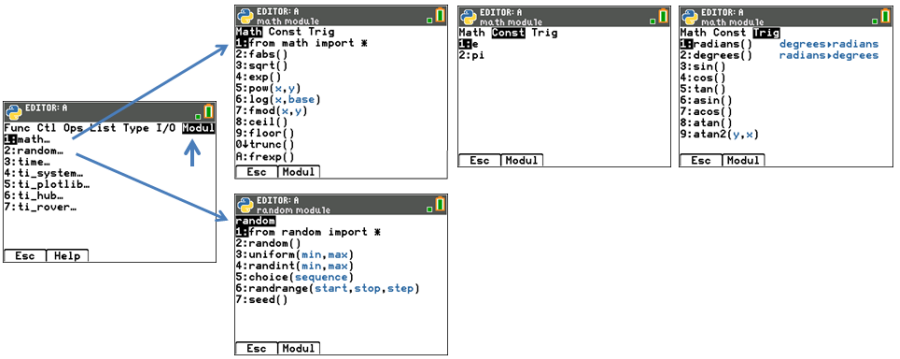


Modul-Untermenüs

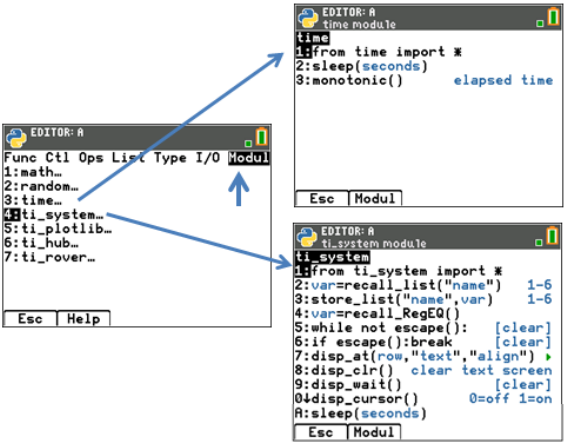
Wenn Sie eine Python-Funktion oder -Konstante aus einem Modul verwenden, verwenden Sie immer eine Import-Anweisung, um den Ort der Funktion, Methode oder Konstante im Modul anzugeben.

Siehe Was ist die Python-Programmierumgebung?

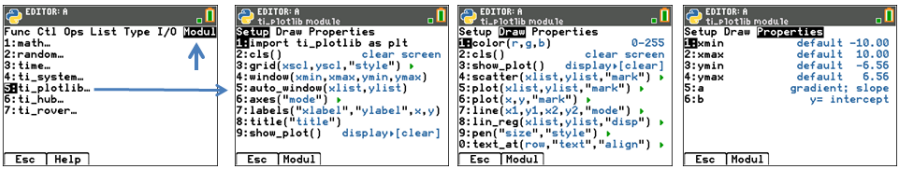
[Fns…]>Modul: math and random modules



[Fns…]>Modul: time and ti_system modules



[Fns…]>Modul: ti_plotlib



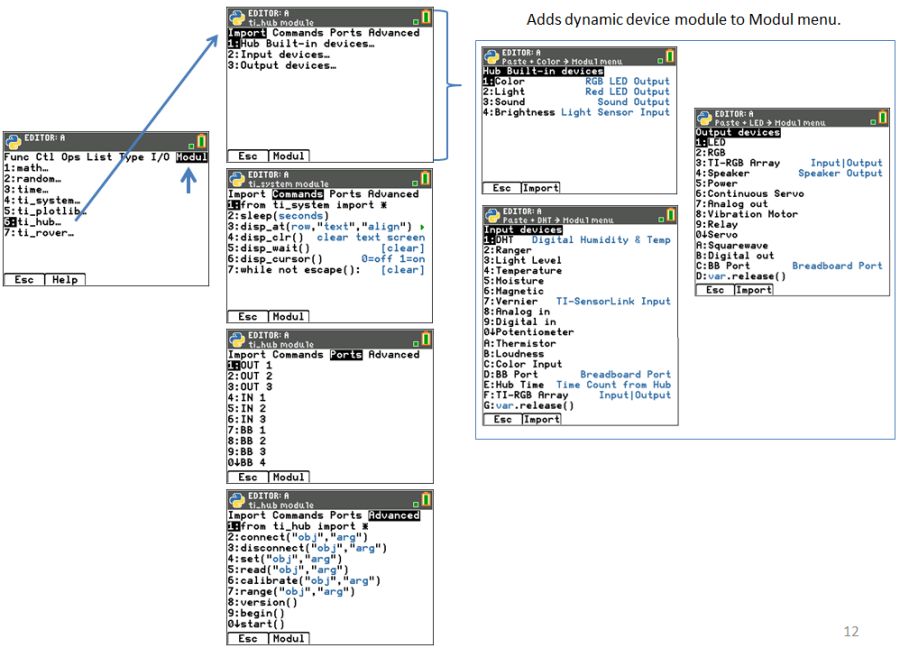
Wichtiger Hinweis für die grafische Darstellung:

- Um die erwarteten Ergebnisse zu gewährleisten, muss die Reihenfolge der Programmzeilen für das Plotting der Reihenfolge im Setup-Menü folgen.
- Die grafische Anzeige wird angezeigt, wenn plt.show_plot() am Ende der Plotting-Objekte in einem Programm ausgeführt wird. Um den Plotting-Bereich in der Shell zu löschen, drücken Sie [clear].
- Die Ausführung eines zweiten Programms, das davon ausgeht, dass die Standardwerte innerhalb derselben Shell-Umgebung festgelegt sind, führt im Allgemeinen zu unerwartetem Verhalten, beispielsweise bei den Farb- oder anderen Standardargument-Einstellungen. Bearbeiten Sie Programme mit erwarteten Argumentwerten oder reinitialisieren Sie die Shell, bevor Sie ein anderes Plotting-Programm ausführen.

[Fns…]>Modul: ti_hub module

ti_hub-Methoden sind nicht im Katalog und somit auch nicht im Referenz-Leitfaden aufgeführt. Bitte benutzen Sie für Argumente und Argumentvorgaben oder Details zu erlaubten Werten die Bildschirminformationen in den Menüs. Weitere Informationen zur Python-Programmierung für
TI-Innovator™ Hub und TI-Innovator™ Rover stehen unter education.ti.com zur Verfügung.

Hinweis: TI-Innovator™ Hub sollte angeschlossen sein, wenn Sie Ihre Python-Programme ausführen.



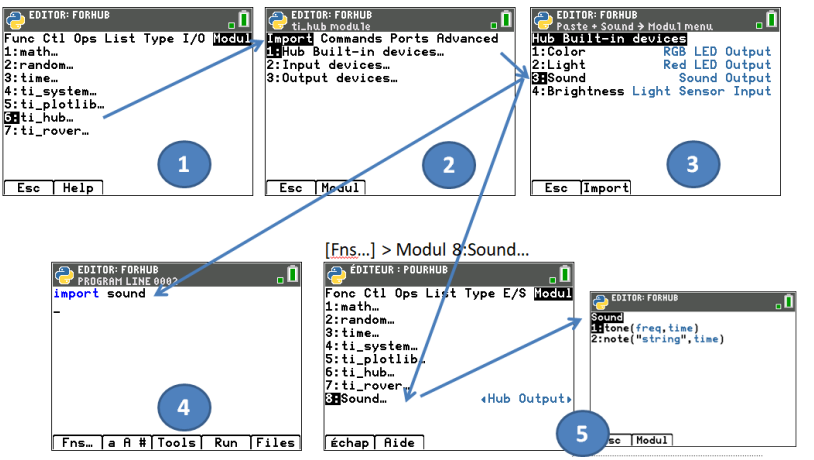
ti_hub-Modul – Import zum Editor hinzufügen und ti_hub-Sensormodul zum Modul-Menü hinzufügen

Beispiel: Ton importieren

Gehen Sie wie folgt vor, um TI-Innovator™ Sensormethoden aus dem Editor in Ihr Python-Programm zu importieren:

1. Wählen Sie **[Fns…] > Modul 6:ti_hub**
2. Wählen Sie das ti_hub-Importmenü. Wählen Sie einen Sensortyp aus Built-in, Input und Output.
3. Wählen Sie einen Sensor.
4. Eine Import-Anweisung wird in den Editor eingefügt und das Sensormodul steht in **[Fns…] > Modul** zur Verfügung, wenn Sie von Ihrem Programm zu diesem Menü zurückkehren.
5. Wählen Sie **[Fns…] > Modul 8:Sound…**, um geeignete Methoden für diesen Sensor einzufügen.

[Fns…]>Modul 6:ti_hub



Hinweis: „Brightns“ ist ein „integriertes“ Objekt in TI-Innovator Hub.

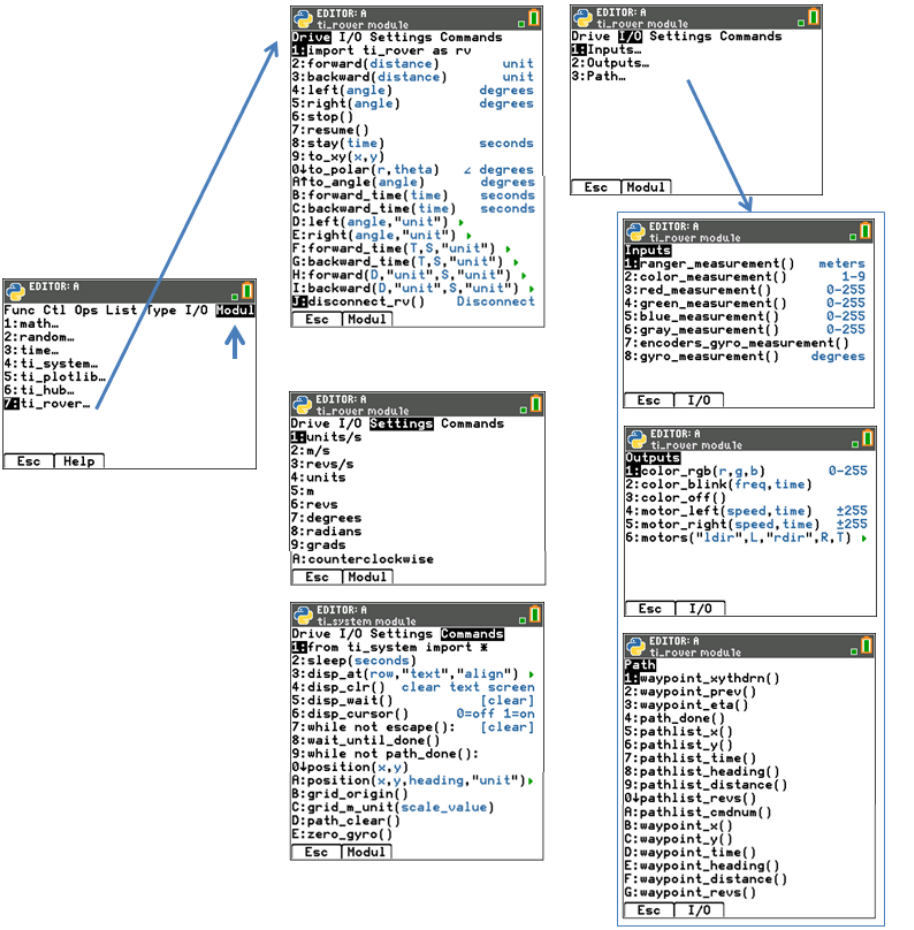
Wenn Sie die Anweisung „import brightns“ verwenden, geben Sie „brightns.range (0,100)“ ein, um sicherzustellen, dass zu Beginn der Programmausführung der korrekten Standardbereich eingestellt ist.

Beispiel:

```
import brightns
brightns.range(0,100)
b=brightns.measurement()
print(b)
```

---

ti_rover-Methoden sind nicht im Katalog und somit auch nicht im Referenz-Leitfaden aufgeführt. Bitte benutzen Sie für Argumente und Argumentvorgaben oder Details zu erlaubten Werten die Bildschirminformationen in den Menüs. Weitere Informationen zur Python-Programmierung für
TI-Innovator™ Hub und TI-Innovator™ Rover stehen unter education.ti.com zur Verfügung.



Hinweise:

- In der TI-Python-Programmierung müssen Sie keine Methoden zum Verbinden und Trennen von TI-Innovator™ Rover einschließen. Die TI-Innovator™ Rover-Python-Methoden schließen das Verbinden und Trennen ohne zusätzliche Methoden ein.

Dies ist ein kleiner Unterschied zur Programmierung des TI-Innovator™ Rovers in TI-Basic.

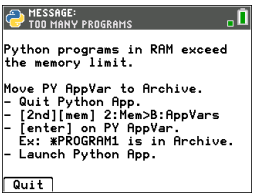
- rv.stop() wird als Pause ausgeführt, anschließend werden die Rover-Bewegungen in der Warteschlange über „resume“ fortgeführt. Wenn nach rv.stop() ein weiterer Bewegungsbefehl ausgeführt wird, wird die Bewegungswarteschlange geleert. Auch dies ist ein kleiner Unterschied zur Programmierung des TI-Innovator™ Rovers in TI-Basic.

Meldungen in der Python-App

Während einer Python-Sitzung können mehrere Meldungen angezeigt werden. Einige ausgewählte Meldungen sind in der Tabelle aufgeführt. Bitte folgen Sie den Anweisungen auf dem Bildschirm und navigieren Sie je nach Bedarf mit [Quit], [Esc] oder [Ok].

Speicherverwaltung

Der verfügbare Speicher der Python-Umgebung beträgt maximal 100 Python-Programme (PY AppVars) oder 50K Speicherplatz. Die Module, die in dieser Python-Version mit der App gebündelt sind, teilen sich mit allen Dateien den gleichen Speicherplatz.



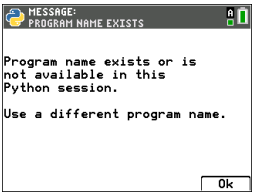
Verwenden Sie [2nd] [quit], um die App zu beenden

Sie werden aufgefordert, zu bestätigen, dass Sie die App beenden möchten. Durch Beenden der App wird die Python-Sitzung gestoppt. Wenn Sie die Python-App erneut ausführen, werden Ihre Python AppVar-Programme und Module synchronisiert. Die Shell wird reinitialisiert.

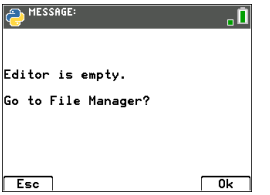
Sie drücken im Dateimanager bei einem ausgewählten Python-Programm auf del oder Sie wählen in **File Manager>Manage 2:Delete Program...**.

Im angezeigten Dialogfeld können Sie wählen, das Programm zu löschen oder zum Dateimanager zurückzukehren.

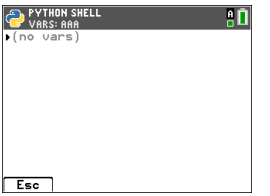
Sie versuchen, ein neues Python-Programm zu erstellen oder ein Python-Programm zu duplizieren, das entweder im RAM oder im Archiv Ihres CE vorhanden oder für den Prüfungsmodus deaktiviert ist. Bitte geben Sie einen anderen Namen ein.



Sie versuchen, von der Shell zum Editor zu navigieren, der Editor ist jedoch leer. Bitte wählen Sie eine geeignete Option für Ihre Arbeit aus.



Wenn Sie ein Python-Programm ausführen, werden die definierten Variablen des zuletzt ausgeführten Programms im **Menü Shell>Tools> 4:Vars...** aufgelistet und stehen für die Verwendung in der Shell zur Verfügung. Wenn keine Variablen angezeigt werden, müssen Sie Ihr Programm möglicherweise erneut ausführen.



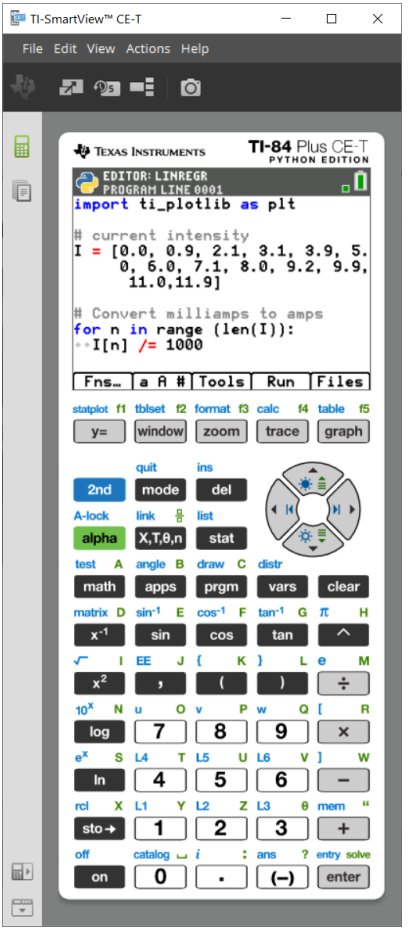
Verwendung von TI-SmartView™ CE-T und der Python-Umgebung

Dieses Handbuch bezieht sich auf die neueste Version von TI-SmartView™ CE-T. Aktualisieren Sie TI-SmartView™ CE-T unter education.ti.com/84cetupdate auf die neueste Version.

Das Update enthält das neueste TI-84 Plus CE-T *Python Edition* Emulator-Betriebssystem mit der neuesten Python-App. Es enthält die aktualisierten Module von time, ti_system, ti_plotlib, ti_rover* und ti_hub*.

Führen Sie die Python-App im TI-84 Plus CE-T *Python Edition* Emulator aus.

- Die Python-App bietet:
  - Dateimanager
  - Editor
  - Ausführung Ihres Python-Programms in der Shell*



Hub-/Rover-Programme

- Wenn im CE-Emulator die Python-App läuft, können Sie ti_hub/ti_rover-Python-Programme erstellen.

  *** Hinweis:** Es besteht keine Konnektivität zwischen TI-SmartView™ CE-T und TI-Innovator™ Hub bzw. TI-Innovator™ Rover. Programme können erstellt und dann auf dem CE-Taschenrechner ausgeführt werden.

- Beenden Sie die Python-App, um die Übertragung der Python-AppVar(s) aus dem Emulator vorzubereiten. Beim nächsten Schritt sollte der Emulator nicht damit beschäftigt sein, eine App oder ein Programm auszuführen.

– Wechseln Sie zum Arbeitsbereich Emulator-Explorer und senden Sie das/die Programm(e) an den Computer.
– Verwenden Sie für die TI-Innovator™ Hub/TI-Innovator™ Rover-Erfahrung TI Connect™ CE, um die Python AppVars vom Computer an den CE-Taschenrechner zu senden.

Hinweis: Um ein laufendes Python-Programm in der Shell zu unterbrechen, z. B. wenn sich ein Programm in einer Endlosschleife befindet, drücken Sie **[on]**. Alternativ können Sie **[Tools] [zoom] > 6:New Shell** drücken, um ein laufendes Programm anzuhalten.

Zur Erinnerung: Für alle Computer/TI-Python-Umgebungen: Nach dem Erstellen eines Python-Programms in einer Python-Entwicklungsumgebung auf dem Computer überprüfen Sie bitte die Ausführung Ihres Programms auf dem Rechner/Emulator in der TI-Python-Umgebung. Nehmen Sie ggf. Änderungen am Programm vor.

Remote-Tastatur der SmartPad CE-App

– Wenn die SmartPad CE-App auf Ihrem angeschlossenen CE ausgeführt wird, verhält sich dieses wie eine Remote-*Tastatur* einschließlich der speziellen Tastaturbelegung, die beim Ausführen der Python-App angeboten wird.

Arbeitsbereich Emulator-Explorer

– Bitte beenden Sie die Python-App, damit der Emulator nicht beschäftigt ist, wenn Sie auf den vollen Funktionsumfang des Arbeitsbereichs Emulator-Explorer zugreifen.
– Umwandlungen zwischen program.py und PY AppVar sind erlaubt. Dies ähnelt dem Einsatz von
TI Connect™ CE beim Senden von Programmen an den angeschlossenen CE-Taschenrechner.
– Zum Senden einer program.py-Datei, die in einer anderen Python-Umgebung erstellt wurde, muss die PY AppVar bearbeitet werden, damit sie in TI-Python wie erwartet läuft. Verwenden Sie den Editor der Python-App, um die erforderlichen Änderungen für die einzelnen Module wie ti_plotlib, ti_system, ti_hub und ti_rover vorzunehmen.

Datenimport-Assistent

– *.csv-Datendateien , die wie im Dialogfeld des Assistenten angegeben formatiert sind, werden in CE-Listenvariablen konvertiert. Anschließend können Methoden in ti_system verwendet werden, um Listen zwischen dem Emulator CE-Betriebssystem und der Python-App auszutauschen. Diese Funktion ähnelt dem Datenimport-Assistenten in TI Connect™ CE.
- Wenn Dezimalzahlen in der *.csv-Datei mit Verwendung eines Kommas dargestellt werden, wird die Datei nicht vom Datenimport-Assistenten konvertiert. Bitte überprüfen Sie die Zahlenformatierung Ihres Computer-Betriebssystems und konvertieren Sie das *.csv-Format, um die Dezimalpunktdarstellung zu verwenden. Beispiel: Die CE-Taschenrechnerliste und der Matrixeditor verwenden das Zahlenformat 12.34 und nicht 12,34.

Verwenden von TI Connect™ CE zum Konvertieren von Python-Programmen

Bitte aktualisieren Sie auf TI Connect™ CE, um die neuesten Funktionen zu erhalten, einschließlich der Konvertierung von *.py-Programmen in eine PY AppVar im CE-Taschenrechnerdateiformat.

Detailliertere Informationen zum CE Taschenrechner, TI-SmartView™ CE-T und TI Connect CE finden Sie im TI-84 Plus CE-T e-Guide.

Was ist die Python-Programmierumgebung?

TI-Python basiert auf CircuitPython, einer Python-Variante, die für kleine Mikrocontroller entwickelt wurde. Die ursprüngliche CircuitPython-Implementierung wurde für die Verwendung durch TI angepasst.

Die interne Speicherung von Zahlen zur Berechnung in dieser Variante von Circuit Python erfolgt in binären Fließkommazahlen mit begrenzter Genauigkeit und kann daher nicht alle möglichen Dezimalwerte exakt darstellen. Die bei der Speicherung dieser Werte auftretenden Abweichungen von den tatsächlichen Dezimaldarstellungen können bei nachfolgenden Berechnungen zu unerwarteten Ergebnissen führen.

- **Bei Fließkommazahlen** – Es werden bis zu 16 signifikante Genauigkeitsziffern angezeigt. Intern werden die Werte mit einer Genauigkeit von 53 Bit gespeichert, was in etwa 15-16 Dezimalstellen entspricht.
- **Bei Ganzzahlen** – Die Größe von Ganzzahlen ist nur durch den zum Zeitpunkt der Berechnungen verfügbaren Speicher begrenzt.

Im TI-84 Plus CE-T Python Edition enthaltene Module

- Built-in (integriert)
- math module
- random module
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Hinweis: Wenn Sie Python-Programme besitzen, die in anderen Python-Entwicklungsumgebungen erstellt wurden, passen Sie Ihr(e) Programm(e) bitte an die TI-Python-Lösung an. Die Module können in einem Programm andere Methoden, Argumente und Methoden-Reihenfolgen verwenden als die Module ti_system, ti_ plotlib, ti_hub und ti_rover modules. Achten Sie bei der Verwendung aller Python-Versionen und Python-Module grundsätzlich auf Kompatibilität.

Beim Übertragen von Python-Programmen von einer Nicht-TI-Plattform auf eine TI-Plattform ODER von einem TI-Produkt auf ein anderes:

- Python-Programme, die Kernsprachenfunktionen und Standardbibliotheken (math, random usw.) verwenden, können ohne Änderungen importiert oder exportiert werden.

  Hinweis: Die Begrenzung der Listenlänge beträgt 100 Elemente.

- Programme, die plattformspezifische Bibliotheken – matplotlib (für PC), ti_ plotlib,
  ti_system, ti_hub usw. – für TI-Plattformen verwenden, müssen bearbeitet werden, bevor sie auf einer anderen Plattform ausgeführt werden können.

- Dies kann sogar zwischen TI-Plattformen erforderlich sein.

Wie bei jeder Python-Version müssen Sie Importe, wie z. B. aus math import *, einbinden, um alle Funktionen, Methoden oder Konstanten verwenden zu können, die im math-Modul enthalten sind. Um beispielsweise die Funktion cos() auszuführen, importieren Sie mit „import“ das math-Modul zur Verwendung.

Siehe KATALOG-Liste.

Beispiel:

```
>>>from math import *
>>>cos(0)
1.0
```

Alternatives Beispiel:

```
>>>import math
>>>math.cos(0)
1.0
```

Verfügbare Module können in der Shell mit folgendem Befehl angezeigt werden:

```
>>> help("modules")
__main__ sys gc
random time array
math builtins collections
```

Der Inhalt von Modulen kann in der Shell wie gezeigt mit „import module“ und „dir (module)“ angezeigt werden.

Nicht alle Modulinhalte werden in den Schnelleinfüge-Menüs wie z. B. [Fns…] oder 2nd [catalog] angezeigt.

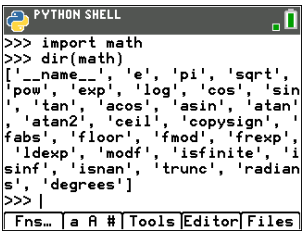
Inhalte ausgewählter Module und Schlüsselwörter

Eine Liste der in dieser Version enthaltenen Module finden Sie in:

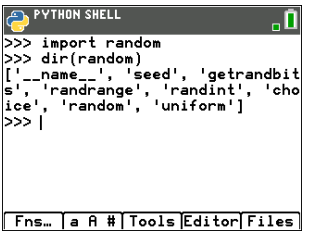
Anhang: Ausgewählte Inhalte der in TI-Python integrierten Funktionen, Schlüsselwörter und Module

Zur Erinnerung: Für alle Computer/TI-Python-Umgebungen: Nach dem Erstellen eines Python-Programms auf dem Computer überprüfen Sie bitte die Ausführung Ihres Programms auf dem Taschenrechner in der TI-Python-Umgebung. Nehmen Sie ggf. Änderungen am Programm vor.

Diese Bildschirme zeigen die Modulinhalte für Mathematik und Zufall an.

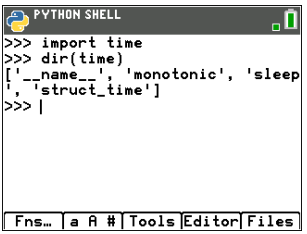


math module

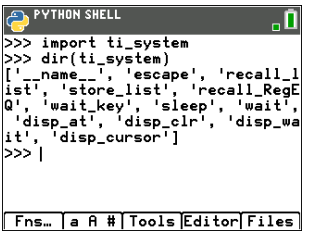


random module

Diese Bildschirme zeigen den Modulinhalt für time und ti_system an.

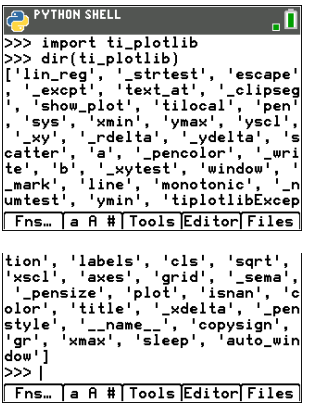


time



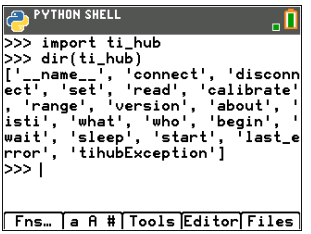
ti_system

Diese Bildschirme zeigen den Modulinhalt für ti_plotlib an.



ti_plotlib

Dieser Bildschirm zeigt den Modulinhalt für ti_hub an.



ti_hub

Diese Bildschirme zeigen den Modulinhalt für ti_rover an.

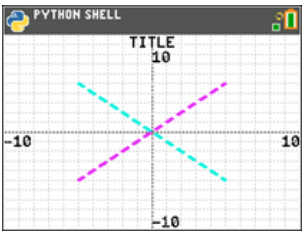
ti_rover

Beispielprogramme

Verwenden Sie die folgenden Beispielprogramme, um sich mit den Methoden aus dem Abschnitt Referenz vertraut zu machen. Diese Beispiele enthalten auch mehrere TI-Innovator™ Hub- und
TI-Innovator Rover™-Programme, die Ihnen den Einstieg in TI-Python erleichtern sollen.

COLORLIN

```
import ti_plotlib as plt
plt.cls()
plt.window(-10,10,-10,10)
plt.axes("on")
plt.grid(1,1,"dot")
plt.title("TITLE")
plt.pen("medium","solid")
plt.color(28,242,221)
plt.pen("medium","dash")
plt.line(-5,5,5,-5,"")
plt.color(224,54,243)
plt.line(-5,-5,5,5,"")
plt.show_plot()
```



Drücken Sie clear, um die Shell-Eingabeaufforderung anzuzeigen.

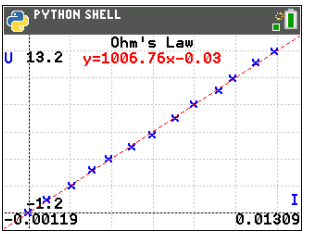
REGEQ1

Stellen Sie eine Regressionsgleichung auf, bevor Sie das Python-Programm in der Python-App ausführen. Ein Beispiel wäre, zuerst zwei Listen in das CE-Betriebssystem einzugeben. Berechnen Sie dann zum Beispiel [stat] CALC 4:LinReg(ax+b) für Ihre Listen. So wird die Regressionsgleichung im Betriebssystem in RegEQ gespeichert. Über das folgende Programm wird RegEQ in der Python-Umgebung aufgerufen.

```
# Example of recall_RegEQ()
from ti_system import *

reg=recall_RegEQ()
print(reg)
x=float(input("Input x = "))
print("RegEQ(x) = ",eval(reg))
```

LINREGR (Im CE-Bundle enthalten)

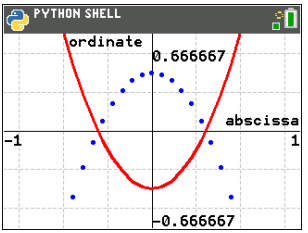
```
import ti_plotlib as plt

# current intensity
I = [0.0, 0.9, 2.1, 3.1, 3.9, 5.0, 6.0, 7.1, 8.0, 9.2, 9.9, 11.0,11.9]

# voltage
for n in range (len(I)):
I[n] /= 1000

# la tension
U = [0, 1, 2, 3.2, 4, 4.9, 5.8, 7, 8.1, 9.1, 10, 11.2, 12]

plt.cls()
plt.auto_window(I,U)
plt.pen("thin","solid")
plt.axes("on")
plt.grid(.002,2,"dot")
plt.title("Ohm's Law")
plt.color (0,0,255)
plt.labels("I","U",11,2)
plt.scatter(I,U,"x")
plt.color (255,0,0)
plt.pen("thin","dash")
plt.lin_reg(I,U,"center",2)
plt.show_plot()
plt.cls()
a=plt.a
b=plt.b
print ("a =",round(plt.a,2))
print ("b =",round(plt.b,2))
```



Drücken Sie clear, um die Shell-Eingabeaufforderung anzuzeigen.

GRAPH (Im CE-Bundle enthalten)

```
import ti_plotlib as plt
#Nach Ausführen des Programms [clear] drücken, um das Diagramm zu
löschen und zur Shell zurückzukehren.

def f(x):
  return 3*x**2-.4

def g(x):
  return -f(x)

def plot(res,xmin,xmax):
  #setup plotting area
  plt.window(xmin,xmax,xmin/1.5,xmax/1.5)
  plt.cls()
  gscale=5
  plt.grid((plt.xmax-plt.xmin)/gscale*(3/4),(plt.ymax-
plt.ymin)/gscale,"dash")
  plt.pen("thin","solid")
  plt.color(0,0,0)
  plt.axes("on")
  plt.labels("abscisse"," ordonnee",6,1)
  plt.pen("medium","solid")

# plot f(x) and g(x)
dX=(plt.xmax -plt.xmin)/res
x=plt.xmin
x0=x
  for i in range(res):
    plt.color(255,0,0)
    plt.line(x0,f(x0),x,f(x),"")
    plt.color(0,0,255)
    plt.plot(x,g(x),"o")
    x0=x
    x+=dX
  plt.show_plot()

#plot(resolution,xmin,xmax)
plot(30,-1,1)
# Ein Diagramm mit parameters(resolution,xmin,xmax) erstellen
# Nach Löschen des ersten Diagramms Taste [var] drücken. Die Funktion
plot() function ermöglicht das Ändern der Anzeigeeinstellungen
(resolution,xmin,xmax).
```

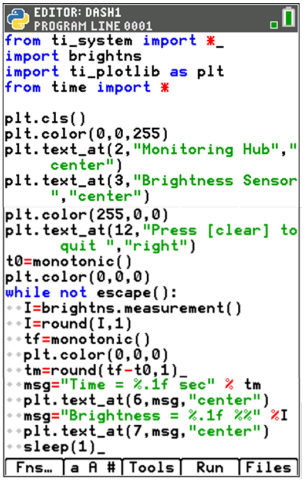


Drücken Sie clear, um die Shell-Eingabeaufforderung anzuzeigen.

DASH1 – TI-Innovator™ Hub-Beispielprogramm

Siehe: [Fns…]>Modul: ti_hub module

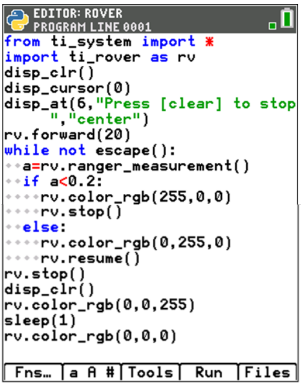
```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *
plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
while not escape():
  I=brightns.measurement()
  I=round(I,1)
  tf=monotonic()
  plt.color(0,0,0)
  tm=round(tf-t0,1)
  msg="Time = %.1f sec" % tm
  plt.text_at(6,msg,"center")
  msg="Brightness = %.1f %%" %I
  plt.text_at(7,msg,"center")
  sleep(1)
```



ROVER – TI-Innovator™ Rover-Beispielprogramm

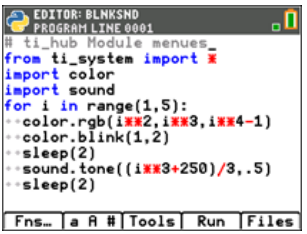
Siehe: [Fns…]>Modul ti_rover module

```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [clear] to stop","center")
rv.forward(20)
while not escape():
••a=rv.ranger_measurement()
••if a<0.2:
••••rv.color_rgb(255,0,0)
••••rv.stop()
••else:
••••rv.color_rgb(0,255,0)
••••rv.resume()
rv.stop()
disp_clr()
rv.color_rgb(0,0,255)
sleep(1)
rv.color_rgb(0,0,0)
```



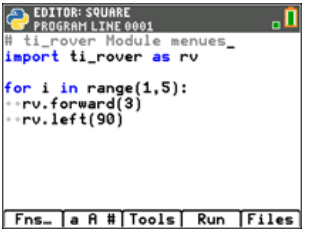
BLNKSND – TI-Innovator™ Hub-Beispielprogramm

Siehe: [Fns…]>Modul: ti_hub module



SQUARE – TI-Innovator™ Rover-Beispielprogramm

Siehe: [Fns…]>Modul ti_rover module



Referenz-Leitfaden für die TI-Python Umgebung

Die Python-App enthält Menüs mit Funktionen, Klassen, Bedienelementen, Operatoren und Schlüsselwörtern zum schnellen Einfügen in den Editor oder die Shell. Die folgende Referenztabelle enthält die Auflistung der Funktionen in [2nd] [catalog], wenn die App ausgeführt wird. Eine vollständige Aufstellung der in dieser Version von Python verfügbaren Funktionen, Klassen, Operatoren und Schlüsselwörter finden Sie unter „Ausgewählte Inhalte der in TI-Python integrierten Funktionen, Schlüsselwörter und Module“.

Diese Tabelle ist nicht als erschöpfende Liste der in diesem Angebot verfügbaren Python-Funktionen gedacht. Andere Funktionen, die in diesem Python-Angebot unterstützt werden, können über die ALPHA-Tasten der Tastatur eingegeben werden.

Die meisten Beispiele in dieser Tabelle werden an der Shell-Eingabeaufforderung (>>>) ausgeführt.

KATALOG-Liste

Alphabetische Auflistung

- A
- B
- C
- D
- E
- F
- G
- H
- I
- L
- M
- N
- O
- P
- R
- S
- T
- U
- W
- X
- Y
- Sonderzeichen

#

Trennzeichen

2nd [catalog]

Syntax: #Ihr Kommentar zu Ihrem Programm.

[a A #]

Beschreibung: In Python beginnt ein Kommentar mit dem Hashtag-Zeichen (#) und reicht bis zum Ende der Zeile.

Beispiel:

```
#Kurze Erklärung des Codes.
```

%

Operator

2nd [catalog]

Syntax: x%y oder x % y

Beschreibung: Gibt den Rest von x/y zurück. Wird bevorzugt verwendet, wenn x und y ganze Zahlen sind.

[a A #]

Beispiel:

```
>>>57%2
1
```

Siehe auch fmod(x,y).

//

Operator

2nd [catalog]

Syntax: x//y oder x // y

[a A #]

Beschreibung: Gibt die floor-Division (Division, die jeden Rest verwirft) von x/y.

Beispiel:

```
>>>26//7
3
>>>65,4//3
21.0
```

[a A #]

Beschreibung: Starten der [a A #]-Zeichenpalette.

Diese enthält Zeichen mit Akzenten, wie z. B. ç à â è é ê ë î ï ô ö ù û

Die [a A #]-Verknüpfung wird im Editor oder in der Shell bei window angezeigt

a gradient; slope

Modul: ti_plotlib

Syntax: plt.a gradient; slope

Beschreibung: Nachdem plt.linreg() in einem Programm zuletzt ausgeführt wurde, werden die berechneten Werte von slope, a, und intercept, b, in plt.a und plt.b gespeichert.

Standardwerte: = 0,0

Beispiel:

Siehe Beispielprogramm: LINREGR.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Properties 5:a

Importbefehle finden Sie unter 2nd [catalog] oder im ti_plotlib Setup-Menü.

abs()

Modul: Built-in

Syntax: abs(x)

Beschreibung: Gibt den Absolutwert einer Zahl zurück. In dieser Version kann das Argument eine Ganzzahl oder eine Fließkommazahl sein.

Beispiel:

```
>>>abs(-35.4)
35.4
```

2nd [catalog]

Hinweis: fabs() ist eine Funktion im math-Modul.

acos()

Modul: math

Syntax: acos(x)

Beschreibung: Gibt den Bogenkosinus von x im Bogenmaß (Radian) zurück

Beispiel:

```
>>>from math import *
>>>acos(1)
0.0
```

Alternatives Beispiel: [Tools] > 6:New Shell

```
>>>import math
>>>math.acos(1)
0.0
```

[sin] 7:acos()

[2nd] [catalog]

[Fns…] Modul 1:math… > Trig 7:acos()

Importbefehle finden Sie unter [2nd] [catalog].

and

Schlüsselwort

Syntax: x and y

Beschreibung: Kann True oder False zurückgeben. Gibt "x" zurück, wenn "x" False ist, anderenfalls "y". Wird mit Leerzeichen vor und hinter dem „and" eingefügt. Kann nach Bedarf bearbeitet werden.

Beispiel:

```
>>>2<5 and 5<10
True
>>>2<5 and 15<10
False
>>>{1} and 3
3
>>>0 and 5 < 10
0
```

[2nd] [test]
Ops 8:and

[Fns…] > Ops 8:and

[2nd] [catalog]

[a A #]

.append(x)

Modul: Built-in

Syntax: listname.append(item)

Beschreibung: Die Methode append() hängt ein Element an eine Liste an.

Beispiel:

```
>>>listA = [2,4,6,8]
>>>listA.append(10)
>>>print(listA)
[2,4,6,8,10]
```

2nd [list]
List
6: .append(x)

2nd [catalog]

[Fns…] > List
6:.append(x)

as

Schlüsselwort

Beschreibung: Verwenden Sie „as“, um beim Importieren eines Moduls einen Alias zu erstellen. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

2nd [catalog]

asin()

Modul: math

Syntax: asin()

Beschreibung: Gibt den Bogensinus von x im Bogenmaß (Radian) zurück

Beispiel:

```
>>>from math import *
>>>asin(1)
1.570796326794897
```

Alternatives Beispiel:

```
>>>import math
>>>math.asin(1)
1.570796326794897
```

sin 6:asin()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
6:asin()

Importbefehle finden Sie unter 2nd [catalog].

assert

Schlüsselwort [2nd] [catalog]

Beschreibung: Verwenden Sie „assert“, um eine Bedingung in Ihrem Code zu testen. Ergibt „None“; anderenfalls wird bei der Ausführung des Programms ein AssertionError angezeigt.

atan()

Modul: math [sin] 8:atan()

Syntax: atan(x)

Beschreibung: Gibt die Bogentangente von x im Bogenmaß (Radian) zurück. [Fns…]>Modul 1:math… > Trig 8 :atan()

Beispiel:

```
>>>from math import *
>>>atan(1)*4
3.141592653589793
```

[2nd] [catalog]

Alternatives Beispiel:

```
>>>import math
>>>math.atan(1)*4
3.141592653589793
```

Importbefehle finden Sie unter [2nd] [catalog].

atan2(y,x)

Modul: math [sin] 9:atan2()

Syntax: atan2(y,x)

Beschreibung: Gibt die Bogentangente von y/x im Bogenmaß (Radian) zurück. Das Ergebnis wird in [-pi, pi] angegeben. [Fns…] > Modul 1:math… > Trig 9:atan2()

Beispiel:

```
>>>from math import *
>>>atan2(pi,2)
1.003884821853887
```

[2nd] [catalog]

Alternatives Beispiel:

```
>>>import math
>>>math.atan2(math.pi,2)
1.003884821853887
```

Importbefehle finden Sie unter [2nd] [catalog].

auto_window(xlist,ylist)

Modul: ti_plotlib

Syntax: plt.auto_window(xlist,ylist)

Beschreibung: Autoskaliert das Plotting-Fenster so, dass es den Datenbereichen in xlist und ylist entspricht, die im Programm vor auto_window() angegeben wurden.

Hinweis: max(list) - min(list) > 0.00001

Beispiel:

Siehe Beispielprogramm: LINREGR.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 5:auto_window ()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

axes("mode")

Modul: ti_plotlib

Syntax: plt.axes("mode")

Beschreibung: Zeigt Achsen im angegebenen Fenster im Plotting-Bereich an.

Argument:

Optionen im Argument "mode":

"off"	keine Achsen
"on"	Achsen+Achsenbezeichnungen
"axes"	nur Achsen
"window"	nur Fensterbeschriftungen

plt.axes() verwendet die aktuelle Zeichenstift-Farbeinstellung. Um sicherzustellen, dass plt.axes() immer wie erwartet gezeichnet werden, verwenden Sie plt.color() VOR plt.axes(), damit die gewünschten Farben verwendet werden.

Beispiel:

Siehe Beispielprogramm LINREGR.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 6:axes()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

b y= intercept

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.b y= intercept

Beschreibung: Nachdem plt.linreg() in einem Programm ausgeführt wurde, werden die berechneten Werte von slope, a, und intercept, b, in plt.a und plt.b gespeichert.

[Fns...]>Modul oder math 5:ti_plotlib...> Properties 6:b

Standardwerte: = 0,0

Beispiel:

Siehe Beispielprogramm LINREGR.

Importbefehle finden Sie unter 2nd [catalog] oder im ti_plotlib Setup-Menü.

bin(integer)

Modul: Built-in

2nd [catalog]

Syntax: bin(integer)

Beschreibung: Zeigt das Binärformat des Arguments „integer“ an.

Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> bin(2)
'0b10'
>>> bin(4)
'0b100'
```

break

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „break“, um aus einer for- oder while-Schleife auszubrechen.

ceil()

Modul: math

Syntax: ceil(x)

Beschreibung: Gibt die kleinste ganze Zahl zurück, die größer oder gleich x ist.

Beispiel:

```
>>>from math import *
>>>ceil(34.46)
35
>>>ceil(678)
678
```

[math] Modul
1:math... Math
8:ceil()

[2nd] [catalog]

[Fns...] Modul
1:math...Math
8:ceil()

Importbefehle finden Sie unter [2nd] [catalog].

choice(sequence)

Modul: random

Syntax: choice(sequence)

Beschreibung: Liefert ein Zufallselement aus einer nicht leeren Folge.

Beispiel:

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA) #Ihr Ergebnis kann abweichen.
4
```

[math] Modul
2:random...
Random
5:choice(sequence)

[2nd] [catalog]

[Fns...] Modul
2:random...
Random
5:choice(sequence)

Importbefehle finden Sie unter [2nd] [catalog].

chr(integer)

Modul: Built-in

2nd [catalog]

Syntax: chr(integer)

Beschreibung: Gibt eine Zeichenfolge aus einer ganzzahligen Eingabe zurück, die das Unicode-Zeichen darstellt.

Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> char(40)
'('
>>> char(35)
'#'
```

class

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „class“, um eine Klasse zu erstellen. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

cls() clear screen

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.cls() clear screen

Beschreibung: Löscht den Shell-Bildschirm für die grafische Darstellung. Tastenkürzel werden während des Plottings nicht angezeigt.

Hinweis: plt.cls() weist ein anderes Verhalten auf als ti_system module disp_clr().

Beispiel:

Siehe Beispielprogramm: GRAPH.

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 2:cls()

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 2:cls()

Importbefehle finden Sie unter 2nd [catalog] oder im ti_plotlib

cls() clear screen

Setup-Menü.

color(r,g,b) 0-255

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.color(r,g,b) 0-255

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 1:color()

Beschreibung: Legt die Farbe für alle folgenden Grafiken/Plots fest. Die (r,g,b)-Werte müssen mit 0-255 angegeben werden. Die angegebene Farbe wird in der Diagrammdarstellung verwendet, bis color() mit einer anderen Farbe erneut ausgeführt wird.

Die Standardfarbe ist nach Import von ti_plotlib Schwarz.

Importbefehle finden Sie unter 2nd [catalog] oder im ti_plotlib Setup-Menü.

Beispiel:

Siehe Beispielprogramm: COLORLIN.

complex(real,imag)

Modul: Built-in

2nd [catalog]

Syntax: complex(real,imag)

[Fns...]>Type> 5:complex()

Beschreibung: Komplexer Zahlentyp.

Beispiel:

```
>>>z = complex(2, -3)
>>>print(z)
(2-3j)
>>>z = complex(1)
>>>print(z)
(1+0j)
>>>z = complex()
>>>print(z)
0j
>>>z = complex("5-9j")
>>>print(z)
(5-9j)
```

Hinweis: "1+2j" ist die korrekte Syntax. Leerzeichen wie bei "1 + 2j" zeigen eine Ausnahme (Exception) an.

continue

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „continue“ in einer for- oder while-Schleife, um die aktuelle Iteration zu beenden. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

cos()

Modul: math

sin Trig
4: cos()

Syntax: cos(x)

Beschreibung: Ergibt cos von x. Das Winkelargument wird in radians (Bogenmaß) ausgegeben.

2nd [catalog]

Beispiel:

```
>>>from math import *
>>>cos(0)
1.0
>>>cos(pi/2)
6.123233995736767e-17
```

[Fns...] Modul
1:math... > Trig
4:cos()

Alternatives Beispiel:

```
>>>import math
>>>math.cos(0)
1.0
```

Hinweis: Python zeigt die wissenschaftliche Notation mit e oder E an. Einige math-Ergebnisse werden in Python anders sein als im CE-Betriebssystem.

.count()

Modul: Built-in

2nd [catalog]

Syntax: listname.count(item)

Beschreibung: count()ist eine Methode, die die Anzahl des Vorkommens eines Elements in einem Listen-, Tupel-, Bytes-, str-, Bytearray- oder array.array-Objekt zurückgibt.

Beispiel:

```
>>>listA = [2,4,2,6,2,8,2,10]
>>>listA.count(2)
4
```

def function ():

Schlüsselwort

2nd [catalog]

Syntax: def function(var, var,...)

Beschreibung: Definieren einer Funktion in Abhängigkeit von angegebenen Variablen. Wird typischerweise mit dem Schlüsselwort „return“ verwendet.

[Fns...]>Func
1:def function():

[Fns...]>Func
2:return

Beispiel:

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

degrees()

Modul: math

sin Trig
2:degrees()

Syntax: degrees(x)

Beschreibung: Konvertiert den Winkel x von Bogenmaß nach Grad.

2nd [catalog]

Beispiel:

```
>>>from math import *
>>>degrees(pi)
180.0
>>>degrees(pi/2)
90.0
```

[Fns...]>Modul
1:math...>Trig
2:degrees()

del

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „del“, um Objekte wie Variablen, Listen usw. zu löschen.
Weitere Einzelheiten finden Sie in der Python-Dokumentation.

disp_at(row,col,"text")

Modul: ti_system

2nd [catalog]

Syntax: disp_at(row,col,"text")

2nd [rcl]
ti_system
7:disp_at()

Beschreibung: Text anzeigen, beginnend an einer bestimmten Zeilen- und Spaltenposition im Plotting-Bereich.

Wenn sich der Text am Ende des Programms befindet, wird hinter dem Text REPL mit dem Cursor >>>| angezeigt. Verwenden Sie disp_cursor(), um die Cursor-Anzeige zu steuern.

[Fns...]>Modul
oder [math]
4:ti_system
7:disp_at()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

Argument:

row	1 - 11, Ganzzahl
colum n	1 - 32, Ganzzahl
"text"	ist ein String, der im Bildschirmbereich umgebrochen wird

Hier gezeigte optionale Argumente für Farbe und Hintergrund: disp_at(row,col,"text","align",color 0-15, background color 0-5)

Beispiel:

Beispielprogramm:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,6,"hello")
disp_cursor(0)
disp_wait()
```

disp_at(row,"text","align")

Modul: ti_system

Syntax: disp_at(row,"text","align")

Beschreibung: Anzeigen von Text, wie im Plotting-Bildschirm für Zeile 1-11 angegeben. Die Zeile wird vor der Anzeige gelöscht. Bei der Verwendung in einer Schleife wird der Inhalt bei jeder Anzeige aktualisiert.

Wenn sich der Text am Ende des Programms befindet, wird hinter dem Text REPL mit dem Cursor >>>| angezeigt. Verwenden Sie disp_cursor() zur Steuerung der Cursor-Anzeige in Ihrem Programm vor disp_at().

Argument:

row	1 - 11, Ganzzahl
"text"	ist ein String, der im Bildschirmbereich umgebrochen wird
"align"	"left" (Standard) "center" "right"

Hier gezeigtes optionales Argument: disp_at (row,col,"text","align",color 0-15, 0-15)

Beispiel:

Beispielprogramm:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

[Fns...]>Modul
oder math
4:ti_system
7:disp_at()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

disp_clr() clear text screen

Modul: ti_system

Syntax: disp_clr() clear text screen

Beschreibung: Löscht den Bildschirm in der Shell-Umgebung. „Row 0-11, integer“ kann als optionales Argument verwendet werden, um eine Anzeigezeile der Shell-Umgebung zu löschen.

Beispiel:

Beispielprogramm:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
8:disp_clr()

[Fns…]>Modul
oder math
4:ti_system
8:disp_clr()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

disp_cursor() 0=off 1=on

Modul: ti_system

Syntax: disp_cursor() 0=off 1=on

Beschreibung: Steuert die Anzeige des Cursors in der Shell, wenn ein Programm ausgeführt wird.

Argument:

0 = aus

nicht 0 = ein

Beispiel:

Beispielprogramm:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
0:disp_cursor()

[Fns…]>Modul oder
math
4:ti_system
0:disp_cursor()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

disp_wait() [clear]

Modul: ti_system

Syntax: disp_wait() [clear]

Beschreibung: Stoppt die Ausführung des Programms an diesem Punkt und zeigt den Bildschirminhalt an, bis [clear] gedrückt und der Bildschirm gelöscht wird.

Beispiel:

Beispielprogramm:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
9:disp_wait()

[Fns…]>Modul
oder math
4:ti_system
9:disp_wait()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

E

e

Modul: math

Syntax: math.e oder e bei importiertem math-Modul

Beschreibung: Die Konstante e wird wie unten dargestellt angezeigt.

Beispiel:

```
>>>from math import *
>>>e
2.718281828459045
```

Alternatives Beispiel:

```
>>>import math
>>>math.e
2.718281828459045
```

2nd [e] (über ÷)

[Fns…] > Modul 1:math… > Const 1:e

elif :

Schlüsselwort

Einzelheiten finden Sie unter if..elif..else...

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Schlüsselwort

Einzelheiten finden Sie unter if..elif..else...

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Modul: ti_system

Syntax: escape()

Beschreibung: escape() gibt True oder False zurück.

Der Anfangswert ist False.

Wenn die Taste [clear] auf dem CE gedrückt wird, wird der Wert auf True gesetzt.

Beim Ausführen des Programms wird der Wert auf False zurückgesetzt.

Anwendungsbeispiel:

while not escape():

In einer while-Schleife eines Programms, wobei das Programm anbietet, die Schleife zu beenden, aber das Skript weiterlaufen zu lassen.

if escape():break

Kann in einem Programm für die Fehlersuche verwendet werden, um Variablen mit Shell [vars] zu untersuchen, nachdem das Programm ausgeführt und diese Pause verwendet wurde.

2nd [catalog]

Als Programmzeile:

2nd [rcl]
ti_system
5:while not escape():
6:if escape():break

[Fns…]>Modul oder math
4:ti-system
5:while not escape():
6:if escape():break

Importbefehle finden Sie unter 2nde [catalog] oder im Menü des Moduls ti_system.

eval()

Modul: Built-in

2nd [catalog]

Syntax: eval(x)

[Fns…] I/O
3:eval()

Beschreibung: Gibt die Auswertung des Ausdrucks x zurück.

Beispiel:

```
>>>a=7
>>>eval("a+9")
16
>>>eval('a+10')
17
```

except exception:

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „except“ in einem try..except-Codeblock. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

exp() (e hoch x)

Modul: math

2nd [e^x] (über ln)

Syntax: exp(x)

Beschreibung: Gibt e**x zurück.

2nd [catalog]

Beispiel:

```
>>>from math import *
>>>exp(1)
2.718281828459046
```

[Fns…] > Modul
1:math…
4:exp()

Alternatives Beispiel: [Tools] > 6:New Shell

```
>>>import math
>>>math.exp(1)
2.718281828459046
```

Importbefehle finden Sie unter 2nd [catalog][format].

.extend()

Modul: Built-in

2nd [catalog]

Syntax: listname.extend(newlist)

Beschreibung: Die Methode extend() ist eine Methode zum Erweitern von „newlist“ bis zum Ende einer Liste.

Beispiel:

```
>>>listA = [2,4,6,8]
>>>listA.extend([10,12])
>>>print(listA)
[2,4,6,8,10,12]
```

fabs()

Modul: math

2nd [catalog]

Syntax: fabs(x)

Beschreibung: Gibt den Absolutwert von x zurück.

[Fns…] > Modul
1:math...
2:fabs()

Beispiel:

```
>>>from math import *
>>>fabs(35-65.8)
30.8
```

Importbefehle finden Sie unter 2nd [catalog][format].

Siehe auch die integrierte Funktion abs().

False

Schlüsselwort

2nd [test] (über math)

Beschreibung: Gibt False zurück, wenn die ausgeführte Anweisung False ist. „False“ steht für einen falschen Wert bei Objekten des Typs „bool“.

2nd [catalog]

Beispiel:

```
>>>64<=32
False
```

[Fns…] > Ops
B:False

[a A #]

finally:

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie "finally“ in einem try..except..finally-Codeblock. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

float()

Modul: Built-in

2nd [catalog]

Syntax: float(x)

Beschreibung: Gibt x als Fließkommazahl zurück.

[Fns…] > Type
2:float()

Beispiel:

```
>>>float(35)
35.0
>>>float(“1234”)
1234.0
```

floor()

Modul: math

math Modul
1:math
9:floor()

Syntax: floor(x)

Beschreibung: Gibt die größte ganze Zahl zurück, die kleiner oder gleich x ist.

2nd [catalog]

Beispiel:

```
>>>from math import *
>>>floor(36.87)
36
>>>floor(-36.87)
-37
>>>floor(254)
254
```

[Fns…] > Modul
1:math
9:floor()

Importbefehle finden Sie unter 2nd [catalog].

fmod(x,y)

Modul: math

Syntax: fmod(x,y)

Beschreibung: Weitere Einzelheiten finden Sie in der Python-Dokumentation. Wird bevorzugt verwendet, wenn x und y Fließkommazahlen sind.

Gibt möglicherweise nicht das gleiche Ergebnis zurück wie x%y.

Beispiel:

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0 #Validierung aus der Beschreibung
-2.0
```

Siehe auch: x%y.

[math] Modul
1:math
7:fmod()

[2nd] [catalog]

[Fns…] > Modul
1:math…
7:fmod()

Importbefehle finden Sie unter [2nd] [catalog].

for i in list:

Schlüsselwort

Syntax: for i in list:

Beschreibung: Zum Iterieren über Listenelemente.

Beispiel:

```
>>> for i in [2,4,6]:
… print(i)
…
…
…
2
4
6
```

[Fns…] Ctl
7:for i in list:

[2nd] [catalog]

for i in range(size):

Schlüsselwort

[Fns…] Ctl 4:for i in range (size):

Syntax: for i in range(size)

Beschreibung: Zum Iterieren über einen Bereich.

Beispiel:

2nd [catalog]

```
>>> for i in range(3):
… print(i)
…
…
…
0
1
2
```

for i in range(start,stop):

Schlüsselwort

[Fns…] Ctl 5:for i in range (start,stop):

Syntax: for i in range(start,stop)

Beschreibung: Zum Iterieren über einen Bereich.

Beispiel:

2nd [catalog]

```
>>> for i in range(1,4):
… print(i)
…
…
…
1
2
3
```

for i in range(start,stop,step):

Schlüsselwort

[Fns…] Ctl
6:for i in range
(start,stop,step):

Syntax: for i in range(start,stop,step)

Beschreibung: Zum Iterieren über einen Bereich.

Beispiel:

2nd [catalog]

```
>>> for i in range(1,8,2):
… print(i)
…
…
…
1
3
4
7
```

str.format() string format

Modul: Built-in

2nd [catalog]

Syntax:str.format()

Beschreibung: Formatiert die angegebene Zeichenkette. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> print("{+f}".format(12.34))
+12.340000
```

frexp()

Modul: math

Syntax: frexp(x)

Beschreibung: Gibt ein Paar (y,n) zurück, wobei x == y * 2**n. y ist eine Fließkommazahl, wobei 0.5<abs(y)<1; und n ist eine Ganzzahl.

Beispiel:

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11 #Beschreibung validieren
2000.0
```

[math] Modul
1:math
A:frexp()

[2nd] [catalog]

[Fns...] > Modul
1:math
A:frexp()

Importbefehle finden Sie unter [2nd] [catalog].

from PROGRAM import *

Schlüsselwort

Syntax: from PROGRAM import *

Beschreibung: Zum Importieren eines Programms. Importiert die öffentlichen Attribute eines Python-Moduls in den aktuellen Namensraum.

Shell [Tools]
A:from
PROGRAM
import *

[2nd] [catalog]

from math import *

Schlüsselwort

Syntax: from math import *

Beschreibung: Zum Import aller Funktionen und Konstanten aus dem math module.

[math] Modul
1:math…
1:from math import *

[Fns..] > Modul
1:math…
1:from math import *

[2nd] [catalog]

from random import *

Schlüsselwort

Syntax: from random import *

Beschreibung: Zum Import aller Funktionen aus dem random module.

[math] Modul
2:random…
1:from random import *

[Fns..] > Modul
2:random…
1:from random import *

[2nd] [catalog]

from time import *

Schlüsselwort

Syntax: from time import *

Beschreibung: Zum Import aller Methoden aus dem time-Modul.

Beispiel:

Siehe Beispielprogramm: DASH1.

2nd [catalog]

math Modul
3:time…
1:from time import *

[Fns…]>Modul
3:time…
1:from time import *

from ti_system import *

Schlüsselwort

Syntax: from ti_system import *

Beschreibung: Zum Import aller Methoden aus dem ti_system-Modul.

Beispiel:

Siehe Beispielprogramm: REGEQ1.

2nd [catalog]

math Modul
4:ti_system…
1:from system import *

[Fns…]>Modul
4:ti_system…
1:from system import *

from ti_hub import *

Schlüsselwort 2nd [catalog]

Syntax: from ti_hub import *

Beschreibung: Zum Import aller Methoden aus dem ti_hub module. Verwenden Sie für einzelne Ein- und Ausgabegeräte die dynamische Modulfunktionalität, indem Sie das Gerät im Editor im Menü [Fns…]>Modul>ti_hub>Import auswählen.

Siehe ti_hub module – Import zum Editor hinzufügen und ti_hub-Sensormodul zum Modul-Menü hinzufügen.

Beispiel:

Siehe Beispielprogramm: DASH1.

G

global

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „global“, um globale Variablen innerhalb einer Funktion zu erstellen.

Weitere Einzelheiten finden Sie in der CircuitPython-Dokumentation.

grid(xscl,yscl,"style")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.grid(xscl,yscl,"style")

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 3:grid()

Beschreibung: Zeigt ein Gitter mit angegebenem Maßstab für die x- und y-Achse an. Hinweis: Das Plotting wird durchgeführt, wenn plt.show_plot() ausgeführt wird.

Die Einstellung der Gitterfarbe ist das optionale Argument von (r,g,b) unter Verwendung der Werte 0-255 mit dem Standardwert Grau (192,192,192).

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Standardwert für xscl oder yscl = 1.0.

"style" = "dot" (Standard), "dash", "solid" oder "point"

Beispiel:

Siehe Beispielprogramme: COLORLIN oder GRAPH.

grid(xscl,yscl,"style",(r,g,b))

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.grid(xscl,yscl,"style",(r,g,b))

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 3:grid()

Beschreibung: Zeigt ein Gitter mit angegebenem Maßstab für die x- und y-Achse an. Hinweis: Das Plotting wird durchgeführt, wenn plt.show_plot() ausgeführt wird.

Die Einstellung der Gitterfarbe ist das optionale Argument von (r,g,b) unter Verwendung der Werte 0-255 mit dem Standardwert Grau (192,192,192).

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Standardwert für xscl oder yscl = 1.0.

"style" = "dot" (Standard), "dash", "solid" oder "point".

Wenn die Werte xscl oder yscl kleiner als 1/50 der Differenz zwischen xmax-xmin oder ymax-ymin betragen, dann wird eine Ausnahme 'Invalid grid scale value' ausgegeben.

Beispiel:

Siehe Beispielprogramm: GRAPH.

hex(integer)

Modul: Built-in

2nd [catalog]

Syntax: hex(integer)

Beschreibung: Zeigt das hexadezimale Format des Arguments „integer“ an. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> hex(16)
'0x10'
>>> hex(16**2)
'0x100'
```

"if :"	
Einzelheiten finden Sie unter if..elif..else...	2nd [catalog] [Fns…] > Ctl 1:if.. 2:if..else.. 3:if..elif..else 9:elif : 0:else:

if..elif..else..

Schlüsselwort

Syntax: ••Graue eingerückte Identifikatoren werden in der Python-App aus Gründen der Benutzerfreundlichkeit automatisch bereitgestellt.

if :

••

elif :

••

else:

Beschreibung: if..elif..else ist eine bedingte Anweisung. Der Editor fügt Einzüge automatisch als graue Punkte ein, um Sie bei der korrekten Programmierung der Einzüge zu unterstützen.

Beispiel: Erstellen und starten Sie ein Programm, z. B. S01, aus dem Editor heraus.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

```
def f(a):
••if a>0:
••••print(a)
••elif a==0:
••••print("zero")
••else:
••••a=-a
••••print(a)
```

Shell-Interaktion

```
>>> # Shell Reinitialized
>>> # Running S01
>>>from S01 import * #fügt automatisch ein
>>>f(5)
5
>>>f(0)
Null
>>>f(-5)
5
```

if..else..

Schlüsselwort 2nd [catalog]

Einzelheiten finden Sie unter if..elif..else...

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Modul: Built-in 2nd [catalog]

Syntax:var.imag

Beschreibung: Bestimmt den Imaginärteil einer angegebenen Variablen eines komplexen Zahlentyps.

Beispiel:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

import math

Schlüsselwort

Syntax: import math 2nd [catalog]

Beschreibung: Über diesen Befehl wird auf das math module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute des Moduls „math“ innerhalb seines eigenen Namensraums.

import random

Schlüsselwort

Syntax: import random [2nd] [catalog]

Beschreibung: Über diesen Befehl wird auf das random module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute des Moduls „random“ innerhalb seines eigenen Namensraums.

import ti_hub

Schlüsselwort [2nd] [catalog]

Syntax: import ti_hub

Beschreibung: Über diesen Befehl wird auf das ti_hub module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute von ti_hub module innerhalb seines eigenen Namensraums.

Verwenden Sie für einzelne Ein- und Ausgabegeräte die dynamische Modulfunktionalität, indem Sie das Gerät im Editor im Menü [Fns…]>Modul>ti_hub>Import auswählen.

Siehe: [Fns…] > Modul: ti_hub module.

import time

Schlüsselwort [2nd] [catalog]

Syntax: import time

Beschreibung: Über diesen Befehl wird auf das time module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute des Moduls „time“ innerhalb seines eigenen Namensraums.

Siehe: [Fns…] > Modul: time and ti_system modules.

import ti_plotlib as plt

Schlüsselwort

Syntax: import ti_plotlib as plt

Beschreibung: Über diesen Befehl wird auf das ti_plotlib module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute von ti_plotlib module innerhalb seines eigenen Namensraums. Attribute von ti_plotlib module müssen als plt.attribute eingegeben werden.

Beispiel:

Siehe Beispielprogramm: COLORLIN.

[2nd] [catalog]

[math] Modul
5:ti_plotlib...
1:import ti_plotlib as plt

[Fns…]>Modul
5:ti_plotlib...
1:import ti_plotlib as plt

import ti_rover as rv

Schlüsselwort

Syntax: import ti_rover as rv

Beschreibung: Über diesen Befehl wird auf das ti_rover module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute von ti_rover module innerhalb seines eigenen Namensraums. Attribute von ti_rover module müssen als rv.attribute eingegeben werden.

Beispiel:

Siehe Beispielprogramm: ROVER.

[2nd] [catalog]

[math] Modul
7:ti_rover...
1:import ti_rover as rv

[Fns…]>Modul
7:ti_rover...
1:import ti_rover as rv

import ti_system

Schlüsselwort 2nd [catalog]

Syntax: import ti_system

Beschreibung: Über diesen Befehl wird auf das ti_system module zugegriffen. Diese Anweisung importiert die öffentlichen Attribute von ti_system module innerhalb seines eigenen Namensraums.

Beispiel:

Siehe Beispielprogramm: REGEQ1.

in

Schlüsselwort 2nd [catalog]

Beschreibung: Zur Überprüfung, ob ein Wert in einer Folge enthalten ist oder um zum Iterieren einer Folge in einer for-Schleife.

.index(x)

Modul: Built-in 2nd [catalog]

Syntax:var.index(x)

Beschreibung: Gibt den Index oder die Position eines Elements einer Liste zurück. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> a=[12,35,45]
>>> print(a.index(12))
0
>>> print(a.index(35))
1
>>> print(a.index(45))
2
```

input()

Modul: Built-in 2nd [catalog]

input()

Syntax: input()

Beschreibung: Eingabeaufforderung

[Fns…] I/O
2:input()

Beispiel:

```
>>>input("Name? ")
Name? Me
'Me'
```

Alternatives Beispiel:

```
CreateProgram A
len=float(input("len: "))
print(len)
```

```
RunProgram A
>>> # Shell Reinitialized
>>> # Running A
>>>from A import *
len: 15(enter15)
15.0(outputfloat 15.0)
```

.insert(index,x)

Modul: Built-in

2nd [list] List
8:.insert(index,x)

Syntax: listname.insert(index,x)

Beschreibung: method insert() fügt innerhalb einer Folge ein Element x nach dem Index ein.

2nd [catalog]

Beispiel:

```
>>>listA = [2,4,6,8]
>>>listA.insert(3,15)
>>>print(listA)
[2,4,6,15,8]
```

[Fns…] > List
8:.insert(index,x)

int()

Modul: Built-in

2nd [catalog]

Syntax: int(x)

Beschreibung: Gibt x als ganzzahliges Objekt zurück.

[Fns…] > Type
1:int()

Beispiel:

```
>>>int(34.67)
34
>>>int(1234.56)
1234
```

is

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „is“, um zu testen, ob zwei Objekte das gleiche Objekt sind.

labels("xlabel","ylabel",x,y)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.labels("xlabel","ylabel",x,y)

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 7:labels()

Beschreibung: Zeigt die Beschriftungen „xlabel“ und „ylabel“ an den Diagrammachsen in Höhe der Zeilenpositionen x und y an. Passen Sie sie ggf. an die Diagrammdarstellung an.

„xlabel“ wird auf der angegebenen Zeile x (Standardzeile: 12) rechtsbündig positioniert.

„ylabel“ wird auf der angegebenen Zeile y (Standardzeile: 2) linksbündig positioniert.

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Hinweis: plt.labels("|","",12,2) wird mit den Standardwerten für die x- und y-Zeile (12,2) eingefügt; dies kann anschließend für Ihr Programm modifiziert werden.

Beispiel:

Siehe Beispielprogramm: GRAPH.

lambda

Schlüsselwort

2nd [catalog]

Syntax: lambda arguments : expression

Beschreibung: Verwenden Sie „lambda“, um eine anonyme Funktion zu definieren. Einzelheiten finden Sie in der Python-Dokumentation.

len()

Modul: Built-in

Syntax: len(sequence)

Beschreibung: Gibt die Anzahl der Elemente im Argument zurück. Das Argument kann eine Folge oder eine Sammlung sein.

Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>>mylist=[2,4,6,8,10]
>>>len(mylist)
5
```

2nd [list] (über stat) List
3:len()

2nd [catalog]

[Fns…] > List
3:len()

line(x1,y1,x2,y2,"mode")

Modul: ti_plotlib

Syntax: plt.line(x1,y1,x2,y2,"mode")

Beschreibung: Zeigt eine Strecke von (x1,y1) bis (x2,y2) an.

Größe und Stil werden mittels pen() und color() vor line() festgelegt.

Argumente:

x1,y1, x2,y2 sind echte Fließkommazahlen.

"mode": Bei Standard "" wird keine Pfeilspitze gezeichnet.
Bei "arrow" wird eine Vektorpfeilspitze bei (x2,y2) gezeichnet.

Beispiel:

Siehe Beispielprogramm: COLORLIN.

2nd [catalog]

[Fns…]>Modul oder math
5:ti_plotlib…>
Draw
7:line or vector

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

lin_reg(xlist,ylist,"disp",row)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.lin_reg(xlist,ylist,"disp",row)

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 8:lin_reg()

Beschreibung: Berechnet und zeichnet das lineare Regressionsmodell ax+b von xlist,ylist. Diese Methode muss nach der scatter-Methode ausgeführt werden. Die Standardanzeige der Gleichung ist "center" in Zeile 11.

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Argument:

"disp"	"left"
	"center"
	"right"
row	1 - 12

plt.a (Steigung) und plt.b (Achsenabschnitt) werden gespeichert, wenn lin_reg ausgeführt wird.

Beispiel:

Siehe Beispielprogramm: LINREGR.

list(sequence)

Modul: Built-in

Syntax: list(sequence)

Beschreibung: Veränderbare Folge von Elementen desselben Typs.

list()" konvertiert die Argumente in den Typ „list“. Wie bei vielen anderen Folgen müssen die Elemente einer Liste nicht vom selben Typ sein.

Beispiel:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
```

Beispiel:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
>>> list({1,2,"c", 7})
[7, 1, 2, 'c']
>>> list("foobar")
['f', 'o', 'o', 'b', 'a', 'r']
```

2nd [list] (über stat) List
2:list(sequence)

2nd [catalog]

[Fns…] > List
2:list(sequence)

log(x,base)

Modul: math

Syntax: log(x,base)

Beschreibung: log(x) ohne Basis gibt den natürlichen Logarithmus x zurück.

Beispiel:

```
>>>from math import *
>>>log(e)
1.0
>>>log(100,10)
2.0
>>>log(32,2)
5.0
```

2nd log for log (x,10)

2nd ln for log (x) (natürlicher Logarithmus)

math Modul
1:math...
6:log(x,base)

2nd [catalog]

[Fns...] > Modul
1:math...
6:log(x,base)

Importbefehle finden Sie unter 2nd [catalog].

math.function

Modul: math

2nd [catalog]

Syntax: math.function

Beschreibung: Ist nach dem Befehl „import math“ zu benutzen, um eine Funktion im math-Modul zu verwenden.

Beispiel:

```
>>>import math
>>>math.cos(0)
1.0
```

max()

Modul: Built-in

2nd [list] (über stat) List 4:max()

Syntax: max(sequence)

Beschreibung: Gibt den Maximalwert der Folge zurück. Weitere Informationen zu max() finden Sie in der Python-Dokumentation.

2nd [catalog]

Beispiel:

```
>>>listA=[15,2,30,12,8]
>>>max(listA)
30
```

[Fns…] > List 4:max()

min()

Modul: Built-in

2nd [list] (über stat) List 5:min()

Syntax: min(sequence)

Beschreibung: Gibt den Minimalwert der Folge zurück. Weitere Informationen zu min() finden Sie in der Python-Dokumentation.

2nd [catalog]

Beispiel:

```
>>>listA=[15,2,30,12,8]
>>>min(listA)
2
```

[Fns…] > List 5:min()

monotonic() elapsed time

Modul: time

2nd [catalog]

Syntax: monotonic() elapsed time

Beschreibung: Gibt einen Zeitwert vom Zeitpunkt der Ausführung zurück. Verwenden Sie den Rückgabewert zum Vergleich mit anderen Werten aus monotonic().

[Fns...]>Modul oder math
3:time
3:momotonic()

Beispiel:

Beispielprogramm:

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls time.

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)

Run the program EXAMPLE until execution stops.
>>>15.0
```

None

Schlüsselwort 2nd [catalog]

Beschreibung: „None“ stellt das Fehlen eines Wertes dar.

[a A #]

Beispiel:

```
>>> def f(x):
…x
…
…
…
>>> print(f(2))
None
```

nonlocal

Schlüsselwort 2nd [catalog]

Syntax: nonlocal

Beschreibung: Verwenden Sie „nonlocal“, um festzulegen, dass eine Variable nicht lokal ist. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

not

Schlüsselwort 2nd [test] Ops 0:not

Syntax: not x

Beschreibung: Wertet als True aus, wenn x False ist, anderenfalls als False. Wird mit Leerzeichen vor und hinter dem Schlüsselwort „not“ eingefügt. Kann nach Bedarf bearbeitet werden.

[Fns…] > Ops 0:not

Beispiel:

2nd [catalog]

```
>>> not 2<5 #Bearbeiten Sie das Leerzeichen vor „not“
False
>>>3<8 and not 2<5
False
```

[a A #]

oct(integer)

Modul: Built-in

2nd [catalog]

Syntax: oct(integer)

Beschreibung: Gibt die oktale Darstellung der Ganzzahl zurück. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> oct(8)
'0o10'
>>> oct(64)
'0o100'
```

or

Schlüsselwort

2nd [test] Ops 9:or

Syntax: x or y

[Fns…] > Ops 9:or

Beschreibung: Kann True oder False zurückgeben. Gibt x zurück, wenn x als True ausgewertet wird, anderenfalls wird y ausgegeben. Wird mit Leerzeichen vor und hinter dem „or“ eingefügt. Kann nach Bedarf bearbeitet werden.

2nd [catalog]

[a A #]

Beispiel:

```
>>>2<5 or 5<10
True
>>>2<5 or 15<10
True
>>>12<5 or 15<10
False
>>> 3 or {}
3
>>> [] or {2}
{2}
```

ord("character")

Modul: Built-in

2nd [catalog]

Syntax: ord("character")

Beschreibung: Gibt den Unicode-Wert des Zeichens zurück. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> ord("#")
35
>>> ord("/")
47
```

pass

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „pass“ in einer leeren Funktions- oder Klassendefinition als Platzhalter für zukünftigen Code, während Sie Ihr Programm aufbauen. Leere Definitionen führen bei der Programmausführung nicht zu einem Fehler.

pen("size","style")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.pen("size","style")

[Fns...]>Modul oder math
5:ti_plotlib...> Draw
9:pen()

Beschreibung: Legt das Aussehen aller folgenden Zeilen fest, bis ein neues pen() ausgeführt wird.

Argument:

Standardmäßig ist pen() "thin" und "solid".

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

"size"	"thin"
	"medium"
	"thick"
"style"	"solid"
	"dot"
	"dash"

Beispiel:

Siehe Beispielprogramme: COLORLIN oder GRAPH.

pi

Modul: math

2nd [π] (über sin)

Syntax: math.pi oder pi bei importiertem math-Modul.

Beschreibung: Die Konstante pi wird wie unten dargestellt angezeigt.

[Fns…] > Modul
1:math… > Const
2:pi

Beispiel:

```
>>>from math import *
>>>pi
3.141592653589793
```

Alternatives Beispiel:

```
>>>import math
>>>math.pi
3.141592653589793
```

plot(xlist,ylist,"mark")

Modul: ti_plotlib

Syntax: plt.plot(xlist,ylist,"mark")

Beschreibung: Es wird ein Liniendiagramm mit geordneten Paaren aus der angegebenen xlist und ylist angezeigt. Linienstärke und -größe werden unter Verwendung von plt.pen() eingestellt.

xlist und ylist müssen echte Fließkommazahlen sein und Listen müssen die gleiche Dimension haben.

Argument:

„mark“ ist das Markierungszeichen, wie nachstehend erläutert:

o	Gefüllter Punkt (Standard)
+	Kreuz
x	x
.	Pixel

Beispiel:

Siehe Beispielprogramm: LINREGR.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 5:Connected Plot with Lists

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

plot(x,y,"mark")

Modul: ti_plotlib

Syntax: plt.plot(x,y,"mark")

Beschreibung: Ein Punktdiagramm (x,y) zeigt den angegebenen x- und y-Wert an.

xlist und ylist müssen echte Fließkommazahlen sein und Listen müssen die gleiche Dimension haben.

Argument:

„mark“ ist das Markierungszeichen, wie nachstehend erläutert:

o	Gefüllter Punkt (Standard)
+	Kreuz
x	x
.	Pixel

Beispiel:

Siehe Beispielprogramm: LINREGR.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 6:plot a Point

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

pow(x,y)

Modul: math

Syntax: pow(x,y)

Beschreibung: Ergibt x hoch y. Konvertiert x und y in Fließkommazahlen. Weitere Informationen finden Sie in der Python-Dokumentation.

Verwenden Sie zur Berechnung exakter ganzzahliger Potenzen die integrierte pow(x,y)-Funktion oder **.

Beispiel:

```
>>>from math import *
>>>pow(2,3)
>>>8.0
```

Beispiel unter Verwendung von: Built-in:

[Tools] > 6:New Shell

```
>>>pow(2,3)
8
>>>2**3
8
```

[math] Modul
1:math
5:pow(x,y)

[2nd] [catalog]

[Fns…] > Modul
1:math
5:pow(x,y)

Importbefehle finden Sie unter [2nd] [catalog].

print()

Modul: Built-in

Syntax: print(argument)

Beschreibung: Zeigt das Argument als Zeichenfolge an.

Beispiel:

```
>>>x=57.4
>>>print("my number is =", x)
my number is= 57.4
```

[2nd] [catalog]

[Fns…] > I/O
1:print()

radians()) degree ▶radians

Modul: math

Syntax: radians(x)

Beschreibung: Konvertiert den Winkel x von degrees nach radians.

Beispiel:

```
>>>from math import *
>>>radians(180.0)
3.141592653589793
>>>radians(90.0)
1.570796326794897
```

[sin] Trig
1:radians()

[2nd] [catalog]

[Fns…] > Modul
1:math… > Trig
1:radians()

raise

Schlüsselwort

Syntax: raise exception

Beschreibung: Verwenden Sie „raise“, um eine bestimmte Ausnahme auszulösen und Ihr Programm zu stoppen.

[2nd] [catalog]

randint(min,max)

Modul: random

Syntax: randint(min,max)

Beschreibung: Ergibt eine ganzzahlige Zufallszahl zwischen min und max.

math Modul
2:random
4:randint
(min,max)

Beispiel:

```
>>>from random import *
>>>randint(10,20)
>>>15
```

[Fns…] > Modul
2:random…
4:randint
(min,max)

Alternatives Beispiel:

```
>>>import random
>>>random.randint(200,450)
306
```

2nd [catalog]

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

Importbefehle finden Sie unter 2nd [catalog].

random()

Modul: random

Syntax: random()

Beschreibung: Ergibt eine Fließkommazahl zwischen 0 und 1,0. Diese Funktion verwendet keine Argumente.

Beispiel:

```
>>>from random import *
>>>random()
0.5381466990230621
```

Alternatives Beispiel:

```
>>>import random
>>>random.random()
0.2695098437037318
```

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

[math] Modul 2:random… Random 2:random()

[Fns…] > Modul 2:random… Random 2:random()

[2nd] [catalog]

Importbefehle finden Sie unter [2nd] [catalog].

random.function

Modul: random

Syntax: random.function

Beschreibung: Nach dem Import von random verwenden, um auf eine Funktion im Modul random zuzugreifen.

Beispiel:

```
>>>import random
>>>random.randint(1,15)
2
```

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

[2nd] [catalog]

randrange(start,stop,step)

Modul: random

Syntax: randrange(start,stop,step)

Beschreibung: Ergibt eine Zufallszahl zwischen Start und Stopp mit definierter Schrittgröße.

Beispiel:

```
>>>from random import *
>>>randrange(10,50,2)
12
```

Alternatives Beispiel:

```
>>>import random
>>>random.randrange(10,50,2)
48
```

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

[math] Modul 2:random… Random 6:randrange (start,stop,step)

[math] Modul 2:random… Random 6:randrange (start,stop,step)

[2nd] [catalog]

Importbefehle finden Sie unter [2nd][catalog].

range(start,stop,step)

Modul: Integriert

[2nd] [catalog]

Syntax: range(start,stop,step)

Beschreibung: Verwenden Sie die Funktion „range“, um eine Zahlenfolge zurückzugeben. Alle Argumente sind optional. Die Standardwerte sind 0 für „start“ und 1 für „step“; bei „stop“ endet die Folge.

Beispiel:

```
>>> x = range(2,10,3)
>>> for i in x
… print(i)
…
…
2
5
8
```

.real

Modul: Built-in 2nd [catalog]

Syntax:var.real

Beschreibung: Bestimmt den Realteil einer angegebenen Variablen eines komplexen Zahlentyps.

Beispiel:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

var=recall_list("name") 1-6

Modul: ti_system

Syntax:var=recall_list("name") 1-6

Beschreibung: Rufen Sie eine vordefinierte Betriebssystem-Liste auf. Die Länge der Liste muss kleiner als oder gleich 100 sein.

Argument: "**name**"

Für L1-L6 im Betriebssystem

1 - 6
"1" - "6"
'1' - '6'

Für benutzerdefinierte Betriebssystem-Liste „name“

----- Max. 5 Zeichen, Zahlen oder Buchstaben, mit Buchstaben beginnend, Buchstaben müssen groß geschrieben werden.

Beispiele:

"ABCDE"

"R12"

"L1" ist ein benutzerdefiniertes L1 und nicht mit dem Betriebssystem-L1 identisch

Zur Erinnerung: Python ist doppelt genau. Python unterstützt mehr Ziffern als das Betriebssystem.

Beispiel:

Beispielprogramm:

Erstellen Sie eine Liste im Betriebssystem.
LIST={1,2,3}

Führen Sie die Python-App aus.
Erstellen Sie ein neues Programm.

```
import ti_system as *
xlist=recall_list("LIST")
print xlist
```

Führen Sie das Programm AA aus.
Die Shell zeigt die Ausgabe an.

```
[1.0, 2.0, 3.0]
```

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_
list()

[Fns...]>Modul
oder math
4:ti_system
4:var=recall_
list()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

var=recall_RegEQ()

Modul: ti_system

Syntax:var=recall_RegEQ()

Beschreibung: Rufen Sie die Variable RegEQ aus dem CE-Betriebssystem auf. Bevor RegEQ in der Python-App aufgerufen werden kann, muss die Regressionsgleichung im Betriebssystem berechnet werden.

Beispiel:

Siehe Beispielprogramm: REGEQ1.

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_
REGEQ()

[Fns…]>Modul
oder math
4:ti_system
4:var=recall_
REGEQ()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

.remove(x)

Modul: Built-in

Syntax: listname.remove(item)

Beschreibung: Die Methode remove() entfernt die erste Instanz eines Elements aus einer Folge.

Beispiel:

```
>>>listA = [2,4,6,8,6]
>>>listA.remove(6)
>>>print(listA)
[2,4,8,6]
```

2nd [list]
List
7:.remove(x)

2nd [catalog]

[Fns…] > List
7:.remove(x)

return

Modul: Built-in

2nd [catalog]

Syntax: return expression

Beschreibung: Eine return-Anweisung definiert den von einer Funktion erzeugten Wert. Python-Funktionen geben standardmäßig „None“ zurück. Siehe auch: def function ():

[Fns…] > Func 1:def function():

Beispiel:

[Fns…] > Func 2:return

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

.reverse()

Modul: Built-in

2nd [catalog]

Syntax: listname.reverse()

Beschreibung: Gibt die Reihenfolge der Elemente in einer Folge zurück.

Beispiel:

```
>>>list1=[15,-32,4]
>>>list1.reverse()
>>>print(list1)
[4,-32,15]
```

round()

Modul: Integriert

2nd [catalog]

Syntax: round(number, digits)

Beschreibung: Verwenden Sie die Rundungsfunktion, um eine Fließkommazahl auf die angegebenen Ziffern gerundet zurückzugeben. Die Standardziffer ist 0, wobei die nächste Ganzzahl angegeben wird.

Beispiel:

```
>>>round(23.12456)
23
>>>round(23.12456,3)
23.125
```

scatter(xlist,ylist,"mark")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.scatter(xlist,ylist,"mark")

[Fns...]>Modul oder math 5:ti_plotlib...> Draw 4:scatter()

Beschreibung: Eine Folge geordneter Paare aus (xlist,ylist) wird unter Angabe des Markierungsstils geplottet. Linienstärke und -größe werden unter Verwendung von plt.pen() eingestellt.

xlist und ylist müssen echte Fließkommazahlen sein und Listen müssen die gleiche Dimension haben.

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Argument:

„mark“ ist das Markierungszeichen, wie nachstehend erläutert:

o	Gefüllter Punkt (Standard)
+	Kreuz
x	x
.	Pixel

Beispiel:

Siehe Beispielprogramm: LINREGR.

seed()

Modul: random

Syntax: seed() oder seed(x). wobei x eine Ganzzahl ist

Beschreibung: Zufallszahlengenerierung initialisieren.

Beispiel:

```
>>>from random import *
>>>seed(12)
>>>random()
0.9079708720366826
>>>seed(10)
>>>random()
0.9063990882481896
>>>seed(12)
>>>random()
0.9079708720366826
```

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

[math] Modul 2:random… Random 7:seed()

[Fns…] > Modul 2:random… Random 7:seed()

[2nd] [catalog]

Importbefehle finden Sie unter [2nd] [catalog].

set(sequence)

Modul: Built-in

[2nd] [catalog]

Syntax: set(sequence)

Beschreibung: Gibt eine Folge als Satz zurück. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> print(set("84CE")
{'E', '8', '4', 'C'}
```

show_plot() display > [clear]

Modul: ti_plotlib

Syntax: plt.show_plot() display > [clear]

Beschreibung: Führt die Anzeige des Plots aus, wie im Programm eingerichtet.

show_plot() muss hinter allen Objekten für das Plot-Setup platziert werden. Die Reihenfolge der Plotting-Objekte im Programm wird durch die Reihenfolge im Setup-Menü vorgeschlagen.

Um die Hilfe für die Plotting-Vorlage aus dem Dateimanager heraus anzuzeigen, wählen Sie [New] ([zoom]) und dann [Types] ([zoom]), um den Programmtyp „Plotting (x,y) & Text“ auszuwählen.

Nach Ausführung des Programms wird die Plotting-Anzeige durch Drücken von [clear] gelöscht, um zur Shell-Eingabeaufforderung zurückzukehren.

Beispiel:

Siehe Beispielprogramme: COLORLIN oder GRAPH.

2nd [catalog]

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 9:show_plot

[Fns...]>Modul oder [math] 5:ti_plotlib... > Draw 9:show_plot()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

sin()

Modul: math

Syntax: sin()

Beschreibung: Ergibt den Sinus von x. Das Argument Winkel wird imBogenmaß (Radian) ausgegeben.

Beispiel:

```
>>>from math import *
>>>sin(pi/2)
1.0
```

sin 3:sin()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
3:sin()

Importbefehle finden Sie unter 2nd [catalog].

sleep(seconds)

Modul: ti_system; time

Syntax: sleep(seconds)

Beschreibung: Das Programm ruht, bis die in Sekunden angegebene Zeit verstrichen ist. Das Argument Sekunden ist eine Fließkommazahl.

Beispiel:

Beispielprogramm:

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

Starten Sie das Programm TIME

```
>>>15.0
```

2nd [catalog]

2nd [rcl]
ti_system
A:sleep()

[Fns…]>Modul
oder math
4:ti_system
A:sleep()

[Fns…]>Modul
oder math
3:time
2:sleep()

Importbefehle finden Sie unter 2nd [catalog] oder im Menü des Moduls ti_system.

.sort()

Modul: Built-in

Syntax: listname.sort()

Beschreibung: Die Methode sortiert eine Liste. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA) #listA wird in eine sortierte Liste
umgewandelt
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nd [list]
(über stat
List A:.sort()

2nd [catalog]

[Fns…] >
List
A:sort()

sorted()

Modul: Built-in

Syntax: sorted(sequence)

Beschreibung: Gibt eine sortierte Liste aus einer Folge zurück.

Beispiel:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>sorted(listA)
[2,3,3,4,4,4,5,6,6,7,8,9]
>>>print(listA) #listA wurde nicht geändert
[4,3,6,2,7,4,8,9,3,5,4,6]
```

2nd [list] (über
stat) List
0:sorted()

2nd [catalog]

[Fns…] > List
0:sorted()

.split(x)

Modul: Built-in

2nd [catalog]

Syntax:var.split(x)

Beschreibung: Die Methode gibt eine Liste mit dem angegebenen Trennzeichen zurück. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>> a="red,blue,green"
>>> a.split(",")
['red', 'blue', 'green']
```

sqrt()

Modul: math

math Modul 1:math 3:sqrt ()

Syntax: sqrt(x)

Beschreibung: Gibt die Quadratwurzel von x zurück.

Beispiel:

2nd [catalog]

```
>>>from math import *
>>>sqrt(25)
5.0
```

[Fns…] > Modul 1:math 3:sqrt()

Importbefehle finden Sie unter 2nd [catalog][format].

store_list("name",var) 1-6

Modul: ti_system

Syntax: store_list("name",var) 1-6

Beschreibung: Speichert eine Liste von der Ausführung eines Python-Skripts in eine Betriebssystem-Listenvariable „name“, wobei var eine definierte Python-Liste ist. Die Länge der Liste muss kleiner als oder gleich 100 sein.

Argument: "**name**"

Für L1-L6 im Betriebssystem

1 - 6
"1" - "6"
'1' - '6'

Für benutzerdefinierte Betriebssystem-Liste „name“

----- Max. 5 Zeichen, Zahlen oder Buchstaben, mit Buchstaben beginnend, Buchstaben müssen groß geschrieben werden.

Beispiele:

"ABCDE"

"R12"

"L1" ist ein benutzerdefiniertes L1 und nicht mit dem Betriebssystem-L1 identisch

Zur Erinnerung: Python ist doppelt genau, d. h. es werden mehr Ziffern als im Betriebssystem unterstützt.

Beispiel:

```
>>>a=[1,2,3]
>>>store_list("1",a)
>>>
```

Beenden Sie die Python-App und drücken Sie [2nd][L1] (über [1]) und [enter] im Hauptbildschirm, um die Liste [L1] als {1 2 3} anzuzeigen.

[2nd] [catalog]

[2nd][rcl]
ti_system
3:var=store_list
()

[Fns…]>Modul
oder [math]
4:ti_system
3:var=store_list
()

Importbefehle finden Sie unter [2nd] [catalog] oder im Menü des Moduls ti_system.

str()

Modul: Built-in

2nd [catalog]

Syntax: str(argument)

Beschreibung: Konvertiert "argument" in einen String.

[Fns…] > Type 3 :str()

Beispiel:

```
>>>x=2+3
>>>str(x)
'5'
```

sum()

Modul: Built-in

2nd [list] (über stat) List 9:sum()

Syntax: sum(sequence)

Beschreibung: Gibt die Summe der Elemente in einer Folge zurück.

2nd [catalog]

Beispiel:

```
>>>listA=[2,4,6,8,10]
>>>sum(listA)
30
```

[Fns…] > List 9:sum()

tan()

Modul: math

Syntax: tan(x)

Beschreibung: Liefert den Tangens von x. Das Winkelargument wird in radians (Bogenmaß) ausgegeben.

Beispiel:

```
>>>from math import *
>>>tan(pi/4)
1.0
```

[sin] 5:tan()

[Fns…] > Modul
1:math… > Trig
5:tan()

[2nd] [catalog]

Importbefehle finden Sie unter [2nd] [catalog].

text_at(row,"text","align")

Modul: ti_plotlib

Syntax: plt.text_at(row,"text","align")

Beschreibung: "text" wird im Plotting-Bereich angezeigt, wie unter "align“ angegeben.

Zeile	Ganzzahl (1 bis 12)
"text"	Der String wird abgeschnitten, wenn er zu lang ist
"align"	"left" (Standard) "center" "right"
optional	1 löscht Zeile vor dem Text (Standard) 0 Zeile wird nicht gelöscht

Beispiel:

Siehe Beispielprogramm: DASH1.

[2nd] [catalog]

[Fns…]>Modul oder [math]
5:ti_plotlib…>
Draw
0:text_at()

Importbefehle finden Sie unter [2nd] [catalog] oder im Menü ti_plotlib Setup.

time.function

Modul: Built-in

2nd [catalog]

Syntax: time.function

Beschreibung: Nach dem Import von time verwenden, um auf eine Funktion im Modul time zuzugreifen.

Beispiel:

Siehe:[Fns...]>Modul: time and ti_system modules.

title("title")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.title("title")

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 8:title()

Beschreibung: "title" wird mittig in der obersten Zeile des Fensters angezeigt. Der Titel wird abgeschnitten, wenn er zu lang ist.

Beispiel:

Siehe Beispielprogramm: COLORLIN.

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

ti_hub.function

Modul: ti_hub [2nd] [catalog]

Syntax: ti_hub.function

Beschreibung: Nach dem Import von ti_hub verwenden, um auf eine Funktion im Modul ti_hub zuzugreifen.

Beispiel:

Siehe:[Fns…]>Modul: ti_hub module.

ti_system.function

Modul: ti_system [2nd] [catalog]

Syntax: ti_system.function

Beschreibung: Nach dem Import von ti_system verwenden, um auf eine Funktion im Modul ti_system zuzugreifen.

Beispiel:

```
>>> # Shell Reinitialized
>>>import ti_system
>>>ti_system.disp_at(6,8,"texte")

texte>>>|

#wird in Zeile 6, Spalte 8 mit der Shell-
Eingabeaufforderung wie abgebildet
angezeigt.
```

Wahr

Schlüsselwort

Beschreibung: Gibt True zurück, wenn die ausgeführte Anweisung True ist. „True“ steht für einen wahren Wert bei Objekten des Typs „bool“.

Beispiel:

```
>>>64>=32
Wahr
```

2nd [test] (über math)

2nd [catalog]

[Fns…] > Ops
A:True

[a A #]

trunc()

Modul: math

Syntax: trunc(x)

Beschreibung: Gibt den Zahlenwert x zurück, abgeschnitten auf eine Ganzzahl.

Beispiel:

```
>>>from math import *
>>>trunc(435.867)
435
```

math Modul
1:math…
0:trunc()

2nd [catalog]

[Fns…] > Modul
1:math…
0:trunc()

Importbefehle finden Sie unter 2nd [catalog].

try:

Schlüsselwort

Beschreibung: Verwenden Sie „try code block“, um den Codeblock auf Fehler zu testen. Wird auch zusammen mit „except“ und „finally“ verwendet. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

2nd [catalog]

tuple(sequence)

Modul: Built-in

2nd [catalog]

Syntax: tuple(sequence)

Beschreibung: Konvertiert eine Folge in ein Tupel um. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

Beispiel:

```
>>>a=[10,20,30]
>>>tuple(a)
(10,20,30)
```

type()

Modul: Built-in

2nd [catalog]

Syntax: type(object)

[Fns...]>Type>6:type ()

Beschreibung: Gibt die Typ des Objekts zurück.

Beispiel:

```
>>>a=1.25
>>>print(type(a))
<class 'float'>
>>>b=100
>>>print(type(b))
<class 'int'>
>>>a=10+2j
>>>print(type(c))
<class 'complex'>
```

uniform(min,max)

Modul: random

Syntax: uniform(min,max)

Beschreibung: Liefert eine Zufallszahl x (float) zurück, wobei min <= x <= max.

Beispiel:

```
>>>from random import *
>>>uniform(0,1)
0.476118
>>>uniform(10,20)
16.2787
```

Die Ergebnisse variieren aufgrund der zufälligen Ausgabe.

[math] Modul 2:random… Random 3:uniform (min,max)

[2nd] [catalog]

[Fns…] > Modul 2:random… Random 3:uniform (min,max)

Importbefehle finden Sie unter [2nd] [catalog].

wait_key()

Modul: ti_system

2nd [catalog]

Syntax: wait_key()

Beschreibung: Gibt einen kombinierten Tastencode zurück, der sich aus der gedrückten Taste und 2nd und/oder alpha zusammensetzt. Die Methode wartet, bis eine Taste gedrückt wird, bevor sie zum Programm zurückkehrt.

Beispiel:

Siehe:[Fns…]>Modul: time and ti_system modules.

while condition:

Schlüsselwort

[Fns…] Ctl 8:while condition:

Syntax: while condition:

Beschreibung: Führt die Anweisungen im folgenden Codeblock aus, bis „condition“ als „False“ ausgewertet wird.

2nd [catalog]

Beispiel:

```
>>> x=5
>>> while x<8:
… x=x+1
… print(x)
…
…
6
7
8
```

window(xmin,xmax,ymin,ymax)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.window(xmin,xmax,ymin,ymax)

[Fns...]>Modul oder math 5:ti_plotlib...> Setup 4:window()

Beschreibung: Definiert das Plotting-Fenster, indem es das angegebene horizontale Intervall (xmin, xmax) und das angegebene vertikale Intervall (ymin, ymax) auf den zugeteilten Plotting-Bereich (Pixel) abbildet.

Diese Methode muss ausgeführt werden, bevor andere Befehle des Moduls ti_plotlib ausgeführt werden.

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Die ti_plotlib-Eigenschaften vars, xmin, xmax, ymin, ymax werden auf die Argumentwerte aktualisiert. Die Standardwerte sind (-10, 10, -6.56, 6.56).

Beispiel:

Siehe Beispielprogramm: GRAPH.

with

Schlüsselwort

2nd [catalog]

Beschreibung: Weitere Einzelheiten finden Sie in der Python-Dokumentation.

xmax default 10.00

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.xmax default 10.00

[Fns...]>Modul oder
math
5:ti_plotlib...>
Properties
2:xmax

Beschreibung: Festgelegte Variable für Fensterargumente, definiert als plt.xmax.

Standardwerte:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Beispiel:

Siehe Beispielprogramm: GRAPH.

xmin default -10.00

Modul: ti_plotlib

Syntax: plt.xmin default -10.00

Beschreibung: Festgelegte Variable für Fensterargumente, definiert als plt.xmin.

Standardwerte:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Beispiel:

Siehe Beispielprogramm: GRAPH.

2nd [catalog]

[Fns...]>Modul oder
math
5:ti_plotlib...>
Properties
1:xmin

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

yield

Schlüsselwort

2nd [catalog]

Beschreibung: Verwenden Sie „yield“ (ergibt), um eine Funktion zu beenden. Es wird ein Generator ausgegeben. Weitere Einzelheiten finden Sie in der Python-Dokumentation.

ymax default 6.56

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.ymax default 6.56

[Fns...]>Modul oder math 5:ti_plotlib...> Properties 4:ymax

Beschreibung: Festgelegte Variable für Fensterargumente, definiert als plt.ymax.

Standardwerte:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Beispiel:

Siehe Beispielprogramm: GRAPH.

ymin **default** **-6.56**

Modul: ti_plotlib

Syntax: plt.ymin default -6.56

Beschreibung: Festgelegte Variable für Fensterargumente, definiert als plt.ymin.

Standardwerte:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Beispiel:

Siehe Beispielprogramm: GRAPH.

2nd [catalog]

[Fns...]>Modul oder
math
5:ti_plotlib...>
Properties
3:ymin

Importbefehle finden Sie unter 2nd [catalog] oder im Menü ti_plotlib Setup.

Sonderzeichen

@

Operator alpha [θ] (über 3)

Beschreibung: Decorator – Einzelheiten entnehmen Sie bitte der allgemeinen Python-Dokumentation.

2nd [catalog]

<<

Operator 2nd [catalog]

Syntax: x<<n

Beschreibung: Bitweises Verschieben nach links um n Bits.

>>

Operator 2nd [catalog]

Syntax: x>>n

Beschreibung: Bitweises Verschieben nach rechts um n Bits.

|

Operator 2nd [catalog]

Syntax: x|y

Beschreibung: Bitweises oder.

&

Operator 2nd [catalog]

Syntax: x&y

Beschreibung: Bitweises und.

^

Operator 2nd [catalog]

Syntax: x^y

Beschreibung: Bitweises exklusives oder.

~

Operator 2nd [catalog]

Syntax: ~x

Beschreibung: Bitweises nicht; die Bits von x invertiert.

x<=y

Operator

Syntax: x<=y

Beschreibung: Vergleich; x ist kleiner oder gleich y.

Beispiel:

```
>>>2<=5
Wahr
>>>3<=0
Falsch
```

math
1:math > Ops
7:x<=y

2nd [catalog]

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Operator

Syntax: x<y

Beschreibung: Vergleich; x ist strikt kleiner als y.

Beispiel:

```
>>>6<10
Wahr
>>>12<-15
Falsch
```

math
1:math > Ops
6:x<y

2nd [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Operator

Syntax: x>=y

Beschreibung: Vergleich; x ist größer oder gleich y.

Beispiel:

```
>>>35>=25
Wahr
>>>14>=65
Falsch
```

math
1:math > Ops
5:x>=y

2nd [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Operator

Syntax: x>y

Beschreibung: Vergleich; x ist strikt größer als y.

Beispiel:

```
>>>35>25
Wahr
>>>14>65
Falsch
```

math
1:math > Ops
4:x>y

2nd [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Operator

Syntax: x!=y

Beschreibung: Vergleich; x ist ungleich zu y.

Beispiel:

```
>>>35!=25
Wahr
>>>14!=10+4
Falsch
```

math
1:math > Ops
3:x!=y

2nd [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Operator

Syntax: x==y

Beschreibung: Vergleich; x ist gleich y.

Beispiel:

```
>>>75==25+50
Wahr
>>>1/3==0.333333
Falsch
>>>1/3==0.3333333 #gleich dem gespeicherten Python-Wert
Wahr
```

math
1:math > Ops
2:x==y

2nd [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Operator

Syntax: x=y

Beschreibung: y ist gespeichert in der Variablen x

Beispiel:

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3 #Verwenden Sie [ ^ ] auf der Tastatur für **
>>>print(B)
8
```

sto→

math
1:math > Ops
1:x=y

2nd [catalog]

[Fns...] > Ops
1:x=y

[a A #]

\

Trennzeichen

Beschreibung: Umgekehrter Schrägstrich (Backslash).

2nd [catalog]

[a A #]

\t

Trennzeichen

Beschreibung: Tabulatorschritt zwischen Strings oder Zeichen.

2nd [catalog]

\n

Trennzeichen

Beschreibung: Neue Zeile zur übersichtlichen Darstellung des Strings auf dem Bildschirm.

2nd [catalog]

' '

Trennzeichen

Beschreibung: Es werden zwei einzelne Anführungszeichen eingefügt.

Beispiel:

```
>>>eval('a+10')
17
```

2nd [mem] (über +)

2nd [catalog]

[a A #]

" "

Trennzeichen

Beschreibung: Es werden zwei doppelte Anführungszeichen eingefügt.

Beispiel:

```
>>>print("Ok")
```

alpha ["] (über +)

2nd [catalog]

[a A #]

Anhang

Integriert

Integriert	Integriert	Integriert
__name__	abs -- <function>	BaseException -- <class 'BaseException'>
__build_class__ -- <function>	all -- <function>	ArithmeticError -- <class 'ArithmeticError'>
__import__ -- <function>	any -- <function>	AssertionError -- <class 'AssertionError'>
__repl_print__ -- <function>	bin -- <function>	AttributeError -- <class 'AttributeError'>
bool -- <class 'bool'>	callable -- <function>	EOFError -- <class 'EOFError'>
bytes -- <class 'bytes'>	chr -- <function>	Exception -- <class 'Exception'>
bytearray -- <class 'bytearray'>	dir -- <function>	GeneratorExit -- <class 'GeneratorExit'>
dict -- <class 'dict'>	divmod -- <function>	ImportError -- <class 'ImportError'>
enumerate -- <class 'enumerate'>	eval -- <function>	IndentationError -- <class 'IndentationError'>
filter -- <class 'filter'>	exec -- <function>	IndexError -- <class 'IndexError'>
float -- <class 'float'>	getattr -- <function>	KeyboardInterrupt -- <class 'KeyboardInterrupt'>
int -- <class 'int'>	setattr -- <function>	ReloadException -- <class

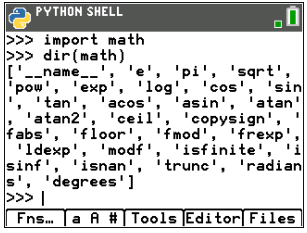
Integriert	Integriert	Integriert
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <class 'MemoryError'>
object -- <class 'object'>	help -- <function>	NameError -- <class 'NameError'>
property -- <class 'property'>	hex -- <function>	NotImplementedError -- <class 'NotImplementedError'>
range -- <class 'range'>	id -- <function>	OSError -- <class 'OSError'>
set -- <class 'set'>	input -- <function>	OverflowError -- <class 'OverflowError'>
slice -- <class 'slice'>	isinstance -- <function>	RuntimeError -- <class 'RuntimeError'>
str -- <class 'str'>	issubclass -- <function>	StopIteration -- <class 'StopIteration'>
super -- <class 'super'>	iter -- <function>	SyntaxError -- <class 'SyntaxError'>
tuple -- <class 'tuple'>	len -- <function>	SystemExit -- <class 'SystemExit'>
type -- <class 'type'>	locals -- <function>	TypeError -- <class 'TypeError'>
zip -- <class 'zip'>	max -- <function>	UnicodeError -- <class 'UnicodeError'>
classmethod -- <class 'classmethod'>	min -- <function>	ValueError -- <class 'ValueError'>
staticmethod -- <class 'staticmethod'>	next -- <function>	ZeroDivisionError -- <class 'ZeroDivisionError'>

Integriert	Integriert	Integriert
Ellipsis -- Ellipsis	oct -- <function>	
	ord -- <function>	
	pow -- <function>	
	print -- <function>	
	repr -- <function>	
	round -- <function>	
	sorted -- <function>	
	sum -- <function>	

Schlüsselwörter

Schlüsselwörter	Schlüsselwörter	Schlüsselwörter
False	elif	lambda
None	else	nonlocal
True	except	not
and	finally	or
as	for	pass
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield

math



math	math	math
__name__	acos -- <function>	frexp -- <function>
e -- 2.71828	asin -- <function>	ldexp -- <function>
pi -- 3.14159	atan -- <function>	modf -- <function>
sqrt -- <function>	atan2 -- <function>	isfinite -- <function>
pow -- <function>	ceil -- <function>	isinf -- <function>
exp -- <function>	copysign -- <function>	isnan -- <function>
log -- <function>	fabs -- <function>	trunc -- <function>
cos -- <function>	floor -- <function>	radians -- <function>
sin -- <function>	fmod -- <function>	degrees -- <function>
tan -- <function>		

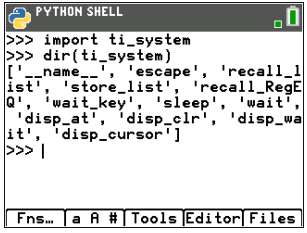
random

random	random	random
__name__	randint -- <function>	
seed -- <function>	choice -- <function>	
getrandbits -- <function>	random -- <function>	
randrange -- <function>	uniform -- <function>	

time

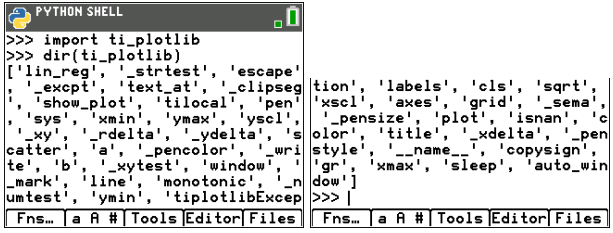
time	time	time
__name__		
monotonic		
sleep		
struc_time		

ti_system



ti_system	ti_system	ti_system
__name__	recall_RegEQ	disp_at
escape	wait_key	disp_clr
recall_list	sleep	disp_wait
store_list	wait	disp_cursor

ti_plotlib



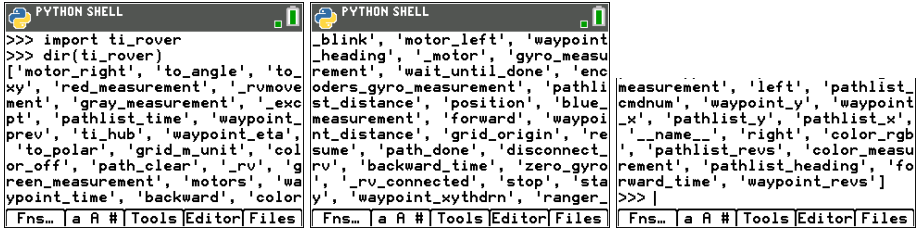
ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle

ti_plotlib	ti_plotlib	ti_plotlib
sys	ymin	copysign
xmin	tiplotlibException	gr
ymax	lables	xmax
yscl	cls	sleep
_xy	sqrt	auto_window
_rdelta	xscl	
_ydelta	axes	
scatter		

ti_hub

ti_hub	ti_hub	ti_hub
__name__	version	last_error
connect	begin	sleep
disconnect	start	tihubException
set	about	wait
read	isti	
calibrate	what	
range	who	

ti_rover



ti_rover	ti_rover	ti_rover
__name__	color_blink	_rv
motor_right	motor_left	stay
to_angle	waypoint_heading	waypoint_xythdrn
to_xy	_motor	ranger_measurement
red_measurment	gyro_measutrment	left
rvmovement	wait_until_done	pathlist_cmdnum
gray_measurment	encoders_gyro_measurement	waypoint_y
_excpt	pathlist_distance	waypoint-x
ti_hub	position	pathlist_y
waypoint_prev	blue_measurement	pathlist_x

ti_rover	ti_rover	ti_rover
pathlist_time	forward	right
waypoint_revs	waypoint_distance	color_rgb
to_polar	grid_origin	pathlist-revs
waypoint_eta	resume	color_measurement
color_off	path_done	tiroverException
grid_m_unit	disconnect_rv	forward_time
path_clear	backward_time	pathlist_heading
green_measurement	zero-gyro	
waypoint_time	_rv_connected	
motors	stop	
backward		

.

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.