Programmeren met Python op de TI-84 Plus CE-T *Python Edition* Grafische rekenmachine

Versie 84CE Bundle 5.6.0.

Lees meer over TI-technologie via de online help op education.ti.com/eguide.

Belangrijke informatie

Tenzij anderszins uitdrukkelijk vermeld in de Licentie bij een programma, geeft Texas Instruments geen garantie, expliciet dan wel impliciet, met inbegrip van maar niet beperkt tot willekeurig welke impliciete garanties van verhandelbaarheid en geschiktheid voor een bepaald doel met betrekking tot welke programma's of boekmaterialen dan ook, en stelt dergelijke materialen uitsluitend beschikbaar "zoals ze zijn". Texas Instruments is in geen enkel geval aansprakelijk voor speciale, indirecte, incidentele of voortvloeiende schade in verband met of voortkomend uit de aankoop of het gebruik van deze materialen, en de enige en uitsluitende aansprakelijkheid van Texas Instruments, ongeacht de actievorm, is niet hoger dan het bedrag dat vermeld is in de licentie voor het programma. Voorts is Texas Instruments niet aansprakelijk voor welke eis van welke aard dan ook tegen het gebruik van deze materialen door enige andere partij.

"Python" en de Python-logo's zijn handelsmerken of geregistreerde handelsmerken van de Python Software Foundation, gebruikt door Texas Instruments Incorporated met toestemming van de Foundation.

Inhoud

Wat is nieuw

Wat is nieuw in de Python Programming App v5.5.0

TI-84 Plus CE-T *Python Edition*

Programmeren met Python

TI-84 Plus CE-T *Python Edition*

- Ondersteunt programmeren in Python met behulp van de Python App uit de 84CE Bundle v5.6.0. Update naar de nieuwste versie op education.ti.com/84cetupdate.
- Open de Python App met 2nd apps of prgm wanneer de Python App geladen is.

Opmerking: wat is uw CE-rekenmachine-ervaring met TI-Python?

- TI-84 Plus CE-T *Python Edition* met 84CE Bundle v5.6.0 of hoger

---

Python-programma's overzenden

Bij het overzenden van Python-programma's van een niet-TI-platform naar een TI-platform OF van het ene TI-product naar een ander:

- Python-programma's die functies uit de hoofdtalen en standaard bibliotheken gebruiken (math, random enz.) kunnen zonder veranderingen worden overgezonden.
  Opmerking: Lijsten mogen maximaal 100 elementen bevatten.
- Programma's die platform-specifieke bibliotheken voor TI-platforms gebruiken - matplotlib (voor pc), ti_plotlib/ti_system/ti_hub/etc., moeten bewerkt worden voordat ze kunnen worden uitgevoerd op een ander platform.

Dit kan ook het geval zijn tussen TI-platformen.

Nieuwe functies en TI-Python-modules

- Ondersteuning van complexe getaltypen zoals a+bj.
  - Zie het menu [Fns…] Types in de Editor of Shell.
- tijdmodule
- TI-modules
  - ti_system
    - Roep een OS-lijst en OS-regressievergelijking op in een Python-programma. Creëer lijsten in een Python-programma en sla deze op in OS-lijstvariabelen. Lijsten mogen maximaal 100 elementen bevatten.
  - ti_plotlib
    - Voer Python-programma's uit om statistische en functieplots weer te geven.
  - ti_hub
    - Creëer TI-Innovator™ Hub Python-programma's.
  - ti_rover

– Bestuur TI-Innovator™ Rover met behulp van programmeren in Python.

Creëer “New” (nieuwe) programma “Types” (soorten) met templates

Wanneer er in uw programma noodzakelijke importopdrachten vereist zijn voor modules, gebruik dan de tab Types wanneer u een nieuw programma creëert. Essentiële programmaregels worden vooraf in uw nieuwe programma geplakt in de Editor. Dit is vooral handig voor STEM-activiteiten! De template voor plot-methode ondersteunt de eerste ervaring met het schrijven van een programma met ti_plotlib.

Ondersteuning bij het opgeven van argumenten (Argument helper) en menuschermtips

Een argumenthulp helpt u bij het selecteren van het juiste argument uit een menu, wanneer methodes stringargumenten bevatten. Typen is niet nodig! Opzoeken van de correcte string is niet nodig!

Menuschermtips worden gegeven bij reeksen van argumenten, standaardinstellingen of als hints bij toetsaanslagen.

Updates van het toetsenblok van de Python App

math blijft alle beschikbare modules weergeven.

2nd [i] (boven [.]) geeft het imaginaire deel j weer bij het complexe getal a+bj in Python.

Zie ook: Toetsenblok

Software-informatie

TI Connect™ CE

Ondersteuning van connectiviteit en *.py <> PY AppVar-conversie voor de TI-84 Plus CE-T *Python Edition*.

TI-SmartView™ CE-T

De TI-84 Plus CE-T *Python Edition*-emulator ondersteunt Python App v5.5.0

De voorbeeldprogramma's HELLO, GRAPH en LINREGR worden geladen na het installeren en resetten

Data Import Wizard converteert correct opgemaakte*. csv-bestanden naar rekenmachinelijsten voor de CE-emulator. Deze functie is handig wanneer u de module ti_system en externe data gebruikt voor het programmeren in Python.

- Als decimale getallen worden weergegeven met een komma in het *.csv bestand, dan wordt het bestand niet geconverteerd met de Data Import Wizard. Controleer de getalnotatie (format) van het besturingssysteem van uw computer en converteer het *.csv bestand zodat het de weergave met

decimale punt gebruikt. De lijst- en matrixeditor van de CE-rekenmachine gebruikt bijvoorbeeld de getalnotatie 12.34 en niet 12,34.

Opmerking: om TI-Innovator™ Hub- of TI-Innovator™ Rover-programma's uit te voeren, moet u de programma's naar de rekenmachine verzenden via TI Connect™ CE. Sluit de Python App af voordat u een Emulator Explorer-verzending naar de computer en vervolgens naar de rekenmachine uitvoert.
TI-Innovator™ Hub- en TI-Innovator™ Rover-programma's werken niet vanaf TI-SmartView™ CE-T.

Ga voor meer informatie over de nieuwe en bijgewerkte functionaliteit naar education.ti.com/84cetupdate.

Python App

Zie de volgende onderwerpen voor het gebruiken, navigeren en uitvoeren van de Python App.

- De Python App gebruiken
- Navigatie in de Python App
- Voorbeeldactiviteit

De Python App gebruiken

De Python App is beschikbaar voor de TI-84 Plus CE-T *Python Edition*. De informatie in deze eGuide is bedoeld voor gebruik met de TI-84 Plus CE-T *Python Edition*, geüpdatet met de meest recente CE Bundle.

Wanneer u de Python App voor het eerst uitvoert op uw TI-84 Plus CE-T *Python Edition*, kan de App u verzoeken om te updaten naar de meest recente CE Bundle voor de meest recente Python App.
Kijk op education.ti.com/84cetupdate voor het updaten van uw TI-84 Plus CE-T *Python Edition*.

De Python App biedt een File Manager, een Editor voor het creëren van programma's en een Shell voor het uitvoeren van programma's en te communiceren met de Python-interpreter. Python-programma's die opgeslagen of gecreëerd zijn als Python AppVars worden uitgevoerd vanuit het RAM-geheugen. Archiveer Python AppVars via het geheugenbeheerscherm van het OS als hulp bij opslagbeheer van Python-bestanden.

Opmerking: als uw rekenmachine een TI-84 Plus CE-T is, zie dan education.ti.com/84cetupdate om de meest recente informatie voor uw CE te vinden.

Navigatie in de Python App

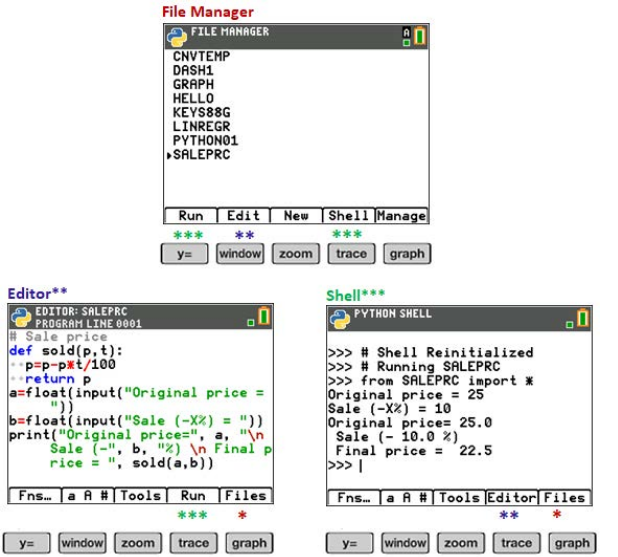
Gebruik de sneltoetsen op het scherm in de App om te navigeren tussen werkruimtes in de Python App. In de afbeelding geven de labels van de sneltabs het volgende aan:

* Navigatie naar de File Manager [Files]

** Navigatie naar de Editor [Edit] of [Editor]

*** Navigatie naar de Shell [Shell]

Krijg toegang tot sneltabs op het scherm met de rij grafische toetsen direct onder het scherm. Zie ook Toetsenblok. Het menu Editor>Tools en het menu Shell>Tools bevatten ook navigatie-acties.



Voorbeeldactiviteit

Gebruik de gepresenteerde voorbeeldactiviteit om ervaring op te doen en vertrouwd te raken met de werkruimtes in de Python App.

- Creëer een nieuw programma vanuit de File Manager
- Schrijf het programma in de Editor
- Voer het programma uit in de Shell in de Python App.

Zie voor meer informatie over programmeren met Python op uw CE de hulpbronnen voor de
TI-84 Plus CE-T *Python Edition*.

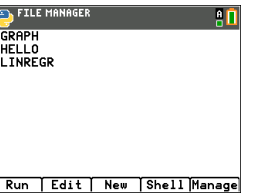
Aan de slag:

- Voer de Python App uit.

Opmerking: De werkelijke schermen kunnen enigszins afwijken van de geleverde beelden.

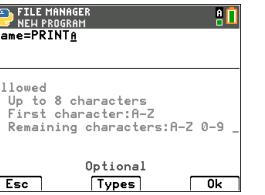
Voer de naam van het nieuwe programma in vanuit de File Manager.

- Druk op [zoom] ([New]) om een nieuw programma te creëren.



Invoeren van een nieuwe bestandsnaam

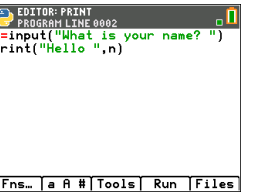
- Het voorbeeldprogramma wordt "PRINT" genoemd. Voer de naam van het programma in en druk op [graph] ([Ok]).
- Merk op dat de cursor in ALPHA-lock (Alfabetstand) staat. Voer een programmanaam altijd in volgens de gegeven vereisten op het scherm.



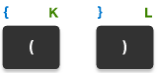
Tip: Als de cursor niet in ALPHA-lock staat, druk dan op [2nd] [alpha] [alpha] voor hoofdletters.

Voer het programma in zoals weergegeven.

Tip: de App biedt een snelle invoer! Houd altijd de status van de cursor in de gaten terwijl u het programma invoert!



Lettertekens op het toetsenblok	[alpha] wisselt de invoercursor-status in de Editor en Shell.

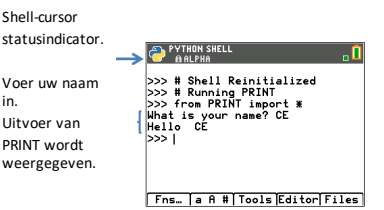
	_ geen letter een kleine letter een HOOFDLETTER
Waar zit het is-gelijkteken?	Druk op sto→ wanneer de cursor _ is. 
Waar zijn deze te vinden? input() print()	[Fns...] I/O 1:print() 2:input()
Waar zitten de dubbele aanhalingstekens?	alpha ["] 
Waar zitten (en)?	Gebruik het toetsenblok wanneer de cursor _ is. 

Probeer dit! [a A #] en 2nd [catalog] zijn ook handig voor snelle invoer waar nodig!

Voer het programma PRINT uit

- Druk vanuit de Editor op trace ([Run]) om uw programma uit te voeren in de Shell.
- Voer uw naam in bij de prompt "What is your name?".
- De uitvoer geeft "HELLO" weer met uw naam.

Opmerking: bij de Shell-prompt >>> kunt u een commando uitvoeren, zoals 2+3. Als u een methode uit math, random of andere beschikbare modules uitvoert, zorg dan dat u eerst een importmodule-opdracht uitvoert, net als in elke andere Python-programmeeromgeving.



Een Python-sessie met uw programma's instellen

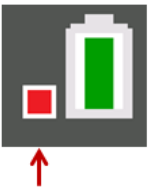
Wanneer de Python App wordt gestart, wordt de CE-verbinding met de TI-Python-omgeving gesynchroniseerd voor uw huidige Python-sessie. U ziet een uw lijst met programma's in het RAM-geheugen en dynamische modules, terwijl ze gesynchroniseerd worden met de Python-omgeving.

Wanneer de Python-sessie gerealiseerd is, bevat de statusbalk een groene vierkante indicator naast het batterijpictogram, die aangeeft dat de Python-sessie gereed is voor gebruik. Als de indicator rood is, wacht dan tot hij weer groen wordt wanneer de Python-omgeving weer beschikbaar is.

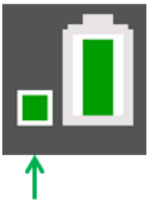
U kunt een update van de Python-uitgave zien wanneer u de Python App samen met programmasynchronisatie start na de meest recente update voor uw TI-84 Plus CE-T *Python Edition* vanaf education.ti.com/84cetupdate.

De Python App loskoppelen en weer aansluiten

Wanneer de Python App wordt uitgevoerd, bevat de statusbalk een indicator die aangeeft of Python gereed is voor gebruik. Het CE-toetsenblok reageert mogelijk niet, totdat de verbinding tot stand is gebracht. Het is een goede gewoonte om de verbindingsindicator op de statusbalk in de gaten te houden terwijl u in uw Python-sessie bent.



Python niet gereed



Python gereed

Schermvastleggingen

Met behulp van TI Connect™ CE op education.ti.com/84cetupdate is het maken van schermafbeeldingen van elk willekeurig scherm van de Python App toegestaan.

Python-werkruimtes

De Python App bevat drie werkruimtes voor het ontwikkelen van uw Python-programma's.

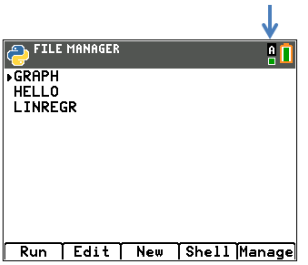
- File Manager
- Editor
- Shell

Python File Manager

De File Manager geeft de Python AppVars, die beschikbaar zijn in het RAM op uw rekenmachine, weer in een lijst. U kunt programma's creëren, bewerken en uitvoeren en u kunt navigeren naar de Shell.

Wanneer u in de alfabet-modus bent, druk dan op een willekeurige letter op het toetsenblok om naar programma's te springen die beginnen met die letter.

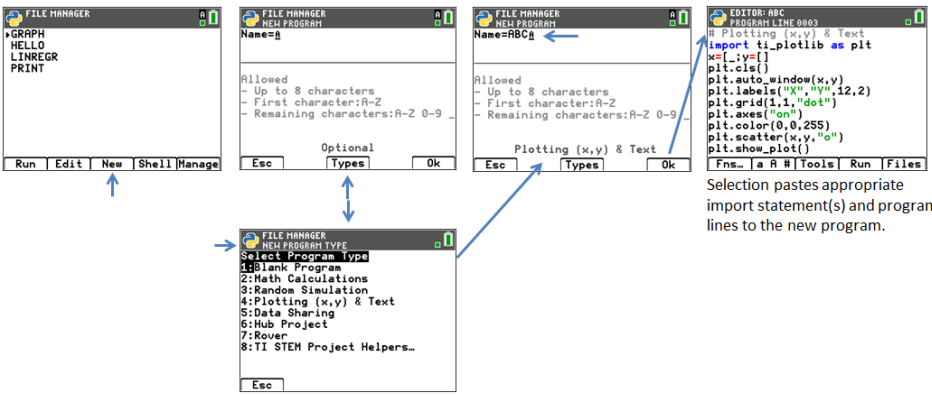
Druk indien nodig op alpha wanneer de indicator **A** niet wordt weergegeven op de statusbalk.



Sneltoetsen en menu's van de File Manager		
Menu's	**Toetsaanslag**	**Beschrijving**
[Run]	y=	Selecteer een programma met ▲ of ▼. Selecteer vervolgens [Run] om uw programma uit te voeren.
[Edit]	window	Selecteer een programma met ▲ of ▼. Selecteer vervolgens [Edit] om het programma weer te geven in de Editor en uw programma te bewerken.
[New]	zoom	Selecteer [New] om een nieuw programma in te voeren en door te gaan naar de Editor om uw nieuwe programma in te voeren. Selecteer op het scherm [New] de optie [Types] (druk op [zoom]) om een type programma te selecteren. Door een type programma te selecteren, wordt er een template van importopdrachten en veelgebruikte functies en methodes in uw nieuwe programma voor geplakt.
[Shell]	trace	Selecteer [Shell] om de Shell-prompt weer te geven (Python-interpreter). De Shell bevindt zich in de huidige status.
[Manage]	graph	Selecteer [Manage] om:

Sneltoetsen en menu's van de File Manager		
Menu's	**Toetsaanslag**	**Beschrijving**
		• Het versienummer te bekijken. • Een geselecteerd programma te kopiëren, te wissen of het een andere naam te geven. • Het scherm About te bekijken. • De App af te sluiten. Gebruik ook [2nd] [quit]

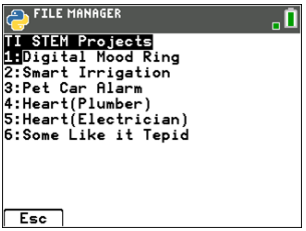
Een nieuw programma maken met templates voor programmatypen



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

Een nieuw programma met een STEM-activiteit maken met behulp van templates

Wanneer deTISTEMEN AppVar geladen is in het Archief, dan wordt het menuonderdeel “TI STEM Project Helpers...” weergegeven in het menu Select Program Type. Selecteer de gewenste template voor de STEM-activiteit als hulp bij het starten met een nieuw STEM-programma.



Python Editor

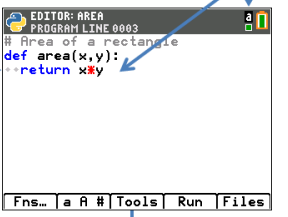
De Python Editor wordt weergegeven vanuit een geselecteerd programma in File Manager of vanaf de Shell. De Editor geeft trefwoorden, operatoren, opmerkingen, strings en inspringingen in kleur weer. Snel plakken van veelgebruikte trefwoorden en functies van Python is mogelijk, evenals directe invoer met het toetsenblok en invoer van speciale tekens met [a A #]. Wanneer u een programmablok zoals if.. elif.. else plakt, biedt de Editor automatische inspringing, die naar wens kan worden aangepast terwijl u uw programma schrijft.

De cursor is altijd in de invoegmodus. Gebruik 2nd en alpha om de cursor te wisselen. Cursorstatussen zijn: numeriek, a en A. [del] werkt als een backspace en delete van een teken.

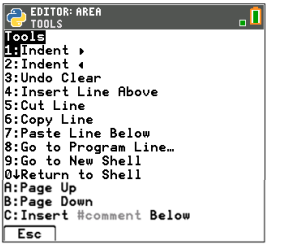
Programmaregellocatie van de cursor.

Automatische inspringing van programmablokken.

Grijze stippen als visuele indicator van ingesprongen regels.



Handige tools voor bewerken en werken in de Shell. Volledige beschrijving hieronder.



Sneltoetsen en menu's van de Python Editor		
Menu's	**Toetsindruk**	**Beschrijving**
[Fns…]	y=	Selecteer [Fns…] om menu's van veelgebruikte functies, trefwoorden en operatoren te openen. Hiermee krijgt u ook toegang tot geselecteerde inhoud van de math- en random-modules. **Opmerking:** 2nd [catalog] is ook handig om snel te plakken.

<table>
<tr><th colspan="3">Sneltoetsen en menu's van de Python Editor</th></tr>
<tr><th>Menu's</th><th>Toetsindruk</th><th>Beschrijving</th></tr>
<tr><td>[a A #]</td><td>window</td><td>Selecteer [a A #] om het palet met speciale tekens te openen als alternatieve manier voor het invoeren van veel tekens.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Selecteer [Tools] om functies te openen die u helpen bij het bewerken of bij uw interactie met de Shell.
<table>
<tr><td>1: Indent ▶</td><td>Laat de programmaregel inspringen naar rechts. De cursor gaat naar het eerste teken van de regel.</td></tr>
<tr><td>2: Indent ◀</td><td>Laat de programmaregel terugspringen naar links. De cursor gaat naar het eerste teken van de regel.</td></tr>
<tr><td>3: Undo Clear</td><td>Plakt de laatste gewiste regel in een nieuwe regel onder de programmaregel die de cursor bevat. De cursor wordt weergegeven aan het einde van de geplakte regel.</td></tr>
<tr><td>4: Insert Line Above</td><td>Voegt een regel toe boven de programmaregel met de cursor. De regel springt in en er worden inspringingsstippen weergegeven, indien van toepassing.</td></tr>
<tr><td>5: Cut Line</td><td>De huidige programmaregel met de cursor wordt geknipt. De cursor wordt weergegeven op de programmaregel onder de geknipte regel.</td></tr>
<tr><td>6: Copy Line</td><td>Kopieert de huidige programmaregel met de cursor. Een gekopieerde programmaregel kan worden geplakt in de Shell-prompt. Zie Shell hieronder.</td></tr>
<tr><td>7: Paste Line Below</td><td>Plakt de laatst opgeslagen programmaregel in de regel onder de cursorpositie.</td></tr>
<tr><td>8: Go to Program Line...</td><td>Geeft de cursor weer aan het begin van de gespecificeerde programmaregel.</td></tr>
<tr><td>9: Go to New Shell</td><td>Geeft de geherinitialiseerde Shell weer.</td></tr>
</table></td></tr>
</table>

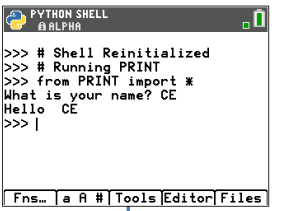
<table>
<tr><th colspan="3">Sneltoetsen en menu's van de Python Editor</th></tr>
<tr><th>Menu's</th><th>Toetsindruk</th><th>Beschrijving</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>0: Return to Shell</td><td>Geeft de Shell in de huidige status weer.</td></tr>
<tr><td>A: Page up</td><td>Geeft 11 programmaregels boven de huidige cursorpositie weer als beschikbaar.</td></tr>
<tr><td>B: Page Down</td><td>Geeft 11 programmaregels onder de huidige cursorpositie weer als beschikbaar.</td></tr>
<tr><td>C: Insert #comment Below</td><td>Voegt # in op een nieuwe regel onder de cursorpositie.</td></tr>
</table></td></tr>
<tr><td>[Run]</td><td>trace</td><td>Selecteer [Run] om uw programma uit te voeren.</td></tr>
<tr><td>[Files]</td><td>X</td><td>Selecteer [Files] om de File Manager weer te geven.</td></tr>
</table>

Python Shell

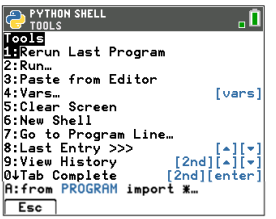
De Python Shell is de console waarin u kunt communiceren met de Python-interpreter of uw Python-programma's kunt uitvoeren. Hierin is snel plakken van veelgebruikte trefwoorden en functies van Python mogelijk, evenals directe invoer met het toetsenblok en invoer van speciale tekens met [a A #]. De Shell-prompt kan worden gebruik om één regel van een programma die geplakt is uit de Editor te testen. Er kunnen ook meerdere regels uit een programma worden in- en uitgevoerd bij de Shell-prompt >>>.

Statusindicator Shell-cursor.

De Shell initialiseert opnieuw, wanneer een nieuw programma wordt uitgevoerd.



Handige tools voor het werken in de Shell. Zie de informatie hieronder.



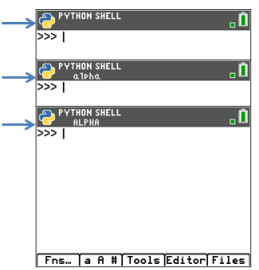
Statussen van de Shell-cursor

niet-alpha

2nd alpha indien nodig om te wisselen

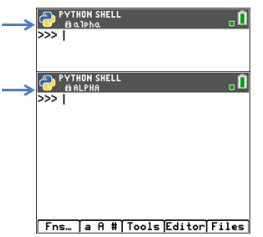
alpha alpha

nogmaals alpha ALPHA



2nd alpha lock alpha

nogmaals alpha lock ALPHA



<table>
<tr><th colspan="3">Sneltoetsen en menu's van de Python Shell</th></tr>
<tr><th>Menu's</th><th>Toetsaanslag</th><th>Beschrijving</th></tr>
<tr><td>[Fns…]</td><td>y=</td><td>Selecteer [Fns…] om menu's van veelgebruikte functies, trefwoorden en operatoren te openen. Hiermee krijgt u ook toegang tot geselecteerde inhoud van de math- en random-modules.
Opmerking: 2nd [catalog] is ook handig om snel te plakken.</td></tr>
<tr><td>[a A #]</td><td>window</td><td>Selecteer [a A #] om het palet met speciale tekens te openen als alternatieve manier voor het invoeren van een groot aantal tekens.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Selecteer [Tools] om de volgende menuonderdelen weer te geven.
<table>
<tr><td>1: Rerun last program</td><td>Voert het laatste programma dat werd uitgevoerd in de Shell opnieuw uit.</td></tr>
<tr><td>2: Run…</td><td>Geeft een lijst weer van de beschikbare Python-programma's om uit te voeren in de Shell.</td></tr>
<tr><td>3: Paste from Editor</td><td>Plakt de laatste gekopieerde programmaregel van de Editor naar de Shell-prompt.</td></tr>
<tr><td>4: Vars…</td><td>Geeft de variabelen weer uit het laatste programma dat is uitgevoerd. Geeft geen programma-gedefinieerde variabelen uit een geïmporteerd programma weer.</td></tr>
<tr><td>5: Clear Screen</td><td>Wist het scherm van de Shell. Herinitialiseert niet een nieuwe Shell.</td></tr>
<tr><td>6: New Shell</td><td>Herinitialiseert een nieuwe Shell.</td></tr>
<tr><td>7: Go to Program Line…</td><td>Geeft de Editor weer vanuit de Shell met de cursor op de gespecificeerde programmaregel.</td></tr>
<tr><td>8: Last Entry>>> ▲▼</td><td>Geeft tot maximaal de laatste 8 invoeren op de Shell-prompt weer tijdens een Shell-sessie.</td></tr>
<tr><td>9: View History 2nd ▲ 2nd ▼</td><td>Scrol door het Shell-scherm om tot maximaal de laatste 60 regels uitvoer in de Shell te bekijken tijdens een Shell-sessie.</td></tr>
<tr><td>0: Tab Complete 2nd [enter]</td><td>Geeft de namen van de variabelen en functies weer die beschikbaar zijn in de huidige Shell-sessie.</td></tr>
</table></td></tr>
</table>

Sneltoetsen en menu's van de Python Shell			
Menu's	**Toetsaanslag**	**Beschrijving**	
			Wanneer een letter van een beschikbare variabele of functie wordt ingevoerd, drukt u op 2nd [enter] om de naam automatisch aan te vullen als er een match is in de huidige Shell-sessie.
		A: from PROGRAM import *...	Wanneer het voor het eerst wordt uitgevoerd in een Shell-sessie, wordt PROGRAM uitgevoerd en zijn de variabelen alleen zichtbaar met behulp van Tab Complete. Wanneer het programma nogmaals wordt uitgevoerd in dezelfde Shell-sessie, dan verschijnt de uitvoering als geen uitvoering. Deze opdracht kan ook worden geplakt uit 2nd [catalog].
[Editor]	trace	Selecteer [Editor] om de Editor weer te geven met de laatste programma's in de Editor. Als de Editor leeg is, kunt u de File Manager weergeven.	
[Files]	graph	Selecteer [Files] om de File Manager weer te geven.	

Opmerking:

- Om een lopend Python-programma te onderbreken, bijvoorbeeld als een programma in een continue lus is, drukt u op on. U kunt ook op **[Tools]** (zoom) > **6:New Shell** drukken om een lopend programma te stoppen.
- Wanneer u de module ti_plotlib gebruikt om te plotten in het plotgebied in de Shell, druk dan op clear om de plot te wissen en terug te keren naar de Shell-prompt.

Uitvoeringsfout: ga naar programmaregel met Shell >Tools

De TI-Python-omgeving toont Python-foutmeldingen in de Shell wanneer een programma wordt uitgevoerd. Als er een fout verschijnt wanneer een programma wordt uitgevoerd, dan wordt er een programmaregelnummer weergegeven. Gebruik **Shell>Tools 7:Go to Program Line...** Voer het regelnummer in en druk op **[OK]**. De cursor wordt weergegeven op het eerste teken van de betreffende programmaregel in de Editor. Het programmaregelnummer wordt weergegeven op de tweede regel van de Statusbalk in de Editor.

Invoer - Toetsenblok, Catalogus, speciale tekens en menu's

Tips voor snelle invoer

- Toetsenblok
- Catalogus
- [a A #] speciale tekens
- [Fns…] menu's

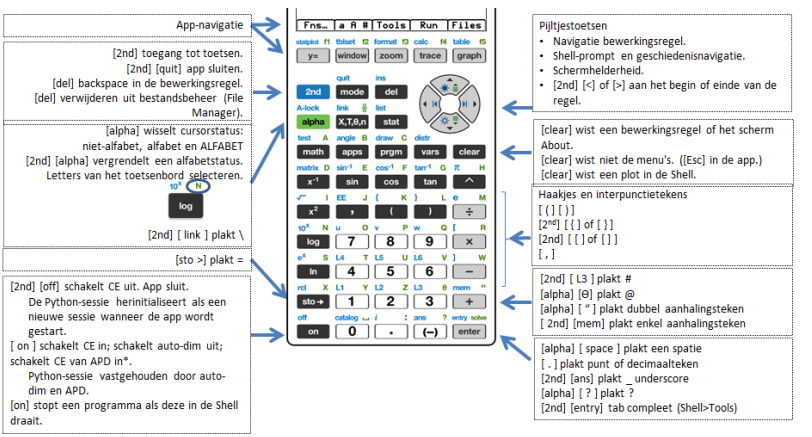
Het toetsenblok, de Catalogus, [a A #] en Fns… menu's gebruiken

Wanneer u programmacode invoert in de Editor of in de Shell, gebruik dan de volgende invoermethoden om snel te plakken op de bewerkingsregel.

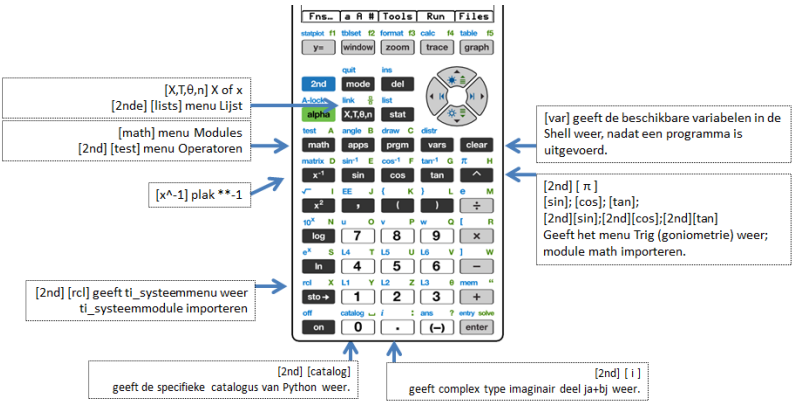
Toetsenblok

Wanneer de Python App wordt uitgevoerd, is het toetsenblok ontworpen om de juiste Python-bewerkingen te plakken of om menu's te openen die bedoeld zijn voor eenvoudige invoer van functies, trefwoorden, methodes, operatoren enz. Door op 2nd en alpha te drukken krijgt u toegang tot de tweede en derde functies op een toets zoals in het besturingssysteem.

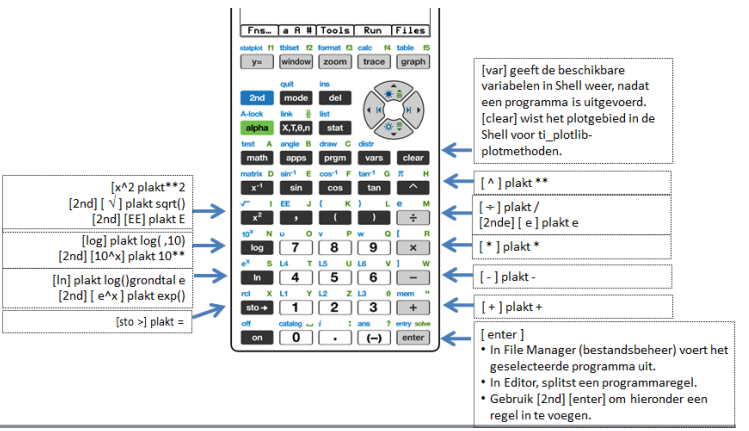
Navigatie, bewerken en speciale tekens per toetsenblokregel in de Python App



Specifieke toetsaanslagen voor menu's en functies per toetsenblokregel in de Python App



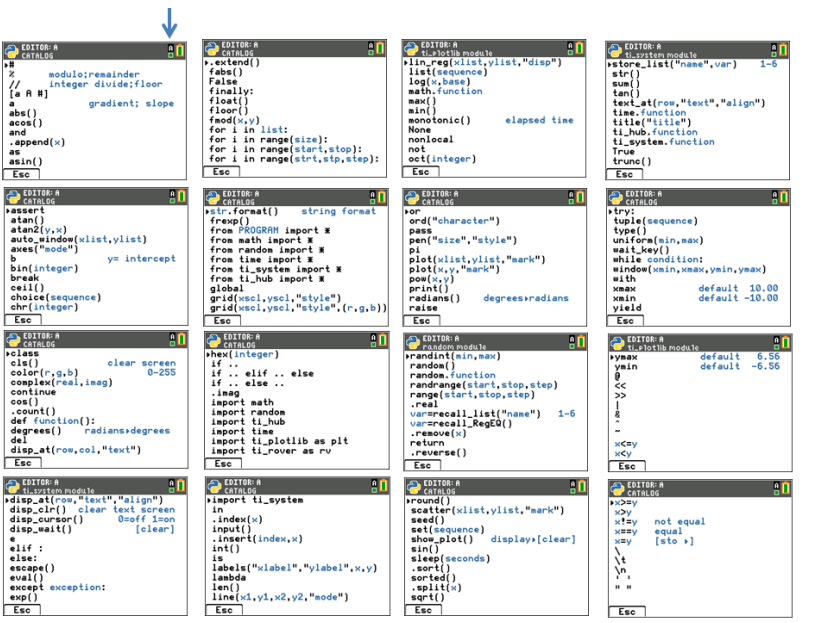
Specifieke toetsaanslagen voor menu's en functies per toetsenblokregel in de Python App (Vervolg)



Catalogus

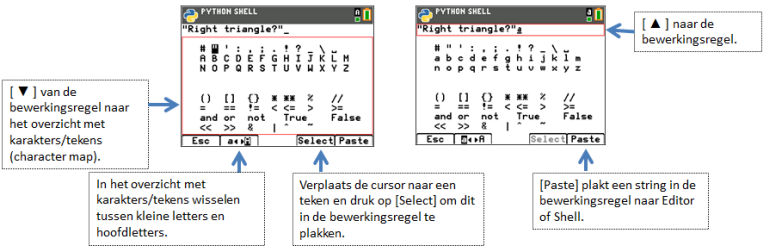
Wanneer de Python App wordt uitgevoerd, geeft [2nd] [catalog] een lijst met veelgebruikte scheidingstekens, trefwoorden, functies en operatoren die snel kunnen worden geplakt in een bewerkingsregel.
[2nd] [catalog] is alleen beschikbaar in de Editor en Shell. Zie voor een uitgebreidere beschrijving van elk onderdeel in de Catalogus de Handleiding. Druk bovenaan het catalogusmenu op [alpha] voor circulaire navigatie in de catalogus.

Selecteer in de Catalogus [alpha] en een lettertoets om de lijst beginnend bij die letter weer te geven.



Character Map [a A #]

De sneltab [a A #] die u brengt naar een palet met speciale tekens, is een handige functie voor het invoeren van strings (tekenreeksen) in de Editor of Shell.



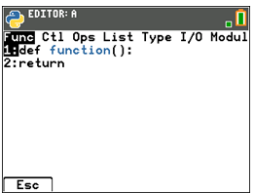
Opmerking: Wanneer de cursor-focus op de bewerkingsregel [a A #] ligt, zijn bepaalde toetsen van het toetsenblok niet beschikbaar. Wanneer de focus op de kaart met speciale tekens (character map) ligt, is het toetsenblok beperkt.

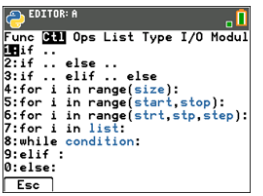
[Fns…] menu's

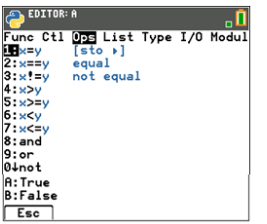
De sneltab [Fns…] geeft menu's weer met veelgebruikte functies, trefwoorden en operatoren van Python. Ook bieden de menu's toegang tot de geselecteerde functies en constanten uit de modules math en random. Hoewel u ook teken voor teken kunt invoeren vanaf het toetsenblok, bieden deze menu's een snelle manier om te plakken in de Editor of Shell. Druk op [Fns…] wanneer u in de Editor of Shell bent. Zie ook Catalogus en Toetsenblok voor alternatieve invoermethodes.

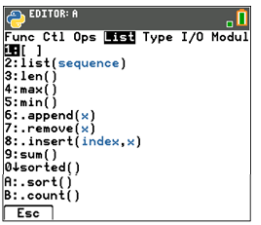
Submenu's van functies en modules

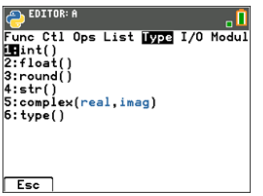
Ingebouwde functies, operatoren en trefwoorden

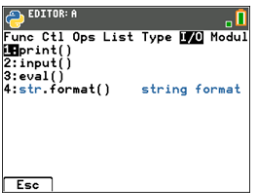










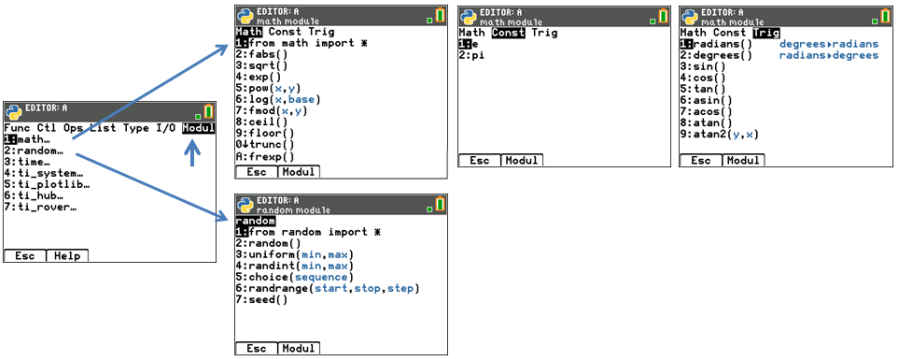


Submenu's van modules

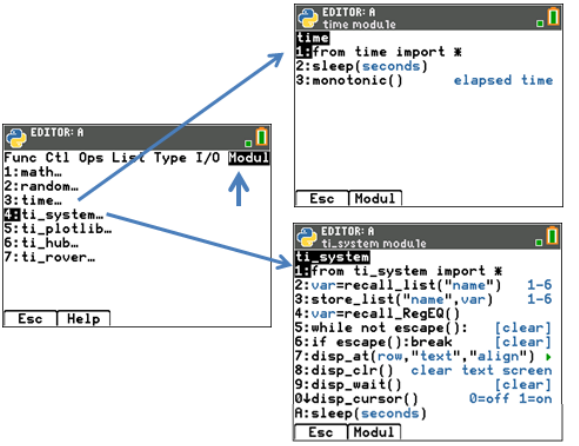
Wanneer u een Python-functie of constante uit een module gebruikt, moet u altijd een importopdracht gebruiken om de modulelocatie van de functie, methode of constante aan te geven.

Zie Wat is de Python-programmeeromgeving?

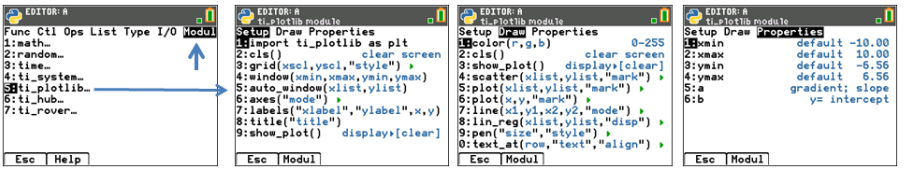
[Fns…]>Modul: math en random modules



[Fns…]>Modul: time en ti_system modules



[Fns…]>Modul: ti_plotlib



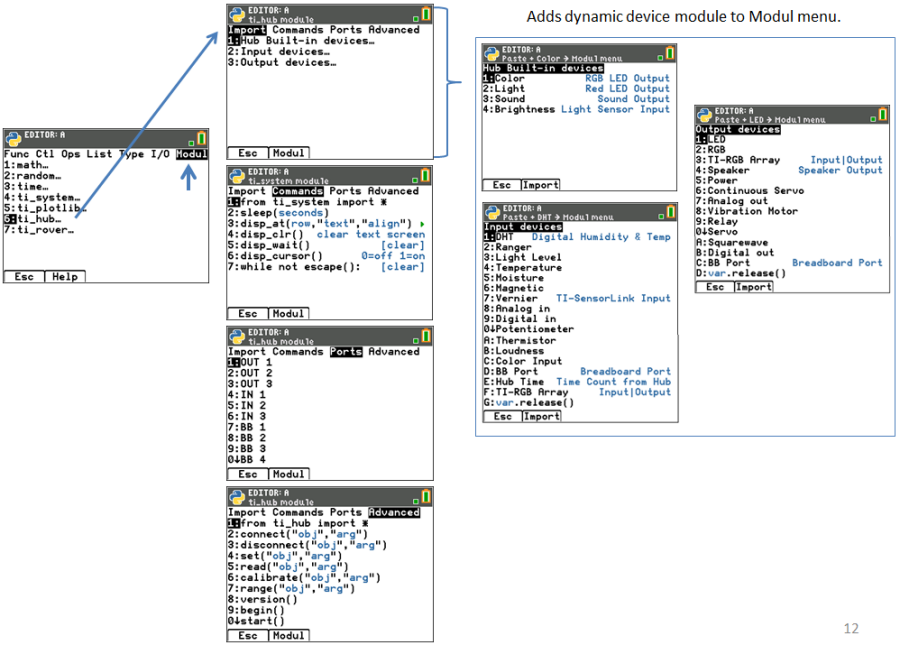
Belangrijke opmerking bij plotten:

- De volgorde van de programmaregels bij plotten moet hetzelfde zijn als de volgorde in het menu Setup om te verzekeren dat u de verwachte resultaten krijgt.
- Het plotten wordt weergegeven wanneer plt.show_plot() wordt uitgevoerd aan het einde van het plotten van objecten in een programma. Om het plotgebied in de Shell te wissen, drukt u op [clear].
- Het uitvoeren van een tweede programma dat ervan uitgaat dat de standaardwaarden zijn ingesteld binnen dezelfde Shell-omgeving, resulteert over het algemeen in onverwacht gedrag, zoals kleur of andere standaard argumentinstellingen. Bewerk programma's met verwachte argumentwaarden of herinitialiseer de Shell voordat u nog een plotprogramma uitvoert.

[Fns…]>Modul: ti_hub module

ti_hub methodes staan niet vermeld in de Catalogus en worden daarom niet vermeld in de Handleiding. Gebruik de informatie op het scherm in de menu's voor informatie over argumenten en standaard argumenten of toegestane waarden. Meer informatie over programmeren in Python voor
TI-Innovator™ Hub en TI-Innovator™ Rover is te vinden op education.ti.com.

Opmerking: TI-Innovator™ Hub moet aangesloten zijn wanneer u uw Python-programma's uitvoert.



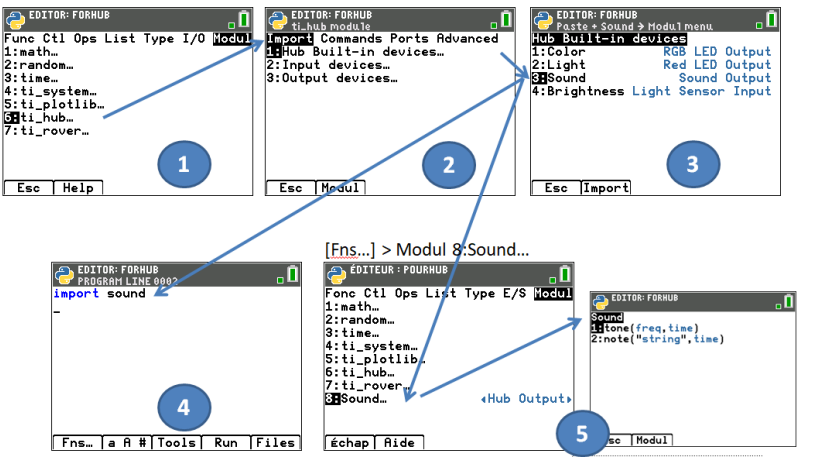
ti_hub module – voeg Import toe aan de Editor en voeg de ti_hub sensormodule toe aan het menu Modul

Schermvoorbeeld: Geluid importeren

Om TI-Innovator™ sensormethoden te importeren naar uw Python-programma, vanuit de Editor:

1. Selecteer **[Fns…] > Modul 6:ti_hub**
2. Selecteer het menu Import van ti_hub. Selecteer een sensortype uit Built-in, Input en Output.
3. Selecteer een sensor.
4. Er wordt een importopdracht geplakt in de Editor en de sensormodule wordt beschikbaar in **[Fns…] > Modul** wanneer u terugkeert naar dat menu vanuit uw programma.
5. Selecteer **[Fns…] > Modul 8:Sound…** om geschikte methoden voor deze sensor te plakken.

[Fns…]>Modul 6:ti_hub



Opmerking: Brightns is een "ingebouwd" object op de TI-Innovator Hub.

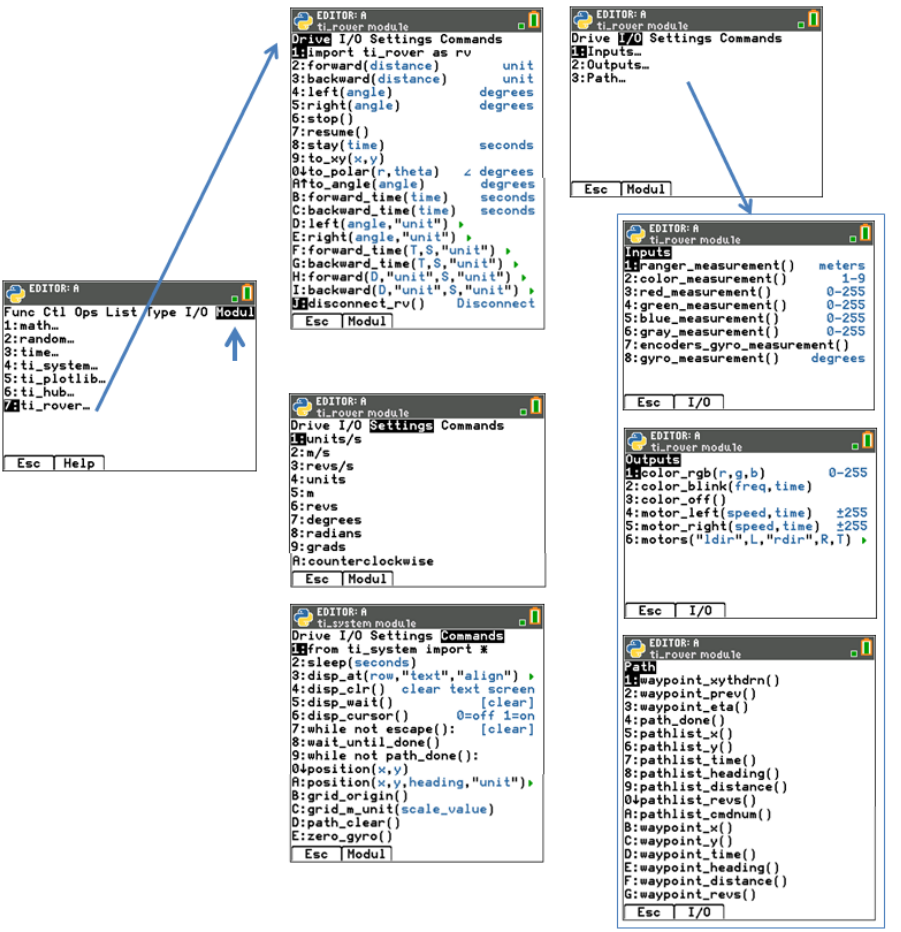
Wanneer u de opdracht 'import brightns' gebruikt, voer dan 'brightns.range(0,100)' in om het correcte standaardbereik aan het begin van de uitvoering van het programma te garanderen.

Voorbeeld:

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

---

ti_rover methoden staan niet vermeld in de Catalogus en worden daarom niet vermeld in de Handleiding. Gebruik de informatie op het scherm in de menu's voor details over argumenten en standaard argumenten of toegestane waarden. Meer informatie over programmeren in Python voor
TI-Innovator™ Hub en TI-Innovator™ Rover is te vinden op education.ti.com.



Opmerkingen:

- Bij programmeren in TI-Python hoeft u geen methodes op te nemen om TI-Innovator™ Rover aan te sluiten en los te koppelen. De TI-Innovator™ Rover Python-methodes kunnen omgaan met aansluiten en loskoppelen zonder aanvullende

methodes. Dit werkt een beetje anders dan het programmeren van TI-Innovator™ Rover in TI-Basic.

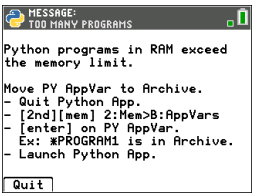
- rv.stop() wordt uitgevoerd als een pauze, waarna ‘resume’ verdergaat met de Rover-bewegingen in de wachtrij. Als er een andere bewegingsopdracht wordt uitgevoerd na
  rv.stop(), dan wordt de bewegingswachtrij gewist. Ook dit werkt een beetje anders dan het programmeren van TI-Innovator™ Rover in TI-Basic.

Berichten in de Python App

Er zijn verschillende berichten die kunnen verschijnen als u in een Python-sessie bent. Enkele geselecteerde berichten worden gegeven in de tabel. Volg de instructies op het scherm en navigeer zo nodig met [Quit], [Esc] of [Ok].

Geheugenbeheer

Het beschikbare geheugen voor de Python-omgeving is maximaal 100 Python-programma's (PY AppVars) of 50K aan geheugen. De modules die gebundeld zijn met de app in deze Python-uitgave, delen dezelfde ruimte met alle bestanden.



Gebruik [2nd] [quit] om de App af te sluiten

Er verschijnt een prompt of u zeker weet dat u de App wilt afsluiten. Als u de App afsluit, dan stopt uw Python-sessie. Wanneer u de Python App opnieuw uitvoert, dan worden uw Python AppVar-programma's en modules gesynchroniseerd. De Shell wordt opnieuw geïnitialiseerd.

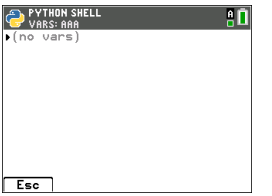
In File Manager drukt u op [del] in een geselecteerd Python-programma of selecteert u **File Manager>Manage 2:Delete Program...**.

U ziet een dialoogvenster om te wissen of terug te gaan naar de File Manager.

U heeft geprobeerd om een nieuw programma te creëren of om een Python-programma te dupliceren dat al bestaat op uw CE, in het RAM of Archief, of dat uitgeschakeld is in de toetsmodus. Voer een andere naam in.

U probeert vanuit de Shell naar de Editor te navigeren, maar de Editor is leeg. Selecteer een geschikte optie voor uw werk.

Wanneer u een Python-programma uitvoert, worden gedefinieerde variabelen uit het laatst uitgevoerde programma vermeld in het menu **Shell>Tools> 4:Vars…** om te gebruiken, en zijn deze beschikbaar voor gebruik in de Shell. Als er geen variabelen worden weergegeven, dan moet u uw programma misschien nogmaals uitvoeren.



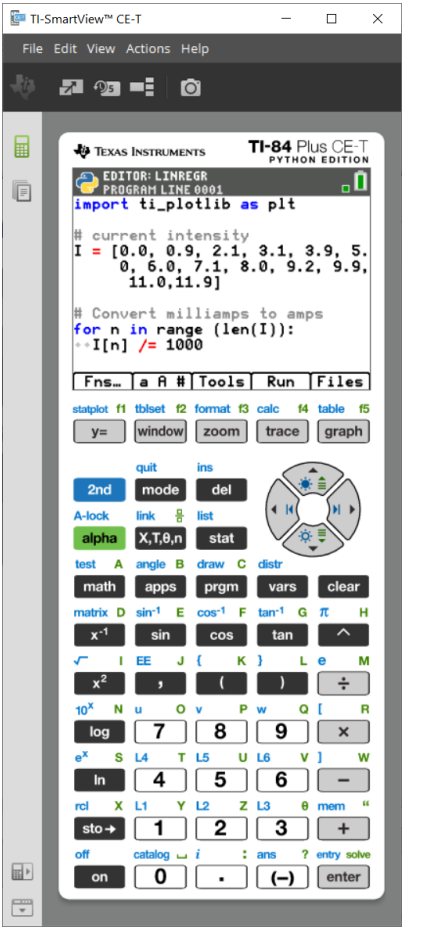
De TI-SmartView™ CE-T en de Python-omgeving gebruiken

In deze handleiding wordt uitgegaan van de nieuwste update van TI-SmartView™ CE-T. Update naar de nieuwste TI-SmartView™ CE-T op education.ti.com/84cetupdate.

De update bevat het nieuwste TI-84 Plus CE-T *Python Edition*-emulator-OS dat de nieuwste Python App uitvoert. De bijgewerkte modules van time, ti_system, ti_plotlib, ti_rover* en ti_hub* zijn erin opgenomen.

Voer de Python App uit op de TI-84 Plus CE-T *Python Edition*-emulator.

- De Python App biedt
  - File Manager
  - Editor
  - Uitvoering van uw Python-programma in de Shell*



Hub/Rover-programma's

- Creëer ti_hub/ti_rover Python-programma's in de CE-emulator met de Python App.

  *** Opmerking:** er is geen connectiviteit tussen TI-SmartView™ CE en TI-Innovator™ Hub of TI-Innovator™ Rover. Programma's kunnen worden gemaakt en vervolgens uitgevoerd op de CE-rekenmachine.

- Sluit de Python App af om het verzenden van de Python AppVar(s) vanaf de emulator voor te bereiden. De emulator mag niet “bezig zijn” met het uitvoeren van een App of programma voor de volgende stap.

– Ga naar de Emulator Explorer-werkruimte en verzend de programma('s) naar de computer.
– Gebruik TI Connect™ CE om de Python AppVars van de computer naar de CE-rekenmachine te verzenden voor de TI-Innovator™ Hub/TI-Innovator™ Rover-omgeving.

Opmerking: om een lopend Python-programma te onderbreken in de Shell, bijvoorbeeld als een programma in een continue lus is, drukt u op **[on]**. U kunt ook op **[Tools] [zoom] > 6:New Shell** drukken om een lopend programma te stoppen.

Onthoud: voor elke computer/TI-Python-activiteit geldt: Na het creëren van een Python-programma in een Python-ontwikkelomgeving op de computer, moet u valideren of uw programma op de rekenmachine werkt in de rekenmachine/emulator in de TI-Python-omgeving. Wijzig het programma zo nodig.

Toetsenblok op afstand SmartPad CE App

– Wanneer u de SmartPad CE App gebruikt op uw aangesloten CE-T, gedraagt deze zich als een toetsenblok op afstand, inclusief de speciale *toetsenblok*-koppeling die aangeboden wordt wanneer de Python App wordt uitgevoerd.

De Emulator Explorer-werkruimte

– Sluit de Python App af zodat de emulator niet bezig is wanneer u de volledige functies van de Emulator Explorer-werkruimte opent.
– program.py < > PY AppVar conversies zijn toegestaan. Dit is vergelijkbaar met wat de
TI Connect™ CE-omgeving doet tijdens het verzenden van programma's naar de aangesloten CE-rekenmachine.
– Wanneer u een program.py bestand dat gecreëerd is in een andere Python-omgeving verzendt, dan moet uw PY AppVar worden bewerkt om te werken zoals verwacht in TI-Python. Gebruik de Python App Editor om waar nodig de unieke modules te wijzigen, zoals
ti_plotlib, ti_system, ti_hub and ti_rover.

Data Import Wizard

– *.csv gegevensbestanden, opgemaakt zoals vermeld in het wizard-dialoogvenster, converteren gegevens naar CE-lijstvariabelen. Vervolgens kunnen methodes in ti_system worden gebruikt om lijsten te delen tussen het emulator-CE-OS en de Python App. Deze functie is hetzelfde als de Data Import Wizard in TI Connect™ CE.
- Als decimale getallen worden weergegeven met een komma in het *.csv bestand, dan wordt het bestand niet geconverteerd met de Data Import Wizard. Controleer de getalnotatie (format) van het besturingssysteem van uw computer en converteer het *.csv bestand zodat het de weergave met decimale punt gebruikt. De lijst- en matrixeditor van de CE-rekenmachine gebruikt bijvoorbeeld de getalnotatie 12.34 en niet 12,34.

TI Connect™ CE gebruiken voor het converteren van Python-programma's

Update naar TI Connect™ CE voor de nieuwste functies, waaronder het converteren van *.py-programma's naar een PY AppVar als de CE-rekenmachinebestandsopmaak.

Zie TI-84 Plus CE-T e-Guide voor meer informatie over de CE-rekenmachine, TI-SmartView™ CE-T en TI Connect CE.

Wat is de Python-programmeringsomgeving?

TI-Python is gebaseerd op CircuitPython, een variant van Python die ontworpen is om te passen in kleine microcontrollers. De originele CircuitPython-implementatie is aangepast voor gebruik door TI.

De interne opslag van getallen voor berekening in deze variant van Circuit Python gebeurt in een binaire opmaak met kommagetallen met beperkte precisie, en kan dus niet exact alle mogelijke decimale waarden weergeven. De verschillen ten opzichte van de werkelijke decimale representaties die het gevolg zijn van het opslaan van deze waarden, kunnen leiden tot onverwachte resultaten in opvolgende berekeningen.

- **Bij getallen met drijvende komma** - Toont een precisie van maximaal 16 significante cijfers. Intern worden waarden opgeslagen met een precisie van 53 bits, wat ongeveer gelijk is aan 15-16 cijfers achter de komma.
- **Bij gehele getallen** - De grootte van gehele getallen wordt alleen beperkt door het beschikbare geheugen op het moment dat de berekeningen worden uitgevoerd.

Modules die opgenomen zijn in de TI-84 Plus CE-T Python Edition

- Ingebouwde functies
- math module
- random module
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Opmerking: Als u bestaande Python-programma's hebt, die gecreëerd zijn in andere Python-ontwikkelomgevingen, bewerk uw programma('s) dan volgens de TI-Python manier. Modules kunnen andere methodes, argumenten en volgorde van methodes in een programma gebruiken in vergelijking met die van de ti_system, ti_plotlib, ti_hub, and ti_rover modules. Wees u in het algemeen bewust van compatibiliteit wanneer u een versie van Python en Python-modules gebruikt.

Bij het overzenden van Python-programma's van een niet-TI-platform naar een TI-platform OF van het ene TI-product naar een ander:

- Python-programma's die functies uit de hoofdtalen en standaard bibliotheken gebruiken (math, random enz.) kunnen zonder veranderingen worden overgezonden

  Opmerking: Lijsten mogen maximaal 100 elementen bevatten.

- Programma's die platform-specifieke bibliotheken voor TI-platforms gebruiken - matplotlib (voor pc), ti_plotlib,
  ti_system, ti_hub, etc. moeten bewerkt worden voordat ze kunnen worden uitgevoerd op een ander platform.

- Dit kan ook het geval zijn tussen TI-platformen.

Net als bij elke versie van Python moet u importopdrachten opnemen zoals uit math import *, om functies, methodes of constanten uit de math-module te kunnen gebruiken. Bijvoorbeeld: om de functie cos() uit te voeren gebruikt u de importopdracht om de math-module te importeren voor gebruik.

Zie de CATALOGUS-lijst.

Voorbeeld:

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

Alternatief voorbeeld:

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

De beschikbare modules kunnen worden weergegeven in de Shell met de volgende opdracht:

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

De inhoud van modules kan worden bekeken in de Shell zoals aangegeven, met “import module” en “dir(module).”

Niet alle module-inhoud verschijnt in de snelplakmenu's zoals [Fns…] of [2nd] [catalog].

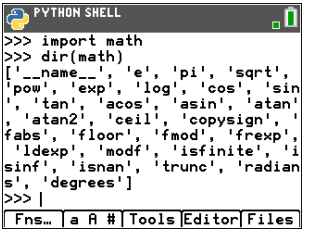
Inhoud van geselecteerde modules en trefwoorden

Voor een lijst van de modules die opgenomen zijn in deze versie, zie:

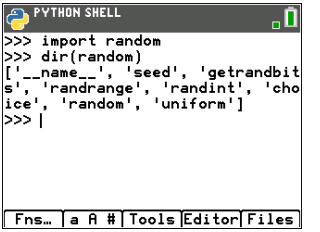
Bijlage: Geselecteerde ingebouwde functies, trefwoorden en module-inhoud van TI-Python

Onthoud: voor elke computer/TI-Python-ervaring geldt: Na het creëren van een Python-programma op de computer, moet u valideren of uw programma op de rekenmachine werkt in de TI-Python-omgeving. Wijzig het programma zo nodig.

Deze schermen geven de module-inhoud voor math en random weer.

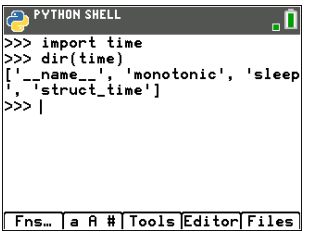


math module

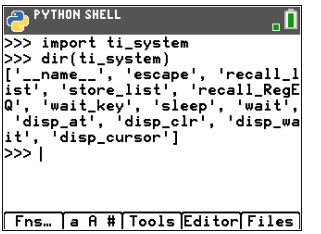


random module

Deze schermen geven de module-inhoud voor time en ti_system weer.

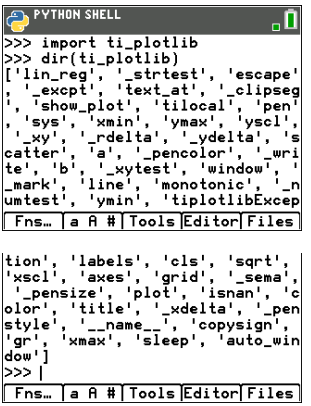


time



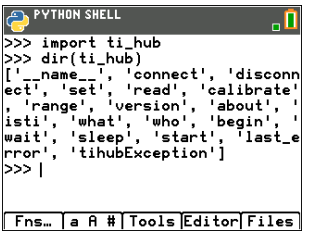
ti_system

Deze schermen geven de module-inhoud voor ti_plotlib weer.



ti_plotlib

Dit scherm geeft de module-inhoud voor ti_hub weer.



ti_hub

Deze schermen geven de module-inhoud voor ti_rover weer.

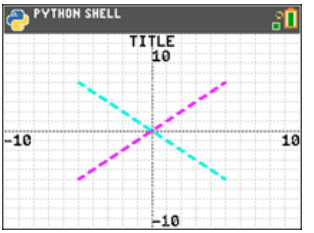
ti_rover

Voorbeeldprogramma's

Gebruik de volgende voorbeeldprogramma's om vertrouwd te raken met methodes uit de Handleiding. Deze voorbeelden bevatten verschillende TI-Innovator™ Hub- en TI-Innovator Rover™-programma's om u te helpen aan de slag te gaan met TI-Python.

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()
```



Druk op clear om de Shell-prompt weer te geven

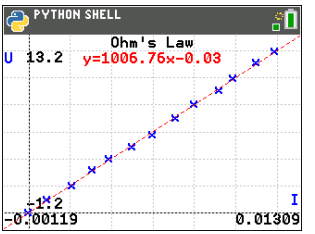
REGEQ1

Maak een regressievergelijking voordat u het Python-programma uitvoert in de Python App. Als voorbeeld kunt u eerst twee lijsten invoeren in het CE OS. Bereken daarna [stat] CALC 4:LinReg(ax+b) voor uw lijsten. Hierdoor wordt de regressievergelijking opgeslagen naar RegEQ in het OS. Hier is een programma om RegEQ op te roepen in de Python-omgeving.

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

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

LINREGR (aanwezig in CE Bundle)

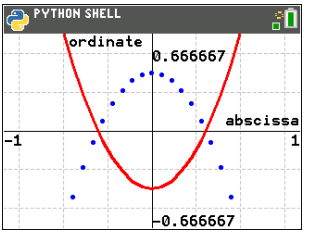
```
import ti_plotlib as plt

# 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))
```



Druk op clear om de Shell-prompt weer te geven

GRAPH (aanwezig in CE Bundle)

```
import ti_plotlib as plt
#After running the program, press [clear] to clear plot and return to
Shell.

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)
# Create a graph with parameters(resolution,xmin,xmax)
# After clearing the first graph, press the [var] key. De plot()
functie stelt u in staat de scherminstellingen (resolution,xmin,xmax)
te veranderen.
```

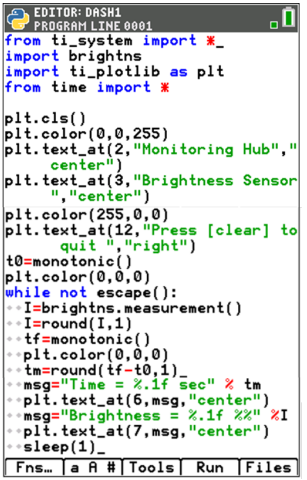


Druk op clear om de Shell-prompt weer te geven

DASH1 – Voorbeeld van een TI-Innovator™ Hub-programma

Zie: [Fns…]>Modul: ti_hub module

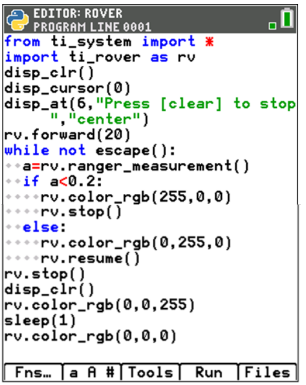
```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *
plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
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 – Voorbeeld van een TI-Innovator™ Rover-programma

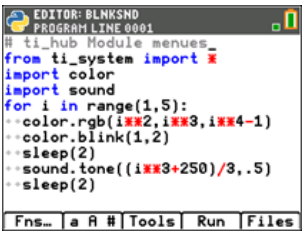
Zie: [Fns…]>Modul ti_rover module

```
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)
```



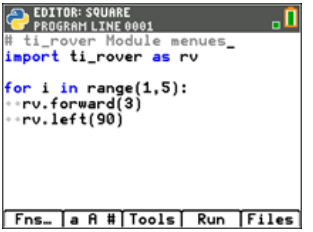
BLNKSND - Voorbeeld van een TI-Innovator™ Hub-programma

Zie: [Fns…]>Modul: ti_hub module



SQUARE- Voorbeeld van een TI-Innovator™ Rover-programma

Zie: [Fns…]>Modul ti_rover module



Handleiding voor de TI-Python-omgeving

De Python App bevat menu's met functies, klassen, bedieningselementen, operatoren en trefwoorden om snel te plakken in de Editor of Shell. De volgende tabel bevat de lijst met functies in 2nd [catalog] wanneer de App wordt uitgevoerd. Zie voor een volledige lijst met Python-functies, klassen, operatoren en trefwoorden in deze versie "Geselecteerde ingebouwde functies, trefwoorden en module-inhoud van TI-Python."

Deze tabel is niet bedoeld als een uitputtende lijst van Python die beschikbaar is in dit aanbod. Overige functies die in dit Python-aanbod worden ondersteund, kunnen worden ingevoerd met behulp van de alfabettoetsen op het toetsenblok.

De meeste voorbeelden die gegeven worden in deze tabel werken bij de Shell-prompt (>>>).

CATALOGUS-lijst

Alfabetische lijst

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

#

Scheidingsteken

2nd [catalog]

Syntax: #Uw opmerking over het programma.

[a A #]

Beschrijving: In Python begint een opmerking met het hashtag-teken, #, en loopt door tot het einde van de regel.

Voorbeeld:

```
#A short explanation of the code.
```

%

Operator

2nd [catalog]

Syntax: x%y or x % y

Beschrijving: Geeft de rest van x/y. Bij voorkeur gebruiken wanneer x en y gehele getallen zijn.

[a A #]

Voorbeeld:

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

Zie ook fmod(x,y).

//

Operator

2nd [catalog]

Syntax: x//y or x // y

[a A #]

Beschrijving: Geeft de floor-deling van x/y.

Voorbeeld:

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

[a A #]

Beschrijving: Starten van het [a A #] palet met speciale tekens.

Bevat tekens met accenten zoals ç à â è é ê ë î ï ô ö ù û

De sneltoets [a A #] staat op het scherm bij window in de Editor of Shell

a gradient; slope

Module: ti_plotlib

Syntax: plt.a gradient; slope

Beschrijving: Nadat plt.linreg() voor het laatst is uitgevoerd in een programma, worden de berekende waarden voor helling, a, en snijpunt, b, opgeslagen in plt.a en plt.b.

Standaardwaarden: = 0.0

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het Setup-menu ti_plotlib.

abs()

Module: Built-in

Syntax: abs(x)

Beschrijving: Geeft de absolute waarde van een getal. In deze uitgave kan het argument een geheel getal of een getal met zwevende komma zijn.

Voorbeeld:

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

2nd [catalog]

Opmerking: fabs() is een functie in de math-module.

acos()

Module: math

Syntax: acos(x)

Beschrijving: Geeft de arccosinus van x in radialen.

Voorbeeld:

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

Alternatief voorbeeld: [Tools] > 6:New Shell

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

[sin] 7:acos()

[2nd] [catalog]

[Fns…] Modul
1:math… > Trig
7:acos()

Importopdrachten zijn te vinden in [2nd] [catalog]

and

Trefwoord

Syntax: x and y

Beschrijving: Kan True of False geven. Geeft "x" als "x" False is en anders "y". Wordt geplakt met een spatie voor en na and. Bewerk indien nodig.

Voorbeeld:

```
>>>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)

Module: Built-in

Syntax: listname.append(item)

Beschrijving: De methode append() voegt een item toe aan een lijst.

Voorbeeld:

```
>>>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

Trefwoord

Beschrijving: Gebruik as om een alias te creëren tijdens het importeren van een module. Zie voor meer informatie de Python-documentatie.

2nd [catalog]

asin()

Module: math

Syntax: asin()

Beschrijving: Geeft de arcsinus van x in radialen.

Voorbeeld:

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

Alternatief voorbeeld:

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

sin 6:asin()

2nd [catalog]

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

importopdrachten zijn te vinden in
2nd [catalog]

assert

Trefwoord

2nd [catalog]

Beschrijving: Gebruik assert om een conditie in uw programma te testen. Geeft None; zo niet, dan geeft de uitvoering van het programma een AssertionError.

atan()

Module: math

sin 8:atan()

Syntax: atan(x)

Beschrijving: Geeft de arctangens van x in radialen.

[Fns…]>Modul 1:math… > Trig 8 :atan()

Voorbeeld:

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

2nd [catalog]

Alternatief voorbeeld:

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

importopdrachten zijn te vinden in 2nd [catalog]

atan2(y,x)

Module: math

sin 9:atan2()

Syntax: atan2(y,x)

Beschrijving: Geeft de arctangens van y/x in radialen. Het resultaat is in [-pi, pi].

[Fns…] > Modul 1:math… > Trig 9:atan2()

Voorbeeld:

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

2nd [catalog]

Alternatief voorbeeld:

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

importopdrachten zijn te vinden in 2nd [catalog]

auto_window(xlist,ylist)

Module: ti_plotlib

Syntax: plt.auto_window(xlist,ylist)

Beschrijving: Zorgt voor automatische schaling van het plotvenster zodat de gegevensbereiken binnen xlist en ylist die in het programma gespecificeerd zijn vóór de auto_window() passen.

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

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

axes("mode")

Module: ti_plotlib

Syntax: plt.axes("mode")

Beschrijving: Geeft assen weer op het gespecificeerde venster in het plotgebied.

Argument:

Argumentopties voor "mode":

"off"	geen assen
"on"	assen+labels
"axes"	alleen assen
"window"	alleen vensterlabels

plt.axes() gebruikt de huidige penkleurinstelling. Om te zorgen dat plt.axes() altijd getekend worden zoals verwacht, gebruikt u plt.color() VOOR plt.axes() om te zorgen dat de kleuren volgens verwachting zijn.

Voorbeeld:

Zie voorbeeldprogramma LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

b y= intercept

Module: ti_plotlib

2nd [catalog]

Syntax: plt.b y= intercept

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

Beschrijving: Nadat plt.linreg() is uitgevoerd in een programma, worden de berekende waarden voor helling, a, en snijpunt met de y-as, b, opgeslagen in plt.a en plt.b.

Standaardwaarden: = 0.0

Voorbeeld:

Zie voorbeeldprogramma LINREGR.

Importopdrachten zijn te vinden in 2nd [catalog] of in het Setup-menu ti_plotlib.

bin(integer)

Module: Built-in

2nd [catalog]

Syntax: bin(integer)

Beschrijving: Geeft de binaire opmaak van het geheel getal-argument weer.

Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

break

Trefwoord

2nd [catalog]

Beschrijving: Gebruik break om uit een for- of while-lus te gaan.

ceil()

Module: math

Syntax: ceil(x)

Beschrijving: Geeft het kleinste gehele getal dat groter dan of gelijk is aan x.

Voorbeeld:

```
>>>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()

importopdrachten zijn te vinden in
[2nd] [catalog]

choice(sequence)

Module: random

Syntax: choice(sequence)

Beschrijving: Geeft een willekeurig element uit een niet-lege rij.

Voorbeeld:

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA) #Uw resultaat kan anders zijn.
4
```

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

[2nd] [catalog]

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

importopdrachten zijn te vinden in
[2nd] [catalog]

chr(integer)

Module: Built-in

2nd [catalog]

Syntax: chr(integer)

Beschrijving: Geeft een string (tekenreeks) uit een invoer van gehele getallen die het unicode-teken vertegenwoordigt.

Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

class

Trefwoord

2nd [catalog]

Beschrijving: Gebruik class om een klasse te creëren. Zie voor meer informatie de Python-documentatie.

cls() clear screen

Module: ti_plotlib

2nd [catalog]

Syntax: plt.cls() clear screen

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

Beschrijving: Wist het Shell-scherm voor het plotten. Sneltoetsen worden niet weergegeven tijdens het plotten.

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

Opmerking: plt.cls() heeft een ander gedrag dan ti_system module disp_clr().

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

Importopdrachten zijn te vinden in 2nd [catalog] of in het Setup-menu ti_plotlib.

color(r,g,b) 0-255

Module: ti_plotlib

2nd [catalog]

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

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

Beschrijving: Stelt de kleur in voor alle volgende grafische afbeeldingen/plots. (R,g,b)-waardenr,g,b)-waarden moet worden gespecificeerd van 0-255. De gespecificeerde kleur wordt gebruikt in de plotweergave tot color() opnieuw wordt uitgevoerd met een andere kleur.

Importopdrachten zijn te vinden in 2nd [catalog] of in het Setup-menu ti_plotlib.

De standaardkleur is zwart bij het importeren van ti_plotlib.

Voorbeeld:

Zie voorbeeldprogramma: COLORLIN.

complex(real,imag)

Module: Built-in

2nd [catalog]

Syntax: complex(real,imag)

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

Beschrijving: Type: complex getal.

Voorbeeld:

```
>>>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)
```

Opmerking: "1+2j" is de correcte syntax. Spaties als "1 + 2j" geven een Exception weer.

continue

Trefwoord

2nd [catalog]

Beschrijving: Gebruik 'continue' in een for- of while-lus om de huidige iteratie te beëindigen. Zie voor meer informatie de Python-documentatie.

cos()

Module: math

sin Trig
4: cos()

Syntax: cos(x)

Beschrijving: Geeft de cos van x. Het hoekargument is in radialen.

2nd [catalog]

Voorbeeld:

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

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

Alternatief voorbeeld:

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

Opmerking: Python geeft de wetenschappelijke notatie weer met e of E. Sommige wiskunderesultaten in Python zullen anders zijn dan in het CE OS.

.count()

Module: Built-in

2nd [catalog]

Syntax: listname.count(item)

Beschrijving: count() is een methode die het aantal keren geeft dat een item voorkomt in een van de volgende objecten: lijst, tupel, bytes, str, bytearray- of array.array.

Voorbeeld:

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

def function ():

Trefwoord

2nd [catalog]

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

Beschrijving: Een functie definiëren afhankelijk van gespecificeerde variabelen. Vaak gebruikt met de trefwoord-return.

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

[Fns...]>Func
2:return

Voorbeeld:

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

degrees()

Module: math

sin Trig
2:degrees()

Syntax: degrees(x)

Beschrijving: Converteert hoek x van radialen naar graden.

2nd [catalog]

Voorbeeld:

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

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

del

Trefwoord

2nd [catalog]

Beschrijving: Gebruik 'del' om objecten zoals variabelen, lijsten enz. te wissen
Zie voor meer informatie de Python-documentatie.

disp_at(row,col,"text")

Module: ti_system

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

Beschrijving: Tekst weergeven vanaf een rij- en kolompositie in het plotgebied.

REPL with cursor >>>| verschijnt na tekst aan het eind van een programma. Gebruik disp_cursor() om de cursorweergave te regelen.

Argument:

row	1 - 11, integer
column	1 - 32, integer
"text"	is een string (tekenreeks) die doorloopt op het schermgebied

Optionele argumenten voor kleur en achtergrond die hier worden weergegeven: disp_at (row,col,"text","align",color 0-15, background color 0-5)

Voorbeeld:

Voorbeeldprogramma:

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

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

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

importopdrachten zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

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

Module: ti_system

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

Beschrijving: Tekst uitgelijnd weergeven zoals gespecificeerd op het plotscherm voor rij 1-11. De rij wordt gewist voordat deze wordt weergegeven. Bij gebruik in een lus wordt de inhoud vernieuwd bij elke weergave.

REPL with cursor >>>| verschijnt na tekst aan het eind van een programma. Gebruik disp_cursor() om de cursorweergave te regelen voordat u disp_at() gebruikt in uw programma.

Argument:

row	1 - 11, integer
"text"	is een string die doorloopt op het schermgebied
"align"	"left" (default) "center" "right"

Optioneel argument dat hier wordt weergegeven: disp_at(row,"text","align","color 0-15, background color 0-15)

Voorbeeld:

Voorbeeldprogramma:

```
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 of
math
4:ti_system
7:disp_at()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

disp_clr() clear text screen

Module: ti_system

Syntax: disp_clr() clear text screen

Beschrijving: Het scherm wissen in de Shell-omgeving. Row 0-11, integer kan worden gebruikt als optioneel argument om een schermregel in de Shell-omgeving te wissen.

Voorbeeld:

Voorbeeldprogramma:

```
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 of
math
4:ti_system
8:disp_clr()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

disp_cursor() 0=off 1=on

Module: ti_system

Syntax: disp_cursor() 0=off 1=on

Beschrijving: De weergave van de cursor in de Shell regelen wanneer een programma wordt uitgevoerd.

Argument:

0 = off

not 0 = on

Voorbeeld:

Voorbeeldprogramma:

```
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 of
math
4:ti_system
0:disp_cursor()

importopdrachten zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

disp_wait() [clear]

Module: ti_system

Syntax: disp_wait() [clear]

Beschrijving: De uitvoering van het programma op dit punt stoppen en de scherminhoud weergeven totdat op [clear] wordt gedrukt en het scherm wordt gewist.

Voorbeeld:

Voorbeeldprogramma:

```
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 of
math
4:ti_system
9:disp_wait()

importopdrachten zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

E

e

Module: math

Syntax: math.e of e als de math-module geïmporteerd is

Beschrijving: Constante e wordt getoond zoals hieronder weergegeven.

Voorbeeld:

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

Alternatief voorbeeld:

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

2nd [e] (boven ÷)

[Fns…] > Modul 1:math…
> Const 1:e

elif :

Trefwoord

Zie if..elif..else.. voor meer informatie.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Trefwoord

Zie if..elif..else.. voor meer informatie.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Module: ti_system

Syntax: escape()

Description: escape() geeft True of False.

De beginwaarde is False.

Wanneer de [clear]-toets op CE wordt ingedrukt, wordt de waarde ingesteld op True.

Wanneer de functie is uitgevoerd, wordt de waarde teruggezet op False.

Voorbeeld van gebruik:

while not escape():

In een while-lus die uitgevoerd wordt in een programma, wanneer het programma aanbiedt om de lus te beëindigen maar het script blijft uitvoeren.

if escape():break

Kan worden gebruikt in een debug-programma om de variabelen te inspecteren gebruikmakend van Shell [vars] na het uitvoeren van het programma en met behulp van deze onderbreking.

2nd [catalog]

Als een programmaregel:

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

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

importopdrachten zijn te vinden in 2nde [catalog] of in het menu van de module ti_system.

eval()

Module: Built-in

2nd [catalog]

Syntax: eval(x)

[Fns…] I/O
3:eval()

Beschrijving: Geeft de uitwerking van de uitdrukking x.

Voorbeeld:

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

except exception:

Trefwoord

2nd [catalog]

Beschrijving: Gebruik except in een try..except codeblok. Zie voor meer informatie de Python-documentatie.

exp()

Module: math

2nd [e^x] (boven ln)

Syntax: exp(x)

Beschrijving: Geeft e**x.

2nd [catalog]

Voorbeeld:

[Fns…] > Modul
1:math…
4:exp()

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

Alternatief voorbeeld: [Tools] > 6:New Shell

importopdrachten zijn te vinden in 2nd [catalog]

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

.extend()

Module: Built-in

2nd [catalog]

Syntax: listname.extend(newlist)

Beschrijving: De methode extend() is een methode om newlist uit te breiden aan het einde van een lijst.

Voorbeeld:

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

fabs()

Module: math

2nd [catalog]

Syntax: fabs(x)

Beschrijving: Geeft de absolute waarde van x

[Fns…] > Modul
1:math...
2:fabs()

Voorbeeld:

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

importopdrachten zijn te vinden in 2nd [catalog]

Zie ook de ingebouwde functie abs().

False

Trefwoord

2nd [test] (boven math)

Beschrijving: Geeft False wanneer de uitgevoerde bewering onwaar is. "False" staat voor de onware waarde van objecten van het type Boole.

2nd [catalog]

Voorbeeld:

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

[Fns…] > Ops
B:False

[a A #]

finally:

Trefwoord

2nd [catalog]

Beschrijving: Gebruik finally in een try..except..finally programmablok. Zie voor meer informatie de Python-documentatie.

float()

Module: Built-in

2nd [catalog]

Syntax: float(x)

Beschrijving: Geeft x als een getal met drijvende komma.

[Fns…] > Type
2:float()

Voorbeeld:

```
>>>float(35)
35.0
>>>float("1234")
1234.0
```

floor()

Module: math

math Modul
1:math
9:floor()

Syntax: floor(x)

Beschrijving: Geeft het grootste gehele getal dat kleiner dan of gelijk is aan x.

2nd [catalog]

Voorbeeld:

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

[Fns…] > Modul
1:math
9:floor()

importopdrachten zijn te vinden in
2nd [catalog]

fmod(x,y)

Module: math

Syntax: fmod(x,y)

Beschrijving: Zie voor meer informatie de Python-documentatie. Bij voorkeur gebruiken wanneer x en y getallen met drijvende komma zijn.

Geeft mogelijk niet hetzelfde resultaat als x%y.

Voorbeeld:

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0 #validatie van beschrijving
-2.0
```

Zie ook: x%y.

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

[2nd] [catalog]

[Fns…] > Modul 1:math… 7:fmod()

importopdrachten zijn te vinden in [2nd] [catalog]

for i in list:

Trefwoord

Syntax: for i in list:

Beschrijving: Gebruikt om te itereren over lijstelementen.

Voorbeeld:

```
>>> 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):

Trefwoord

Syntax: for i in range(size)

Beschrijving: Gebruikt om te itereren over een reeks waarden.

Voorbeeld:

[Fns…] Ctl 4:for i in range (size):

2nd [catalog]

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

for i in range(start,stop):

Trefwoord

Syntax: for i in range(start,stop)

Beschrijving: Gebruikt om te itereren over een bereik.

Voorbeeld:

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

2nd [catalog]

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

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

Trefwoord

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

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

Beschrijving: Gebruikt om te itereren over een bereik.

Voorbeeld:

2nd [catalog]

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

str.format() string format

Module: Built-in

2nd [catalog]

Syntax:str.format()

Beschrijving: Formatteert de gegeven string (tekenreeks). Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

frexp()

Module: math

Syntax: frexp(x)

Beschrijving: Geeft een paar (y,n) waarbij x == y * 2**n. y een getal met drijvende komma is waarbij 0.5<abs(y)<1; en n een geheel getal is.

Voorbeeld:

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11 #beschrijving valideren
2000.0
```

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

[2nd] [catalog]

[Fns…] > Modul 1:math A:frexp()

importopdrachten zijn te vinden in [2nd] [catalog]

from PROGRAM import *

Trefwoord

Syntax: from PROGRAM import *

Beschrijving: Gebruikt om een programma te importeren. Importeert de openbare gegevenskenmerken van een Python-module in de huidige naamruimte.

Shell [Tools] A:from PROGRAM import *

[2nd] [catalog]

from math import *

Trefwoord

Syntax: from math import *

Beschrijving: Gebruikt om alle functies en constanten uit de math-module te importeren.

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

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

[2nd] [catalog]

from random import *

Trefwoord

Syntax: from random import *

Beschrijving: Gebruikt om alle functies uit de random-module te importeren
.

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

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

[2nd] [catalog]

from time import *

Trefwoord

Syntax: from time import *

Beschrijving: Gebruikt om alle methodes uit de time-module te importeren.

Voorbeeld:

Zie voorbeeldprogramma: DASH1.

2nd [catalog]

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

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

from ti_system import *

Trefwoord

Syntax: from ti_system import *

Beschrijving: Gebruikt om alle methodes uit de ti_system-module te importeren.

Voorbeeld:

Zie voorbeeldprogramma: 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 *

Trefwoord 2nd [catalog]

Syntax: from ti_hub import *

Beschrijving: Gebruikt om alle methodes uit de ti_hub-module te importeren. Gebruik bij afzonderlijke invoer- en uitvoerapparaten de dynamische modulefunctionaliteit door het apparaat te selecteren in het [Fns…]>Modul>ti_hub>Import menu wanneer u in de Editor bent.

Zie:ti_hub module – Import toevoegen aan de Editor en de ti_hub sensormodule toevoegen aan het menu Modul.

Voorbeeld:

Zie voorbeeldprogramma: DASH1.

global

Trefwoord

2nd [catalog]

Beschrijving: Gebruik global om globale variabelen binnen een functie te creëren.

Zie voor meer informatie de CircuitPython-documentatie.

grid(xscl,yscl,"style")

Module: ti_plotlib

2nd [catalog]

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

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

Beschrijving: Geeft een rooster weer met een gespecificeerde schaal voor de x- en y-as. Opmerking: Al het plotten vindt plaats wanneer plt.show_plot() wordt uitgevoerd.

Het instellen van een roosterkleur is het optionele argument van (r,g,b) met waarden 0-255 met de standaardwaarde grijs (192,192,192).

De standaardwaarde voor xscl of yscl = 1.0.

"style" = "dot" (default), "dash", "solid" of "point"

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

Voorbeeld:

Zie voorbeeldprogramma's: COLORLIN of GRAPH.

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

Module: ti_plotlib

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

Beschrijving: Geeft een rooster weer met een gespecificeerde schaal voor de x- en y-as. Opmerking: Al het plotten vindt plaats wanneer plt.show_plot() wordt uitgevoerd.

Het instellen van een roosterkleur is het optionele argument van (r,g,b) met waarden 0-255 met de standaardwaarde grijs (192,192,192).

De standaardwaarde voor xscl of yscl = 1.0.

"style" = "dot" (default), "dash", "solid" of "point".

Als de waarden van xscl of yscl kleiner zijn dan 1/50e van het verschil tussen xmax-xmin of ymax-ymin, dan verschijnt de uitzondering 'Invalid grid scale value.'

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

hex(integer)

Module: Built-in

2nd [catalog]

Syntax: hex(integer)

Beschrijving: Geeft de hexadecimale opmaak van het geheel getal-argument weer. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

"if :"	
Zie if..elif..else.. voor meer informatie.	2nd [catalog] [Fns…] > Ctl 1:if.. 2:if..else.. 3:if..elif..else 9:elif : 0:else:

if..elif..else..

Trefwoord

2nd [catalog]

Syntax: ••Grijze inspringingsstippen die automatisch worden aangeboden in de Python App voor gebruiksgemak.

[Fns…] > Ctl

if :

1:if..

••

2:if..else..

elif :

3:if..elif..else

••

9:elif :

else:

0:else:

Beschrijving: if..elif..else is een voorwaardelijke bewering. De Editor biedt automatische inspringingen in de vorm van grijze stippen als hulp bij het maken van correcte inspringingen bij het programmeren.

Voorbeeld: Creëer en voer dit programma uit, laten we zeggen S01, vanuit de Editor

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

Shell-interactie

```
>>> # Shell Reinitialized
>>> # Running S01
>>>from S01 import * #wordt automatisch geplakt
>>>f(5)
5
>>>f(0)
nul
>>>f(-5)
5
```

if..else..

Trefwoord 2nd [catalog]

Zie if..elif..else.. voor meer informatie.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Module: Built-in 2nd [catalog]

Syntax:var.imag

Beschrijving: Geeft het imaginaire deel van een gespecificeerde variabele van het type: complex getal.

Voorbeeld:

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

import math

Trefwoord

Syntax: import math 2nd [catalog]

Beschrijving: Met deze opdracht wordt de math-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de "math" module binnen zijn eigen naamruimte.

import random

Trefwoord

Syntax: import random 2nd [catalog]

Beschrijving: Met deze opdracht wordt de random-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de "random" module binnen zijn eigen naamruimte.

import ti_hub

Trefwoord 2nd [catalog]

Syntax: import ti_hub

Beschrijving: Met deze opdracht wordt de ti_hub-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de ti_hub module binnen zijn eigen naamruimte.

Gebruik bij afzonderlijke invoer- en uitvoerapparaten de dynamische modulefunctionaliteit door het apparaat te selecteren in het [Fns…]>Modul>ti_hub>Import menu wanneer u in de Editor bent.

Zie:[Fns…] > Modul: ti_hub module.

import time

Trefwoord 2nd [catalog]

Syntax: import time

Beschrijving: Met deze opdracht wordt de time-module geopend. Deze opdracht importeert de openbare attributen van de time-module binnen zijn eigen naamruimte.

Zie:[Fns…] > Modul: time en ti_system modules.

import ti_plotlib as plt

Trefwoord

Syntax: import ti_plotlib as plt

Beschrijving: Met deze opdracht wordt de ti_plotlib-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de ti_plotlib module binnen zijn eigen naamruimte. Gegevenskenmerken van de ti_plotlib module moeten worden ingevoerd als plt.attribute.

Voorbeeld:

Zie voorbeeldprogramma: 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

Trefwoord

Syntax: import ti_rover as rv

Beschrijving: Met deze opdracht wordt de ti_rover-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de ti_rover module binnen zijn eigen naamruimte. Gegevenskenmerken van de ti_rover module moeten worden ingevoerd als rv.attribute.

Voorbeeld:

Zie voorbeeldprogramma: 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

Trefwoord 2nd [catalog]

Syntax: import ti_system

Beschrijving: Met deze opdracht wordt de ti_system-module geopend. Deze opdracht importeert de openbare gegevenskenmerken van de ti_system module binnen zijn eigen naamruimte.

Voorbeeld:

Zie voorbeeldprogramma: REGEQ1.

in

Trefwoord 2nd [catalog]

Beschrijving: Gebruik 'in' om te controleren of een waarde in een rij staat of om een rij te itereren in een for-lus.

.index(x)

Module: Built-in 2nd [catalog]

Syntax:var.index(x)

Beschrijving: Geeft de index of positie van een element van een lijst. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

input()

Module: Built-in 2nd [catalog]

input()

Syntax: input()

Beschrijving: Prompt voor invoer

[Fns…] I/O
2:input()

Voorbeeld:

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

Alternatief voorbeeld:

```
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)

Module: Built-in

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

Syntax: listname.insert(index,x)

Beschrijving: method insert() voegt item x in na een index binnen een rij.

2nd [catalog]

Voorbeeld:

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

[Fns…] > List
8:.insert(index,x)

int()

Module: Built-in

2nd [catalog]

Syntax: int(x)

Beschrijving: Geeft x als een geheel getalobject.

[Fns…] > Type
1:int()

Voorbeeld:

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

is

Trefwoord

2nd [catalog]

Beschrijving: Gebruik 'is' om te testen of twee objecten hetzelfde object zijn.

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

Module: ti_plotlib

2nd [catalog]

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

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

Beschrijving: Geeft de labels "xlabel" en "ylabel" weer op de plotassen op rijposities x en y. Pas dit zo nodig aan voor uw plotweergave.

"xlabel" wordt geplaatst op gespecificeerde rij x (standaard rij 12) en wordt rechts uitgelijnd.

"ylabel" wordt geplaatst op gespecificeerde rij y (standaard rij 2) en wordt links uitgelijnd.

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

Opmerking: plt.labels("|","",12,2) wordt geplakt met de standaardrijwaarde voor x en y, 12,2, die vervolgens kunnen worden aangepast voor uw programma.

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

lambda

Trefwoord

2nd [catalog]

Syntax: lambda arguments : expression

Beschrijving: Gebruik lambda om een anonieme functie te definiëren. Zie voor informatie de Python-documentatie.

len()

Module: Built-in

Syntax: len(sequence)

Beschrijving: Geeft het aantal items in het argument. Het argument kan een rij of een verzameling zijn.

Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

2nd [list] (boven stat) List
3:len()

2nd [catalog]

[Fns…] > List
3:len()

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

Module: ti_plotlib

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

Beschrijving: Toont een lijnstuk van (x1,y1) naar (x2,y2)

De grootte en stijl worden ingesteld met pen() en color() voor line().

Argumenten:

x1,y1, x2,y2 zijn reële getallen met drijvende komma.

"mode": Bij de standaardinstelling "" wordt er geen pijlkop getekend.
Bij "arrow" wordt er een vectorpijlkop getekend bij (x2,y2).

Voorbeeld:

Zie voorbeeldprogramma: COLORLIN.

2nd [catalog]

[Fns...]>Modul of math
5:ti_plotlib...> Draw
7:line or vector

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

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

Module: ti_plotlib

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

Beschrijving: Berekent en tekent het lineaire regressiemodel, ax+b, van xlist,ylist. Deze methode moet volgend op de scatter-methode (spreidingsdiagram). De standaardweergave van de vergelijking is "center" op rij 11.

Argument:

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

plt.a (helling) en plt.b (snijpunt met y-as) worden opgeslagen wanneer lin_reg wordt uitgevoerd.

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

list(sequence)

Module: Built-in

Syntax: list(sequence)

Beschrijving: Veranderlijke rij van items van het opgeslagen type.

list()" converteert het argument ervan naar het "list" type. Net als bij veel andere rijen, hoeven de elementen van een lijst niet van hetzelfde type te zijn.

Voorbeeld:

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

Voorbeeld:

```
>>>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] (boven stat) List
2:list(sequence)

2nd [catalog]

[Fns…] > List
2:list(sequence)

log(x,base)

Module: math

Syntax: log(x,base)

Beschrijving: log(x) zonder grondtal geeft de natuurlijke logaritme x.

Voorbeeld:

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

2nd log voor log(x,10)

2nd ln voor log(x) (natuurlijke logaritme)

math Modul
1:math…
6:log(x,base)

2nd [catalog]

[Fns…] > Modul
1:math…
6:log(x,base)

importopdrachten zijn te vinden in
2nd [catalog]

math.function

Module: math

2nd [catalog]

Syntax: math.function

Beschrijving: Gebruiken na het importeren van een math-opdracht om een functie in de math-module te gebruiken.

Voorbeeld:

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

max()

Module: Built-in

2nd [list] (boven stat) List 4:max()

Syntax: max(sequence)

Beschrijving: Geeft de maximumwaarde in de rij. Zie de Python-documentatie voor meer informatie over max().

2nd [catalog]

Voorbeeld:

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

[Fns…] > List 4:max()

min()

Module: Built-in

2nd [list] (boven stat) List 5:min()

Syntax: min(sequence)

Beschrijving: Geeft de minimumwaarde in de rij. Zie de Python-documentatie voor meer informatie over min().

2nd [catalog]

Voorbeeld:

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

[Fns…] > List 5:min()

monotonic() elapsed time

Module: time

Syntax: monotonic() elapsed time

Beschrijving: Geeft een tijdswaarde vanaf het moment van uitvoering. Gebruik de gegeven waarde om deze te vergelijken met andere waarden uit monotonic().

Voorbeeld:

Voorbeeldprogramma:

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

```
Voer het programma EXAMPLE uit totdat de
uitvoering stopt.
>>>15.0
```

2nd [catalog]

[Fns...]>Modul of
math
3:time
3:monotonic()

importopdrachten zijn te vinden in 2nd [catalog] of in het menu van de time-module.

None

Trefwoord [2nd] [catalog]

Beschrijving: None vertegenwoordigt de afwezigheid van een waarde.

[a A #]

Voorbeeld:

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

nonlocal

Trefwoord [2nd] [catalog]

Syntax: nonlocal

Beschrijving: Gebruik nonlocal om te aan te geven dat een variabele niet lokaal is. Zie voor meer informatie de Python-documentatie.

niet

Trefwoord [2nd] [test] Ops 0:not

Syntax: not x

Beschrijving: Wordt uitgewerkt naar True als xonwaar is en anders naar False. Wordt geplakt met een spatie voor en na het trefwoord not. Bewerk indien nodig.

[Fns…] > Ops 0:not

Voorbeeld: [2nd] [catalog]

```
>>> not 2<5 #bewerk de spatie voor not
False
>>>3<8 and not 2<5
False
```

[a A #]

oct(integer)

Module: Built-in

2nd [catalog]

Syntax: oct(integer)

Beschrijving: Geeft de octale (achttallige) representatie van het gehele getal. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

or

Trefwoord

2nd [test] Ops 9:or

Syntax: x or y

[Fns…] > Ops 9:or

Beschrijving: Kan True of False geven. Geeft x als x wordt uitgewerkt naar True en anders y. Wordt geplakt met een spatie voor en na or. Bewerk indien nodig.

2nd [catalog]

Voorbeeld:

[a A #]

```
>>>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")

Module: Built-in 2nd [catalog]

Syntax: ord("character")

Beschrijving: Geeft de unicode-waarde van het teken.
Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

pass

Trefwoord

2nd [catalog]

Beschrijving: Gebruik pass in een lege functie of klassedefinitie als een tijdelijke aanduiding voor toekomstige code terwijl u uw programma verder afmaakt. Lege definities veroorzaken geen fout wanneer een programma wordt uitgevoerd.

pen("size","style")

Module: ti_plotlib

2nd [catalog]

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

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

Beschrijving: Stelt het uiterlijk van alle volgende regels in tot de volgende pen() wordt uitgevoerd.

Argument:

De standaardwaarde voor pen() is "thin" en "solid."

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

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

Voorbeeld:

Zie voorbeeldprogramma's: COLORLIN of GRAPH.

pi

Module: math

2nd [π] (boven sin)

Syntax: math.pi of pi als de math-module geïmporteerd is.

Beschrijving: De constante pi wordt getoond zoals hieronder weergegeven.

[Fns…] > Modul 1:math… > Const 2:pi

Voorbeeld:

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

Alternatief voorbeeld:

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

plot(xlist,ylist,"mark")

Module: ti_plotlib

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

Beschrijving: Er wordt een lijnplot met geordende paren weergegeven uit de gespecificeerde xlist en ylist. Het lijntype en de lijngrootte worden ingesteld met plt.pen().

xlist en ylist moeten reële drijvendekommagetallen zijn en lijsten moeten dezelfde afmeting hebben.

Argument:

"mark" is het markeringsteken, als volgt:

o	filled dot (default)
+	cross
x	x
.	pixel

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

plot(x,y,"mark")

Module: ti_plotlib

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

Beschrijving: Er wordt een puntplot (x,y) weergegeven met de gespecificeerde x en y.

xlist en ylist moeten reële drijvendekommagetallen zijn en lijsten moeten dezelfde afmeting hebben.

Argument:

"mark" is het markeringsteken, als volgt:

o	filled dot (default)
+	cross
x	x
.	pixel

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

pow(x,y)

Module: math

Syntax: pow(x,y)

Beschrijving: Geeft x tot de macht y. Converteert x en y in drijvendekommagetallen. Zie voor meer informatie de Python-documentatie.

Gebruik de ingebouwde pow(x,y) functie of ** voor het berekenen van machten van gehele getallen.

Voorbeeld:

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

Voorbeeld met: Ingebouwde functie:

[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)

importopdrachten zijn te vinden in [2nd] [catalog]

print()

Module: Built-in

Syntax: print(argument)

Beschrijving: Geeft een argument weer als string.

Voorbeeld:

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

[2nd] [catalog]

[Fns…] > I/O 1:print()

radians()) degree ▶radians

Module: math

Syntax: radians(x)

Beschrijving: Converteert hoek x van graden in radialen.

Voorbeeld:

```
>>>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

Trefwoord

Syntax: raise exception

Beschrijving: Gebruik raise om een gespecificeerde uitzondering te genereren en uw programma te stoppen.

[2nd] [catalog]

randint(min,max)

Module: random

Syntax: randint(min,max)

Beschrijving: Geeft een willekeurige geheel getal tussen min en max.

Voorbeeld:

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

Alternatief voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

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

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

[2nd] [catalog]

importopdrachten zijn te vinden in
[2nd] [catalog]

random()

Module: random

Syntax: random()

Beschrijving: Geeft een getal met drijvende komma tussen 0 en 1.0. Deze functie heeft geen argumenten.

Voorbeeld:

```
>>>from random import *
>>>random()
0,5381466990230621
```

Alternatief voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

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

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

[2nd] [catalog]

importopdrachten zijn te vinden in [2nd] [catalog]

random.function

Module: random

Syntax: random.function

Beschrijving: Gebruik deze na het importeren van random om een functie te openen in de random-module.

Voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

[2nd] [catalog]

randrange(start,stop,step)

Module: random

Syntax: randrange(start,stop,step)

Beschrijving: Geeft een willekeurig getal van start tot stop per stap.

Voorbeeld:

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

Alternatief voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

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

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

[2nd] [catalog]

importopdrachten zijn te vinden in [2nd] [catalog]

range(start,stop,step)

Module: Built-in

[2nd] [catalog]

Syntax: range(start,stop,step)

Beschrijving: Gebruik de range-functie om een getallenrij te geven. Alle argumenten zijn optioneel. De standaardwaarde voor start is 0, de standaardwaarde voor stap is 1 en de rij eindigt bij stop.

Voorbeeld:

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

.real

Module: Built-in 2nd [catalog]

Syntax:var.real

Beschrijving: Geeft het reële deel van een gespecificeerde variabele van het type complex getal.

Voorbeeld:

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

var=recall_list("name") 1-6

Module: ti_system

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

Beschrijving: Een voorgedefinieerde OS-lijst oproepen. De lengte van de lijst moet kleiner dan of gelijk zijn aan 100.

Argument: "**name**"

Voor OS L1-L6

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

Voor aangepaste OS-lijst "name"

----- Max. 5 tekens, getallen of letters, beginnend met letters, en letters moeten hoofdletters zijn.

Voorbeelden:

"ABCDE"

"R12"

"L1" is de aangepaste lijst L1 en niet OS L1

Onthoud: Python kent dubbele precisie. Python ondersteunt meer cijfers dan in het OS.

Voorbeeld:

Voorbeeldprogramma:

Creëer een lijst in het OS.
LIST={1,2,3}

Voer de Python App uit.
Maak een nieuw programma AA.

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

Voer programma AA uit.
De uitvoer wordt in de Shell weergegeven.

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

2nd [catalog]

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

[Fns...]>Modul of
math
4:ti_system
4:var=recall_list()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

var=recall_RegEQ()

Module: ti_system

Syntax:var=recall_RegEQ()

Beschrijving: De RegEQ-variabele oproepen vanuit het CE OS. De regressievergelijking moet berekend zijn in het OS voordat RegEQ wordt opgeroepen in de Python App.

Voorbeeld:

Zie voorbeeldprogramma: REGEQ1.

2nd [catalog]

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

[Fns…]>Modul of
math
4:ti_system
4:var=recall_REGEQ
()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

.remove(x)

Module: Built-in

Syntax: listname.remove(item)

Beschrijving: De method remove() verwijdert een item de eerste keer dat het voorkomt uit een rij.

Voorbeeld:

```
>>>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

Module: Built-in

2nd [catalog]

Syntax: return expression

Beschrijving: Een return-statement definieert de waarde die geproduceerd is door een functie. Python-functies geven standaard None. Zie ook: def function ():

[Fns…] > Func 1:def function():

Voorbeeld:

[Fns…] > Func 2:return

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

.reverse()

Module: Built-in

2nd [catalog]

Syntax: listname.reverse()

Beschrijving: Keert de volgorde van elementen in een rij om.

Voorbeeld:

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

round()

Module: Built-in

2nd [catalog]

Syntax: round(number, digits)

Beschrijving: Gebruik de round-functie om een getal met drijvende komma te geven dat afgerond is op het gespecificeerde aantal cijfers. De standaardwaarde is 0 en geeft het dichtstbijzijnde gehele getal.

Voorbeeld:

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

scatter(xlist,ylist,"mark")

Module: ti_plotlib

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

Beschrijving: Een rij van geordende paren uit (xlist,ylist) wordt geplot met de gespecificeerde markeringsstijl. Het lijntype en de lijngrootte worden ingesteld met plt.pen().

xlist en ylist moeten reële drijvendekommagetallen zijn en lijsten moeten dezelfde afmeting hebben.

Argument:

"mark" is het markeringsteken, als volgt:

o	filled dot (default)
+	cross
x	x
.	pixel

Voorbeeld:

Zie voorbeeldprogramma: LINREGR.

2nd [catalog]

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

importopdrachten zijn te vinden in 2nd [catalog] of in het Setup-menu van ti_plotlib.

seed()

Module: random

Syntax: seed() of seed(x) waarbij x een geheel getal is

Beschrijving: Initialiseren van de generator van toevalsgetallen.

Voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

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

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

[2nd] [catalog]

importopdrachten zijn te vinden in [2nd] [catalog]

set(sequence)

Module: Built-in

Syntax: set(sequence)

Beschrijving: Geeft een rij als een verzameling elementen. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

[2nd] [catalog]

show_plot() display > [clear]

Module: ti_plotlib

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

Beschrijving: Voert de weergave van de plot uit zoals ingesteld in het programma.

show_plot() moet achter alle plot-instelobjecten worden geplaatst. De programmavolgorde voor het plotten van objecten wordt voorgesteld door de volgorde in het menu Setup.

Selecteer voor een hulptemplate bij het plotten [New] ([zoom]) en vervolgens [Types] ([zoom]) om het programmatype "Plotting (x,y) & Text" te selecteren.

Na het uitvoeren van het programma wordt de plotweergave gewist door op [clear] te drukken. U keert dan terug naar de Shell-prompt.

Voorbeeld:

Zie voorbeeldprogramma's: COLORLIN of GRAPH.

2nd [catalog]

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

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

sin()

Module: math

Syntax: sin()

Beschrijving: Geeft de sinus van x. De hoek van het argument is in radialen.

Voorbeeld:

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

sin 3:sin()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig 3:sin()

importopdrachten zijn te vinden in
2nd [catalog]

sleep(seconds)

Module: ti_system; time

Syntax: sleep(seconds)

Beschrijving: Slaapstand gedurende een gegeven aantal seconden. Het argument voor seconden is een getal met drijvende komma.

Voorbeeld:

Voorbeeldprogramma:

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

Voer het programma TIME uit
```
>>>15.0
```

2nd [catalog]

2nd [rcl]
ti_system
A:sleep()

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

[Fns…]>Modul of
math
3:time
2:sleep()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

.sort()

Module: Built-in

Syntax: listname.sort()

Beschrijving: De methode sorteert een lijst op zijn plaats. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA) #listA bijgewerkt naar een gesorteerde lijst
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nd [list] (boven stat) List A:.sort()

2nd [catalog]

[Fns…] > List A:sort()

sorted()

Module: Built-in

Syntax: sorted(sequence)

Beschrijving: Geeft een gesorteerde lijst uit een rij.

Voorbeeld:

```
>>>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 is niet veranderd
[4,3,6,2,7,4,8,9,3,5,4,6]
```

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

2nd [catalog]

[Fns…] > List 0:sorted()

.split(x)

Module: Built-in

2nd [catalog]

Syntax:var.split(x)

Beschrijving: Methode geeft een lijst met het gespecificeerde scheidingsteken. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

sqrt()

Module: math

math Modul 1:math 3:sqrt()

Syntax: sqrt(x)

Beschrijving: Geeft de vierkantswortel van x.

2nd [catalog]

Voorbeeld:

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

[Fns…] > Modul 1:math 3:sqrt()

importopdrachten zijn te vinden in 2nd [catalog]

store_list("name",var) 1-6

Module: ti_system

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

Beschrijving: Slaat een lijst op vanaf de uitvoering van een Python-script naar een OS-lijstvariabele "name", waarbij var een gedefinieerde Python-lijst is. De lengte van de lijst moet kleiner dan of gelijk zijn aan 100.

Argument: "**name**"

Voor OS L1-L6

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

Voor aangepaste OS-lijst "name"

----- Max. 5 tekens, getallen of letters, beginnend met letters, en letters moeten hoofdletters zijn.

Voorbeelden:

"ABCDE"

"R12"

"L1" is de aangepaste lijst L1 en niet OS L1

Onthoud: Python kent dubbele precisie, dat is meer cijfers dan ondersteund wordt in het OS.

Voorbeeld:

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

Verlaat de Pyton App en druk op 2nd[L1] (boven [1]) en [enter] op het Home-scherm om lijst [L1] als {1 2 3} te zien.

2nd [catalog]

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

[Fns...]>Modul of
math
4:ti_system
3:var=store_list()

importcommando's zijn te vinden in 2nd [catalog] of in het menu van de module ti_system.

str()

Module: Built-in

2nd [catalog]

Syntax: str(argument)

Beschrijving: Converteert "argument" naar een string (tekenreeks).

[Fns…]
> Type
3 :str()

Voorbeeld:

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

sum()

Module: Built-in

2nd [list] (boven stat) List
9:sum()

Syntax: sum(sequence)

Beschrijving: Geeft de som van de elementen in een rij.

2nd [catalog]

Voorbeeld:

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

[Fns…] > List
9:sum()

tan()

Module: math

Syntax: tan(x)

Beschrijving: Geeft de tangens van x. Het hoekargument is in radialen.

Voorbeeld:

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

sin 5:tan()

[Fns…] > Modul
1:math… > Trig
5:tan()

2nd [catalog]

importopdrachten zijn te vinden in 2nd [catalog]

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

Module: ti_plotlib

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

Beschrijving: Geeft "text" in het plotgebied weer volgens de gespecificeerde "align".

rij	geheel getal 1 tot 12
"text"	string (tekenreeks) wordt ingekort als hij te lang is
"align"	"left" (standaard) "center" "right"
optioneel	1 wist de regel voorafgaand aan de tekst (standaard) 0 regel wordt niet gewist

Voorbeeld:

Zie voorbeeldprogramma: DASH1.

2nd [catalog]

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

importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

time.function

Module: Built-in

2nd [catalog]

Syntax: time.function

Beschrijving: Gebruik dit na het importeren van time om een functie te openen in de time-module.

Voorbeeld:

Zie:[Fns…] > Modul: time en ti_system modules.

title("title")

Module: ti_plotlib

2nd [catalog]

Syntax: plt.title("title")

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

Beschrijving: "title" wordt gecentreerd weergegeven op de bovenste regel van het venster. “title” wordt ingekort als hij te lang is.

Voorbeeld:

Zie voorbeeldprogramma: COLORLIN.

importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

ti_hub.function

Module: ti_hub

2nd [catalog]

Syntax: ti_hub.function

Beschrijving: Gebruik dit na het importeren van ti_hub om een functie te openen in de ti_hub-module.

Voorbeeld:

Zie:[Fns...]>Modul: ti_hub module.

ti_system.function

Module: ti_system

2nd [catalog]

Syntax: ti_system.function

Beschrijving: Gebruik dit na het importeren van ti_system om een functie te openen in de ti_system-module.

Voorbeeld:

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

texte>>>|

#verschijnt in rij 6, kolom 8 met de Shell-
prompt zoals weergegeven.
```

True

Trefwoord

Beschrijving: Geeft True wanneer de uitgevoerde bewering True is. "True" vertegenwoordigt de ware waarde van objecten van het type Boole.

Voorbeeld:

```
>>>64>=32
True
```

2nd [test] (boven math)

2nd [catalog]

[Fns…] > Ops
A:True

[a A #]

trunc()

Module: math

Syntax: trunc(x)

Beschrijving: Geeft de reële waarde x afgekapt tot een geheel getal.

Voorbeeld:

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

math Modul
1:math…
0:trunc()

2nd [catalog]

[Fns…] > Modul
1:math…
0:trunc()

importopdrachten zijn te vinden in 2nd [catalog]

try:

Trefwoord

Beschrijving: Gebruik het codeblok try om het codeblok te testen op fouten. Ook gebruikt met except en finally. Zie voor meer informatie de Python-documentatie.

2nd [catalog]

tuple(sequence)

Module: Built-in

2nd [catalog]

Syntax: tuple(sequence)

Beschrijving: Converteert een rij naar een tupel. Zie voor meer informatie de Python-documentatie.

Voorbeeld:

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

type()

Module: Built-in

2nd [catalog]

Syntax: type(object)

[Fns…]>Type>6:type ()

Beschrijving: Geeft het type van het object.

Voorbeeld:

```
>>>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)

Module: random

Syntax: uniform(min,max)

Beschrijving: Geeft een toevalsgetal x (float) zodat min <= x <= max.

Voorbeeld:

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

De resultaten variëren vanwege de willekeurige uitvoer.

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

[2nd] [catalog]

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

importopdrachten zijn te vinden in [2nd] [catalog]

wait_key()

Module: ti_system

2nd [catalog]

Syntax: wait_key()

Beschrijving: Geeft een gecombineerde toetscode die de ingedrukte toets voorstelt, samengevoegd met 2nd en/of alpha. De methode wacht tot er op een toets wordt gedrukt voordat hij terugkeert naar het programma.

Voorbeeld:

Zie:[Fns…] > Modul: time en ti_system modules.

while condition:

Trefwoord

[Fns…] Ctl 8:while condition:

Syntax: while condition:

Beschrijving: Voert de opdrachten in het volgende codeblok uit tot "condition" wordt uitgewerkt naar False.

2nd [catalog]

Voorbeeld:

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

window(xmin,xmax,ymin,ymax)

Module: ti_plotlib

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

Beschrijving: Definieert het plotvenster door het gespecificeerde horizontale interval (xmin, xmax) en verticale interval (ymin, ymax) te passen op het toegewezen plotgebied (pixels).

Deze methode moet worden uitgevoerd voordat er eventuele andere ti_plotlib-moduleopdrachten worden uitgevoerd.

De ti_plotlib Properties vars, xmin, xmax, ymin, ymax worden bijgewerkt naar de argumentwaarden. De standaardwaarden zijn (-10, 10, -6.56, 6.56).

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

2nd [catalog]

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

importopdrachten zijn te vinden in 2nd [catalog] of in het ti_plotlib Setup-menu.

met

Trefwoord

Beschrijving: Zie voor meer informatie de Python-documentatie.

2nd [catalog]

xmax default 10.00

Module: ti_plotlib

Syntax: plt.xmax default 10.00

Beschrijving: Gespecificeerde variabele voor window-argumenten gedefinieerd als plt.xmax.

Standaardwaarden:

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

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

xmin default -10.00

Module: ti_plotlib

Syntax: plt.xmin default -10.00

Beschrijving: Gespecificeerde variabele voor window-argumenten gedefinieerd als plt.xmin.

Standaardwaarden:

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

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

yield

Trefwoord

2nd [catalog]

Beschrijving: Gebruik yield om een functie te beëindigen. Geeft een generator. Zie voor meer informatie de Python-documentatie.

ymax default 6.56

Module: ti_plotlib

2nd [catalog]

Syntax: plt.ymax default 6.56

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

Beschrijving: Gespecificeerde variabele voor window-argumenten gedefinieerd als plt.ymax.

Standaardwaarden:

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

ymin default -6.56

Module: ti_plotlib

Syntax: plt.ymin default -6.56

Beschrijving: Gespecificeerde variabele voor window-argumenten gedefinieerd als plt.ymin.

Standaardwaarden:

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

Voorbeeld:

Zie voorbeeldprogramma: GRAPH.

2nd [catalog]

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

Importopdrachten zijn te vinden in 2nd [catalog] of in het menu ti_plotlib Setup.

Symbolen

@

Operator

Beschrijving: Decorator – Zie de algemene Python-documentatie voor meer informatie.

alpha [θ]
(boven 3)

2nd [catalog]

<<

Operator

2nd [catalog]

Syntax: x<<n

Beschrijving: Bitsgewijze verschuiving naar links met n bits.

>>

Operator

2nd [catalog]

Syntax: x>>n

Beschrijving: Bitsgewijze verschuiving naar rechts met n bits.

|

Operator

2nd [catalog]

Syntax: x|y

Beschrijving: Bitsgewijze or.

&

Operator

2nd [catalog]

Syntax: x&y

Beschrijving: Bitsgewijze and.

^

Operator 2nd [catalog]

Syntax: x^y

Beschrijving: Bitsgewijze exclusive or.

~

Operator 2nd [catalog]

Syntax: ~x

Beschrijving: Bitsgewijze not; de bits van x geïnverteerd.

x<=y

Operator

Syntax: x<=y

Beschrijving: Vergelijking; x is kleiner dan of gelijk aan y.

Voorbeeld:

```
>>>2<=5
True
>>>3<=0
False
```

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

2nd [catalog]

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Operator

Syntax: x<y

Beschrijving: Vergelijking; x is kleiner dan y.

Voorbeeld:

```
>>>6<10
True
>>>12<-15
False
```

math
1:math > Ops
6:x<y

2nd [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Operator

Syntax: x>=y

Beschrijving: Vergelijking; x is groter dan of gelijk aan y.

Voorbeeld:

```
>>>35>=25
True
>>>14>=65
False
```

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

2nd [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Operator

Syntax: x>y

Beschrijving: Vergelijking; x is groter dan y.

Voorbeeld:

```
>>>35>25
True
>>>14>65
False
```

math
1:math > Ops
4:x>y

2nd [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Operator

Syntax: x!=y

Beschrijving: Vergelijking; x is niet gelijk aan y.

Voorbeeld:

```
>>>35!=25
True
>>>14!=10+4
False
```

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

2nd [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Operator

Syntax: x==y

Beschrijving: Vergelijking; x is gelijk aan y.

Voorbeeld:

```
>>>75==25+50
True
>>>1/3==0.333333
False
>>>1/3==0.3333333 #equal to stored Python value
True
```

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

2nd [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Operator

Syntax: x=y

Beschrijving: y is opgeslagen in variabele x

Voorbeeld:

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3 #Gebruik [ ^ ] op het toetsenblok voor **
>>>print(B)
8
```

sto→

math
1:math > Ops
1:x=y

2nd [catalog]

[Fns…] > Ops
1:x=y

[a A #]

\

Scheidingsteken

Beschrijving: Backslash-teken.

2nd [catalog]

[a A #]

\t

Scheidingsteken

Beschrijving: Tabspatie tussen strings (tekenreeksen) of tekens.

2nd [catalog]

\n

Scheidingsteken

Beschrijving: Nieuwe regel om de string (tekenreeks) netjes op het scherm weer te geven.

2nd [catalog]

' '

Scheidingsteken

2nd [mem]
(boven +)

Beschrijving: Plakken van twee enkele aanhalingstekens.

Voorbeeld:

2nd [catalog]

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

[a A #]

" "

Scheidingsteken

alpha ["]
(boven +)

Beschrijving: Plakken van twee dubbele aanhalingstekens.

Voorbeeld:

2nd [catalog]

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

[a A #]

Bijlage

Geselecteerde TI-Python Built-in, Trefwoorden en module-inhoud

Geselecteerde TI-Python Built-in, Trefwoorden en module-inhoud

Ingebouwde functies

Ingebouwde functies	Ingebouwde functies	Ingebouwde functies
__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

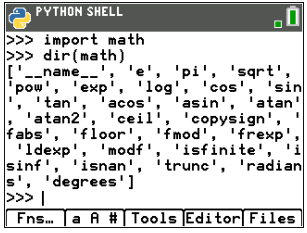
Ingebouwde functies	Ingebouwde functies	Ingebouwde functies
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <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'>

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

trefwoorden

trefwoorden	trefwoorden	trefwoorden
False	elif	lambda
Geen	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>		

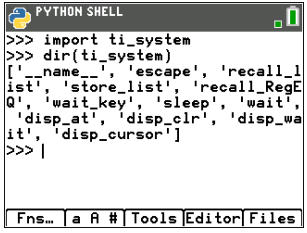
random

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

time

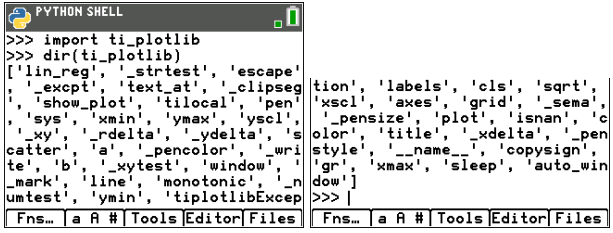
time	time	time
__name__		
monotonic		
sleep		
struc_time		

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



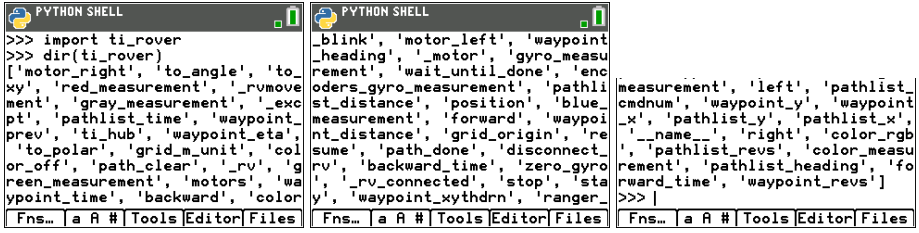
ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle

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		

.

Algemene informatie

Online Help

education.ti.com/eguide

Selecteer uw land voor meer productinformatie.

Neem contact op met TI Ondersteuning

education.ti.com/ti-cares

Selecteer uw land voor technische en andere ondersteuningsbronnen.

Service- en garantie-informatie

education.ti.com/warranty

Selecteer uw land voor meer informatie over de duur en voorwaarden van de garantie of over de productservice.

Beperkte garantie. Deze garantie heeft geen invloed op uw wettelijke rechten.