Python Programmierbüchlein für TI-Nspire™ CX II-T CAS – Einführung –

Didier Deses (T^3 Flandern)
Aus dem Niederländischen übersetzt und für den TI-Nspire™ CX II-T CAS bearbeitet von Josef Böhm

T³ EUROPE

Teachers Teaching with Technology™

Wiskak Python TI-84+ Programmeerboekje
Aus dem Niederländischen übersetzt und für den TI-Nspire™ CX II-T CAS bearbeitet von Josef Böhm

Dieses und weiteres Material steht Ihnen zum pdf-Download bereit: **www.t3europe.eu** sowie unter **www.ti-unterrichtsmaterialien.net**

Ich habe vor einiger Zeit schöne Skripten von Kollegen von T³-Flandern aus dem Holländischen übersetzt und für frühere Versionen des TI-Nspire adaptiert (*Einführung zum TI-Nspire CX CAS, Aufgaben zur Analysis mit dem TI-Nspire CX CAS, Mathematik in wirtschaftlichen Zusammenhängen, Einführung ins Programmieren mit LUA*), die alle auf der Materialienbank von T³-Deutschland zu finden sind (www.ti-unterrichtsmaterialien.net).

Anlässlich einer Mail zu belgischen Kollegen erhielt ich den Link zu einer Programmsammlung für die Programmiersprache Python, die in Belgien schon weit verbreitet ist. Das Skriptum *Wiskak Python TI84+ Programmboekje* von Didier Deses hat mir sofort sehr gut gefallen. Ich habe eine Idee, kurze „Einschirmprogramme“ anzubieten, außerordentlich gelungen gefunden. Mit langen Programmen kann man anfänglich Interessierte oft abschrecken.

Auf Anfrage bei TI wurde ich eingeladen, das Skript zu übersetzen und für den TI-Nspire™ CX II-T CAS zu bearbeiten (da das Original ja für den TI84+ gedacht ist). Wir bekamen dafür die Erlaubnis aus Belgien, wofür herzlich gedankt sei.

So machte ich mich an die Arbeit. Didier Deses hat dieses Büchlein für seine Schüler geschrieben und sehr viele Ideen und Anregungen hineingesteckt. Ich habe einige Stellen anpassen und zusätzliche Erklärungen anbringen müssen, die Didier sicherlich seinen Schülern gegeben hat. So werden ein paar Befehle verwendet, die weder im deutschen Manual, noch in der Menüstruktur von Python zu finden sind. Die nötige Hilfe findet sich in den unten angegebenen und sehr empfehlenswerten Internetquellen.

Ein paar wertvolle Tipps werden uns von Didier Deses im Abschnitt 2.1. verraten. Ein Problem bereitete Kopfzerbrechen: es kann notwendig sein, eine Abfrage im Lauf eines Programms mit einer Liste zu beantworten. Listen werden in eckigen Klammern geschrieben. Es hat sich nun herausgestellt, dass hier – im Gegensatz zum TI-84+ – die [,]-Zeichen am Handheld nur im Editor und in der Shell angesprochen werden können. Die Tastenkombinationen mit der ctrl-Taste funktionieren am Handheld und auch in der Emulation am PC nicht (über die PC-Tastatur schon!). Beachte bitte den Hinweis dazu in 2.2.22.

Herzlichen Dank an Didier Deses, sowie an Guido Herweyers und Veit Berger für ihre wertvolle Unterstützung, sowie an Dirk Ritschel von TI-Deutschland für die Ermunterung zu dieser Übersetzung bzw. Aufbereitung für den TI-Nspire™ CX II-T CAS. Natürlich hoffen wir alle, dass es nicht dabei bleibt, dass sich die Leser sich damit begnügen, die Programme abzutippen oder gleich auf ihr Gerät zu laden, sondern, dass sie die eine oder andere Anregung aufnehmen und auch selbst Freude und Neugierde beim Programmieren mit Python finden werden.

Josef Böhm, im Februar 2022

Alle hier vorgestellten Programm sind in der tns-Datei PyKurz.tns gesammelt aufzufinden.

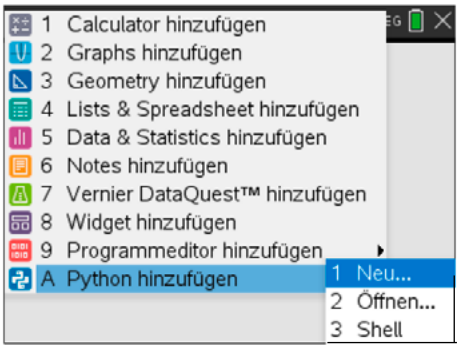
https://www.programiz.com/python-programming
https://docs.python.org/3/library/
https://docs.python.org/3/tutorial/

Einführung in Python auf dem TI-Nspire™ CX II-T CAS

1.1 Die Python App

Auf den TI-Nspire™ CX II-T CAS Rechnern hat man die Wahl zwischen zwei Programmiersprachen: TI-BASIC und Python.

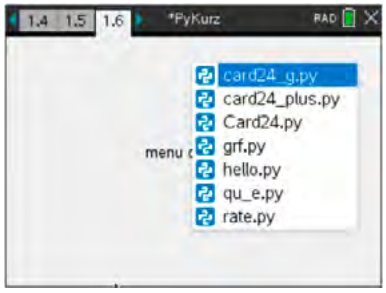
Über menu und Menüpunkt 9 wählt man TI-BASIC und über Menüpunkt A die Programmiersprache Python. Der Editor zum Schreiben eines Programms oder die Benutzerplattform Shell lassen sich sofort öffnen.

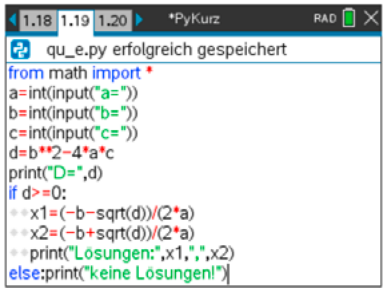


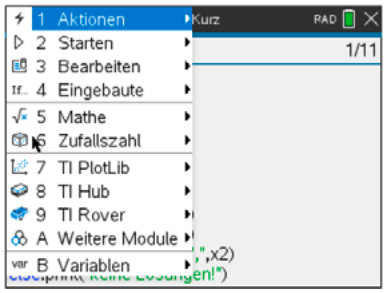
Mit ctrl doc auf dem Handheld und über Einfügen > Python am PC gelangt man auch in die Python App. Falls schon Programme vorhanden sind, kann man diese von hier aus gleich öffnen.

Option 2 zeigt die schon vorhandenen Programme, die in den Editor geladen werden können. Über menu > 2 Starten oder kürzer mit ctrl R werden die Programme gestartet und die Ergebnisse zeigen sich dann in der Shell oder auf dem Grafikschirm.

In einer kurzen Übersicht wollen wir die Menüs vorstellen, die für das Programmieren wichtig sind. Im Editor öffnet sich das Angebot der Dokumentwerkzeuge wieder über das menu.

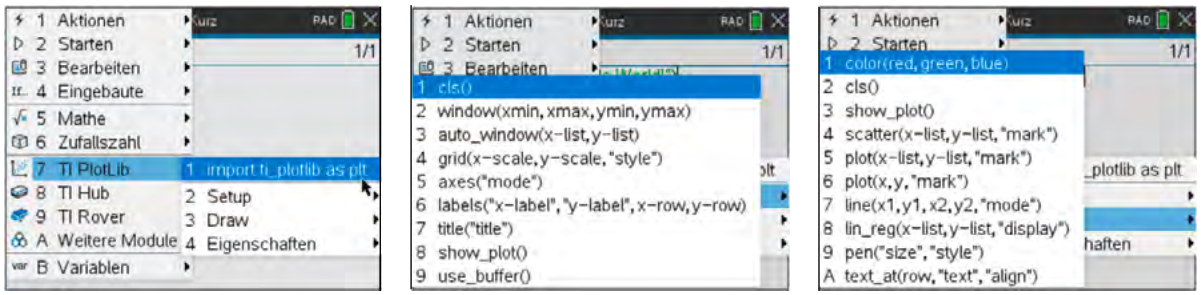






In der Software findet sich das gleiche Angebot ganz links in der Werkzeugpalette Dokumente.

Teile von Python werden in einzelnen Modulen aufgerufen. So befinden sich z.B. alle mathematischen Funktionen im Modul Mathe. Das Modul wird geladen mit from math import *. Dann erst kann z.B. die Funktion sqrt verwendet werden (siehe obiges Beispiel qu_e.py). Neben einer Anzahl von weiteren Modulen ist auch das Modul ti_plotlib verfügbar. Damit kann man auf die grafischen Möglichkeiten des TI-Nspire™ CX II-T CAS zurückgreifen. Es wird geladen mit import ti_plotlib as plt. Dann werden alle Grafikbefehle mit plt. eingeleitet.



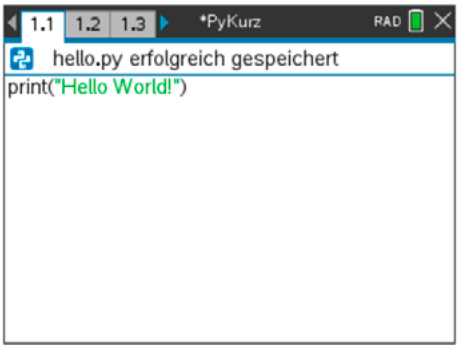
TI PlotLib 2 Setup 3 Draw

Wie man an den Befehlen im Menüsystem erkennen kann, ist die Python-Version auf dem TI-Nspire™ CX II-T CAS ziemlich umfangreich in Bezug auf die Möglichkeiten. Dabei reden wir noch gar nicht von den anderen Modulen, z.B. TI Rover, mit denen ein Minifahrzeug programmiert und gesteuert werden kann.

1.2 Programmiergrundlagen

Programm 1 hello.py

Es ist schon Tradition, dass das erste Programm, das man in einer neuen Programmiersprache schreibt, die Botschaft *Hello World* auf den Bildschirm bringt. In Python ist das sehr einfach und das Programm besteht aus nur einer Zeile. Probiere das einmal!



Mit ctrl B wird das Programm auf syntaktische Fehler geprüft und gespeichert und dann mit ctrl R ausgeführt.

Bemerkung: Jede Programmiersprache hat neben Variablen und Befehlen zwei Kontrollstrukturen:

- bedingte Anweisungen
- Schleifen

1.2.1 Bedingte Anweisungen

Die bedingte Anweisung gibt die Möglichkeit, einen Teil des Programmcodes nur dann auszuführen, wenn eine oder mehrere Bedingungen erfüllt sind. In Python erfolgt dies mit der if-Konstruktion. Diese hat verschiedene Formen, unter anderen:

```
If Bedingung:
  Anweisung(en)
```

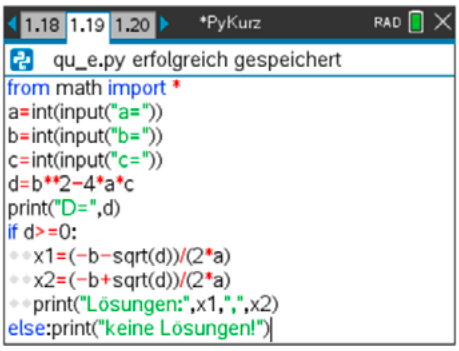
oder

```
If Bedingung:
  Anweisung(en)
else:
  Anweisung(en)
```

Es ist zu beachten, dass die Anweisung(en) immer durch eine Einrückung gruppiert werden müssen. Sie bilden einen Codeblock.

Programm 2 qu_e.py

Wir zeigen ein kurzes Beispiel eines Pythonprogramms, mit dem man quadratische Gleichungen in $\mathbb{R}$ löst.



Das soll nun kurz besprochen werden:

- In Python befinden sich alle mathematischen Funktionen wie sin(), cos(), tan(), sqrt(), exp(), … im Modul 5 Mathe. Mit der Programmzeile from math import * können alle diese Funktionen eingesetzt werden.
- In den nächsten drei Zeilen werden die Koeffizienten abgefragt. Da input() eine Zeichenkette zurückgibt, muss diese erst in eine Zahl umgewandelt werden. Hier erfolgt dies für ganze Zahlen mit int(). Wenn auch Dezimalzahlen eingegeben werden sollen, muss man float() nehmen.
- Dann wird die Diskriminante D berechnet, wobei zu beachten ist, dass b^2 als b**b einzugeben ist. Anschließend wird ihr Wert mit der print()-Anweisung ausgegeben.
- Mit der if-Konstruktion wird zwischen den Fällen $D \geq 0$ und $D < 0$ unterschieden.
- Im ersten Fall werden die beiden Lösungen x_1 und x_2 berechnet und ausgegeben und im zweiten Fall wird angezeigt, dass es keine (reellen) Lösungen gibt. Die Ausgaben (Text oder Zahlen) sind jeweils durch ein Komma zu trennen.

1.2.2 Schleifen

Die zweite Steuermöglichkeit in jeder Programmiersprache sind die Schleifen. Die erste von diesen ist die for-Schleife:

```
for i in range (n)
  Anweisungsblock
```

oder

```
for i in range (a,b)
  Anweisungsblock
```

Die zweite Schleife ist die while-Schleife. (Erste Anwendung in 2.2.1)

```
while Bedingung
… Anweisungsblock
```

Die Anweisungen im for-Block werden für alle Werte für die Variable i von $i = 0$ bis $i = n - 1$ ausgeführt. Im zweiten Fall läuft i von a bis $b - 1$. Dabei müssen alle Parameter ganze Zahlen sein. Der Block in der while-Schleife wird so oft wiederholt, so lange die Bedingung erfüllt ist. Wenn eine Reihe von Befehlen in einer Schleife wiederholt wird, nennt man dies eine **Iteration**.

Programm 3 rate.py

Es folgt ein ausgearbeitetes Programmbeispiel für das Spiel „Größer – Kleiner“.

Hier ist wieder eine kurze Erklärung:

- Die Befehle zur Erzeugung von Zufallszahlen stehen im Modul 6 Zufallszahl. Das Modul random wird importiert.
- Nach der Ausgabe der Überschrift erzeugt der Computer eine ganze Zufallszahl zwischen 1 und 100 (mit dem Befehl randint(0,100)).
- Dann beginnt die for-Schleife, die den Spieler fünfmal raten lässt.
- Nach der Eingabe des Spielers wird ihm mit Hilfe der if-Abfrage der passende Hinweis oder eine Erfolgsantwort gegeben.
- Wenn der Spieler die Zahl erraten hat, wird nach Ausgabe von „Erraten!“ mit break die for-Schleife abgebrochen und die Zahl angezeigt. Wenn nach fünf Versuchen die gesuchte Zahl noch nicht gefunden wurde, wird nur diese ausgegeben.

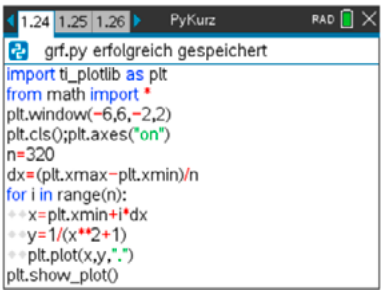
1.3 Das Grafikmodul

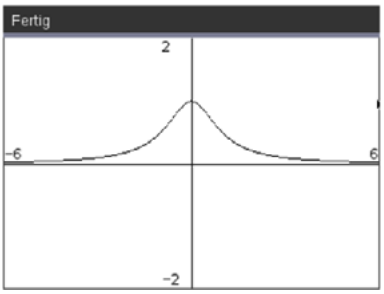
Das Modul ti_plotlib wird über menu 5 angesteuert. Damit lassen sich die grafischen Möglichkeiten ansprechen.

- Am einfachsten wird es über import tiplotlib as plt. geladen. Dann findet man im Menü verschiedene Befehle:
- plt.window(xmin,xmax,ymin,ymax) stellt das Koordinatensystem her. Mit plt.xmin, plt.max, lassen sich später die Begrenzungen abfragen bzw. abändern.
- plt.cls() leert den Grafikschirm.
- plt.axes("on") zeichnet die Achsen und gibt die Endwerte an. Mit dem Parameter axes oder nur plt.axes werden nur die Achsen gezeichnet.
- plt.grid(xycl,yscl, "solid") zeichnet das Koordinatengitter auf den Schirm.
- plt.plot(x,y, ".") zeichnet einen dünnen Punkt an die Stelle (x,y), mit "o" wird der Punkt dick gezeichnet.
- plt.line(x1,y1,x2,y2, "") zeichnet eine Strecke von (x_1,y_1) nach (x_2,y_2). Mit "arrow" lässt sich auch ein Pfeil zeichnen.
- plt.color(r,g,b) verwendet *r,g,b*-Werte, um die Zeichenfarbe einzustellen (siehe 2.1).

Programm 4 grf.py

Es folgt einfaches Programm, um einen Funktionsgraph zu zeichnen Hier soll der Graph von $f(x)=1/(x^2+1)$ für $-6\le x\le 6$ gezeichnet werden. Das lässt sich folgendermaßen verwirklichen.





- Die Module tI_plotlib und math werden geladen.
- Nachdem der Schirm gelöscht wurde, wird das Koordinatensystem festgelegt und gemeinsam mit den Achsen gezeichnet.
- Wir wollen *n* Punkte der Kurve zeichnen. Daher wird die *x*-Achse in *n* Teile mit der Länge *dx* geteilt.
- In einer for-Schleife werden die *n* Punkte (*x*,*y*) berechnet und mit …
- … plt.plot(x,y,".") werden sie auch gezeichnet.
- plt.show_plot() bringt den fertigen Graph auf das Display.

2.1 Programmiertricks

Wir zeigen einige Programmbeispiele. Es ist zu beachten, dass wir hier manchmal gegen den *Zen of Python* sündigen, da alle Programme so geschrieben sind, dass sie auf einen (Taschenrechner-) Schirm passen. Dafür muss man öfters mehrere Anweisungen, getrennt durch ein Semikolon, in eine Zeile schreiben. Wenn man ein Stück Code so kurz wie möglich hält, nennt man das *code golfing*. Dafür muss man einige Kniffe anwenden, um den Code zu verkürzen oder effizienter zu machen. Einige davon werden hier besprochen.

■ Definition von mehreren Variablen: Wenn mehreren Variablen Werte zugewiesen werden sollen, schreibt man im Editor:

```
a=1
b=-1
c=2
```

Das lässt sich in einer Zeile zusammenfassen als

```
a,b,c=1,-1,2
```

■ Initialisieren von Listen: Wir werden häufig `[0]*5` verwenden, um z.B. die Liste `[0,0,0,0,0]` zu definieren.

■ Module importieren: Module lassen sich auf zwei Arten importieren:

```
import … as …
```

oder

```
from … import *
```

Die Anweisungen aus den Modulen werden dann verwendet wie folgt:

```
import ti_plotlib as plt        → plt.cls(), plt.plot(x,y)
```

oder

```
from ti_plotlib import *        → cls(), plot(x,y)
```

Die erste Version ist wohl deutlicher, nimmt aber viel Platz ein. Mit der zweiten Version lassen sich die Anweisungen direkt einsetzen. Das macht den Code ein Stück kürzer. Die Menüs des TI-Nspire™ CX II-T CAS verwenden standardmäßig Version 1.

■ Verkürzte Farbpalette: Python verwendet *r*, *g*, *b*-Werte, um Farben zu kodieren. Jeder dieser Werte reicht von 0 bis 255, daher sind $256^3 \approx 16$ Millionen Farben erzeugbar. Es ist aber meist so, dass wir nur ein paar kontrastreiche Farben verwenden wollen. Dann kann man einen „Malkasten“ zusammenstellen.

```
frgb = [(0,0,0),(255,0,0,0),(0,255,0),(0,0,255)]
```

Die Liste frgb der Zahlentripel enthält die Farben Schwarz, Rot, Grün und Blau. Die Farbe Rot wird dann mit plt.color(frgb[1]) angesprochen. (Hinweis: die Indizierung von Listen beginnt in Python bei 0!). Es wäre unhandlich, eine ganze Liste von Farben einzugeben. Das geschieht einfacher: aus dem Zahlentupel

```
fa=(0,0,0,255,0,0)
```

werden mit fa[k%4:k%4+3][1] jedes Mal drei aufeinanderfolgende Elemente ausgewählt. Das ergibt die folgenden Farben:

$$k = 0 : \{\underbrace{0,0,0}_{\text{schwarz}},255,0,0)$$
$$k = 1 : \{0,\underbrace{0,0,255}_{\text{blau}},0,0)$$
$$k - 2 : \{0,0,\underbrace{0,255,0}_{\text{grün}},0) \text{ ¶}$$
$$k - 3 : \{0,0,0,\underbrace{255,0,0}_{\text{rot}})$$

Auf diese Weise kommen wir zu den gleichen vier Farben wie vorher. Mit den untenstehenden Tupeln können wir auf analoge Weise von drei bis zu acht Farben definieren.

```
farb3=(0,0,255,0,0)
farb4=(0,0,0,255,0,0) # das ist obige fa
farb6=(0,0,255,0,255,255,0,0)
farb7=(0,0,0,255,0,255,255,0,0)
farb8=(0,0,0,255,0,255,255,255,0,0)
```

[1] Das %-Zeichen bedeutet die mod-Funktion (Ganzzahliger Rest nach Division):
20%3 = 2 entspricht mod(20,3) oder 20 modulo 3

T³ Teachers Teaching with Technology

Netzwerk
Das T³ Lehrerfortbildungsnetzwerk richtet sich an Sie, an Lehrerinnen und Lehrer, die sich zum sinnvollen Einsatz digitaler Werkzeuge im MINT-Unterricht austauschen und weiterentwickeln wollen. T³ Deutschland ist Teil des internationalen T³ Netzwerks.

Fortbildungen
T³ Deutschland bietet Ihnen pädagogisch-didaktische Unterstützung in Form von schulinternen Fortbildungen, Online-Seminaren und Tagungen an.

Materialien
Aufgabenbeispiele, Tutorials, Videos und mehr nützliche Materialien für Ihren MINT-Unterricht stellen wir auf der Materialdatenbank kostenlos zur Verfügung.

Der T³ EduBlog bietet exklusive Interviews, inspirierende Erfahrungsberichte und mehr

Informieren Sie sich. Machen Sie mit!

Nehmen Sie Kontakt zu uns auf unter:
www.t3deutschland.de | info@t3deutschland.de

Abonnieren Sie unseren Newsletter!

www.t3europe.eu **@T3Europe** **T3 Europe**

TI-Nspire™ CX CAS Technologie

Ob Handheld, Software (Win/Mac) oder Tablet (Win/iPad) - alle Produkte sind einzeln oder als integrierte Lösung einsetzbar. Passendes Zubehör unterstützt den fächerübergreifenden Einsatz in Mathematik, Informatik, Naturwissenschaft und Technik (MINT).

www.tinspirecas.de

Praxisorientierte Unterrichtsmaterialien

Nützliche Aufgabenbeispiele für Ihren Unterricht, kostenlose Downloads und Hinweise auf Verlagspublikationen finden Sie auf der TI Materialdatenbank, auch ganz speziell zur TI-Nspire™ CX Technologie.

Schauen Sie mal rein:
TI Materialdatenbank: www.ti-unterrichtsmaterialien.net

- Nutzen Sie beispielsweise unser kostenloses Ausleihprogramm!
- Ausführliche Produkt- und Serviceinformationen sowie Bezugsquellen finden Sie auf unseren TI Webseiten education.ti.com/de
- Die TI Schulberater unterstützen Sie gerne bei allen Fragen rund um den Einsatz von TI Rechnern im Unterricht: schulberater-team@ti.com

Abonnieren Sie unseren Newsletter!

www.youtube.com/TIedtechDE

education.ti.deutschland

@TIEducationDE

www.t3europe.eu

education.ti.com

T³ EUROPE

Teachers Teaching with Technology™

Texas Instruments