**Skript**

**Python im Mathematikunterricht**

**von**

**Georg Sahliger**

Mainz, 29.03.2021

Inhaltsverzeichnis

[0 Vorwort 1](#_Toc6582000)

[1. Installation von Python 1](#_Toc6582001)

[2. Erste Befehle – Vorübung 2](#_Toc6582002)

[3. Das erste Programm 3](#_Toc6582003)

[3 Programm Addition 4](#_Toc6582004)

[4. if – Anweisung 5](#_Toc6582005)

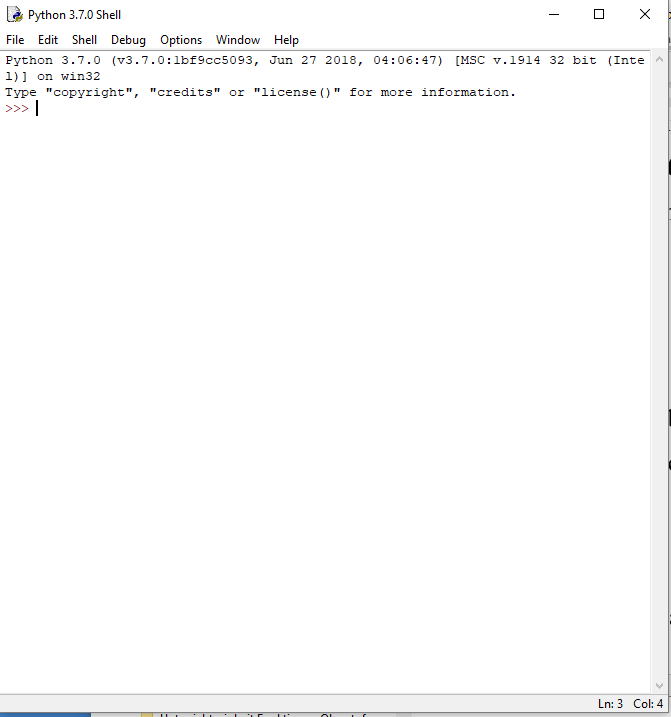
# 0 Vorwort

Im folgenden Skript geht es um eine Einführung in Python und wie man diese Programmiersprache im Mathematikunterricht einsetzen kann. Gedacht ist es für Schüler und Neueinsteiger. Es soll helfen, sich in die Denkweise der Informatik hineinzuversetzen und hoffentlich auch den Spaß an Informatik und Mathematik zu fördern.

# Python auf den Schulrechnern bzw. Installation von Python

Auf den Schulrechnern ist Python schon installiert, daher muss man unten im Suchfenster „IDLE“ eingeben und das Programm öffnen. Es erscheint das sogenannte „Shellfenster“. Hier kann man nun einige Befehle ausprobieren und die Arbeitsweise von Python verstehen. Gehe hierfür zu Kapitel 2: Erste Befehle

Installation:

Unter www.python.org kann man das Programm bei „Downloads“ herunterladen und entsprechend den Anweisungen installieren. Nach der Installation öffnen wir das Programm „IDLE“ und erhalten folgendes Fenster, das sogenannte Shell-Fenster.

Hier können wir schon erste Befehle eingeben und ausprobieren, was passiert.

Hilfen im Internet:

Neben Youtube Tutorials gibt es viele gute, auch kostenlose Onlinekurse, z.B. <https://cscircles.cemc.uwaterloo.ca/using-website-de/>

# Erste Befehle – Vorübung

Wir geben folgenden Befehle ein und schauen, was passiert. Nach jedem Befehl bestätigen wir mit „Enter“. Wichtig ist es, den Befehl genau abzuschreiben und beispielsweise Groß- und Kleinschreibung zu beachten.

|  |  |  |
| --- | --- | --- |
| Befehl | Bemerkung |  |
| #Kommentar | Setzt man ein Doppelkreuz #, dann färbt Python alles dahinter rot. Das bedeutet für Python, dass dies ein Kommentar ist und nicht weiter beachtet wird. |  |
| print(“Hallo Welt“) | Ob man a = “Hallo“ oder  a = ‘hallo‘ schreibt ist egal.  Wie man sieht färbt Python den Befehl in einer anderen Farbe, dies zeigt, dass erkannt hat, dass „print“ ein Befehl ist. |
| print(2) | Zahlen kann man ohne Anführungszeichen ausgeben. |
| a = 5  print(a) | Nun arbeiten wir mit Variablen. Hier weisen wir a einen Wert 5 zu und lassen den uns ausgeben. |
| b = 7  a + b |  |
| a = 4  b = 5  c = a + b  print(c) |  |
| a = “Hallo“  b = ‘‘Welt‘‘  a + b | Dies geht nicht nur mit Zahlen, sondern auch mit Zeichen. |
| a = input(“Wie alt bist du? “)  print(“Du bist “ + a) | Nach Ausführen muss man jetzt eine Zahl eingeben! |
| Probiere folgende Rechnungen aus:  6:3 entspricht bei Python 6/3  3² entspricht bei Python 3\*\*2  entspricht bei Python 16\*\*(1/2) Dies entspricht der Schreibweise, dass man Wurzel auch als hoch ½ schreiben kann. Also  Ebenso entspricht (-8)\*\*(1/3)  Die Modulorechnen schaut, ob es bei einer Division einen Rest gibt:  Beispiel 14 :3 = 4 plus Rest 2  Diesen Rest rechnet man bei Python mit 13%3 aus. | | |

# Das erste Programm

Klicke auf File => New File und es öffnet sich ein zweites Fenster. Das Code Fenster.

In dieses Fenster kann man nun ein größeres Programm schreiben.

Beginnen wir mit einem einfachen Programm:

Man gibt einen Namen ein und Python begrüßt einen mit Namen.

Will man das Programm starten, speichert man das Programm mit STRG + S ab (oder bei „File“ => speichern) und drückt anschließend auf F5.

Dabei überprüft Python, ob alles richtig geschrieben wurde. Wenn alles richtig ist, wird das Programm gestartet.

Anmerkung: Wörter und Buchstaben nennt der Programmierer „Strings“.

|  |  |  |
| --- | --- | --- |
| #Programm Addition  a = input("Wie ist dein Name? ")  print ("Hallo " +a) | Der Programmname  Hiermit gibt man einen Namen ein speichert diesen unter der Variablen a ab.  „print“ schreibt das Wort „Hallo“ und den Namen der unter a gespeichert ist. | Ein Bild, das Screenshot enthält.  Automatisch generierte Beschreibung |

Aufgabe: Schreibe ein weiteres Programm, bei dem man Strings eingeben kann. Was passiert, wenn man zwei Strings addiert?

Probiere aus:

a = input("Name 1? ")

b = input("Name 2? ")

print (a+b)

# 3 Programm Addition

|  |  |  |
| --- | --- | --- |
| #Programm Addition  a = input("Gib die erste Zahl ein: ")  b = input("Gib die zweite Zahl ein: ")  c = int(a) + int(b)  print ("Die Summe beträgt: ")  print (c) | Der Programmname  Hiermit gibt man eine Zahl ein und speichert diese unter der Variablen a bzw. b ab.  gibt man z.B. eine 4 ein, dann weiss Python allerdings nicht, dass das eine Zahl ist.    Hiermit wandelt man a in eine Zahl INTEGER um.  Mit print kann man etwas schreiben. |  |

# if – Anweisung

Mit dem folgendem Programm probieren wir if-Anweisungen aus:

Hier lernen wir kennen, dass es wichtig ist bei Python, die Befehle einzurücken. Also nicht

if a == "mathe":

print ("Du bist toll")

sondern:

if a == "mathe":

print ("Du bist toll")

alles wann nun eingerückt ist gehört zum if-Befehl

Am besten nur mit der Tabulatortaste einrücken und nicht mit Leerzeichen. Wenn man beides verwendet, kann es passieren, dass das Programm nicht funktioniert.

|  |  |  |
| --- | --- | --- |
| # if - Anweisungen  a = input("Was ist dein Lieblingsfach? ")  if a == "mathe":    print ("Du bist toll")  else:  print ("Schade") | Beachte, dass man „if“ klein schreiben muss und zwei  = = Zeichen schreiben muss.  Um zu verdeutlichen, was alles zur if-Anweisung gehört, muss man die Befehle einrücken.  Was passiert, wenn man „Mathe“ groß schreibt?  Um diesen Fehler zu beheben, kann man eine zweite Zeile mit  elif a == “Mathe“:  print(“Toll“)  Das bedeutet also:  WENN …. schreibe…  ODER … schreibe…  ansonsten …. schreibe | Ein Bild, das Screenshot enthält.  Automatisch generierte Beschreibung  Ein Bild, das Screenshot enthält.  Automatisch generierte Beschreibung |
| Schreibe ein Programm, bei dem man eine Zahl zwischen 1 und 20 erraten muss. Das Programm sagt einem nach jedem Versuch, ob die Zahl größer oder kleiner ist. Am Schluss wird gezählt und ausgegeben, wie viele Versuche man benötigt hat. | | |

# For Schleifen

Mit For Schleifen kann man Befehle beliebig oft wiederholen. Auch hier ist das einrücken wieder wichtig.

|  |  |  |
| --- | --- | --- |
| # For Schleife  for i in range (0,10):  print ("Mathe ist toll") | Hier gibt es eine Variable namens i .  „in range(1,10)“ zählt von 0 bis 10  und führt 10 mal die Printanweisung aus.  Varianten:  „in range (1, 10, 2)“ zählt von 1 bis 10 in 2er Schritten  In range (20, 10, -1) zählt rückwärts von 20 bis 10 |  |
| Beispielprogramm, das eine Zahl beliebig oft mit sich selbst addiert. | | |

1. Module in Python

Bisher haben wir die Grundbefehle kennengelernt. Das schöne bei Python ist nun, dass schon viele Programme geschrieben sind, um das Programmiererleben zu erleichtern. Solche Programme nennt man „Module“.

Als erstes lernen wir das Modul „math“ kennen.

Bisher haben wir wurzeln immer umständlich berechnet.

entspricht bei Python 16\*\*(1/2) Den Befehlt „Wurzel“ oder englisch „Sqare root“

kennt Python. Hier kann man nun ein Modul importieren, das viele ein Programm für Wurzeln enthält!

# Zusammenfassung der Befehle

Jetzt können wir schon einige Programm selbst schreiben, daher nun einige Befehle, die schon bekannt sind, bzw. noch eingeführt werden:

|  |  |
| --- | --- |
| **Kommentar**  # …. | **Schleifen**  For i in range (5):  break  continue |
| **Ein- und Ausgabe**  a = input()  a = input(„Was ist dein Lieblingstier? „)  print(a) |  |
| **if – Anweisung**  if a == "mathe":    print ("Du bist toll")  elif a ==“Mathe“:  print ("Du bist toll")  else:  print ("Schade") |  |
| **Module import**  from math import \*  from math import sqr  from random import randint  from tkinter import \*  from pygame import \* |  |

# For Schleife

Mit Schleifen kann man Befehle beliebig oft wiederholen.

# Anhang

### Primfaktoren bestimmen

**def** prim(n):

*'''Calculates all prime factors of the given integer.'''*

**from** **math** **import** sqrt

pfactors = []

limit = int(sqrt(n)) + 1

check = 2

num = n

**if** n == 1:

**return** [1]

**for** check **in** range(2, limit):

**while** num % check == 0:

pfactors.append(check)

num /= check

**if** num > 1:

pfactors.append(num)

**return** pfactors

Links

<http://www.michael-holzapfel.de/progs/python/python_beisp.htm>