1 Python - Eine kleines Tutorial

Python ist eine moderne, freie Programmiersprache und bietet viele Funktionen die einem Anfänger den Einstieg vereinfachen. Python ist eine interpretierte Sprache. Im Gegensatz zu compilierten Sprachen, werden interpretierte Sprachen nicht direkt vom Computer sondern von einem zusätzlichen Programm, dem Interpreter, ausgeführt. Dadurch werden die Programme zwar etwas langsamer, dafür bist du bei der Entwicklung viel flexibler. Weitere Vorteile vom Python sind:

- Freie Software: Jeder kann Python benutzen, weitergeben, verändern und auch seine Veränderungen weitergeben.
- viele Module mit Funktionen für ein breites Spektrum an Aufgaben.
- diverse Hilfsmittel, die helfen sauberen Code zu schreiben und Fehler zu finden

1.1 Python starten

Im Unterricht benutzen wir den pythoneigenen Editor *Idle*. Du findest ihn, indem du im Menü 'Informatik' *Idle* direkt startest. *Idle* ist mit dem Pythoninterpreter direkt gekoppelt. Du musst nur oben in der Menüzeile unter *File* den Punkt *New File* starten, und schon öffnet sich eine neue Datei. Die Benutzung kennst du schon von anderen Programmen, z.B. Haskell. Hinweis: Man kann auch direkt auf der Konsole die Pythonbefehle eingeben.

```
macuser@MacBook:~/Documents/Schule/2015/Informatik/WPU 9/Python\>python3
Python 3.5.0b2 (v3.5.0b2:7a088af5615b, May 31 2015, 01:00:01)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

So kannst du direkt $\gg 21 + 34$ eingeben und das Ergebnis berechnen lassen. Python arbeitet dann praktisch wie ein programmierbarer Taschenrechner. Das funktioniert aber nur gut bei kleineren Programme. Daher arbeiten wir hier gleich "wie die Profis". Um die Eigenschaften von Python kennenzulernen, ist es am besten Beispielsprogramme zu analysieren. Hier ein Beispiel:

Listing 1: Ein erstes Beispiel: Spaghetti.py

```
1  from turtle import *
2  import random
3
4  reset()
5
6  speed("fastest")
7  tracer(False)
8  pensize(5)
9
10 def arm():
```

```
11      up()
12      goto(0,0)
13      down()
14      for i in range(1,11):
15          l = random.randint(20,50)
16          w = random.randint(-45,45)
17          #write(l)
18          forward(l)
19          left(w)
20
21          #print "i: ", i , " l:", l, " w:", w
22
23  for i in range(1,100):
24      arm()
25
26  tracer(True)
27  done()
```

Das Schöne an Python ist, dass man den Quelltext so lesen kann, wie ein Buch. Die Sprache ist fast immer selbsterklärend.
In Zeile 1 und 2 werden Module geladen. Das geschieht auf zwei unterschiedliche Weisen. Bei der Variante in Zeile zwei werden alle Funktionen dieses Modules geladen. Aufgerufen werden sie dann immer mit dem Modulnamen, gefolgt von einem Punkt und dann dem gewünschte Funktionsnamen. In Zeile 15 und 16 ist das dargestellt für die Funktion *randint*.
In Zeilen 6 bis 8 erfolgen ein paar Funktionsaufrufe. Was die genau machen kannst du in der Referenz für die einzelnen Module (hier Modul turtle) nachlesen. Du merkst, Spezifikationen und Signaturen sind, wie bei Haskell, von enormer Bedeutung.
Ab Zeile 10 erfolgt eine Definition einer Funktion. Dabei wird eine Zählschleife verwendet. Zeile 14 ist so zu lesen, dass die Variable i im Intervall $1 \leq i < 11$ jeden ganzzahligen Wert annimmt, d.h. die Schleife genau 10 mal durchlaufen wird. Die Einrückungen sind wichtig.Sie zeigen, welche Anweisungen zur Funktion gehören (alles, was unterhalb def arm() eingerückt ist) und was wiederum zur Zählschleife gehört (alles was unterhalb von for eingerückt ist). Da die zweite Zählschleife ab Zeile 25 nicht eingerückt ist, so weiß man, dass diese Schleife nicht zur Funktion arm() gehört, sondern zum Hauptprogramm. Verändere selbständig einige Parameter der Funktionen (die Werte innerhalb der Klammern) und beobachte was geschieht.

1.2 Schleifen und besondere Anweisungen

Eine Hauptaufgabe der Informatik ist Daten effizient zu verwalten und Algorithmen mit ihnen zu realisieren. Deshalb besitzt auch Python Datentypen. Python kennt

- Ganzzahl (Integer) z.B. 4321
- lange Ganzzahl. Sie können beliebig lang werden. Sie werden mit einem l am Anfang bzw. L am Ende bezeichnet.

- Fließkommazahlen, Zahlen der Form 3.14159 oder 17.3e+02
- komplexe Zahlen, z.B. 1.2+3j

Dann gibt es noch die wichtige Zeichenketten (Strings), z.B. "Python". Strings sind sogenannte Arrays und bestehen aus hintereinander geschriebene Zeichen (Char). Python liefert eine Menge von Funktionen für die Benutzung von Strings. Strings schauen wir uns mal extra an.
Aufgabe: Schreibe ein Programm mit Python, welches zwei Zahlen von der Konsole einliest und dann die Summe ausgibt.
Eigentlich würde das schon reichen, denn Haskell hat auch nicht mehr zur Verfügung gestellt. Allerdings gibt es da noch was , was die Möglichkeit schafft, mehrere Anweisungen in einer Folge auszuführen. Das sind die Schleifen.

Listing 2: While-Schleifen

```
1  a = 2
2  while a < 500:
3    print a
4    a *= a
5
6  #Beispiel
7
8  def frage():
9    ans = raw_input("Wann war der erste Mensch \ auf dem Mond? (YYYY-mm-dd) ")
10   return ans == '1969-07-20'
11
12 again = True
13 while again:
14   result = frage()
15   print result
16   if not result:
17     ans = raw_input("Weiterer Versuch? (J,n) ")
18     if (ans.lower() == "n"):
19       again = False
20     else:
21       again = False
```

Aufgabe: Neben den While-Schleifen gibt es noch die Zählschleifen. Finde heraus, wie diese funktionieren und stelle ein Beispiel vor.