Übungsblatt 1

Willkommen zum Praktikum zu Programmieren in Python.

Aufgabe 1. Starten Sie Python mit der interaktiven Konsole `ipython3`. Experimentieren Sie mit einigen Vorlesungsbeispielen. Evaluieren Sie die folgenden Ausdrücke und machen Sie sich klar, was warum passiert.

```
2 / 5
2 // 5
3 ** 4
int("42")
float("42")
"Hallo"
print("Hallo" + " Welt")
(i, j) = (1, 2)
i
i,j
i,j = 2,3
i,j
True/2
range(10)
list(range(1,1000,100))
s = "ham"
"eggs " + 2
"ham " "and " "eggs"
s * 5
s[:0]
s[0][0][0]
("x",)[0]
"eggs"[2]
("x", "y")[1]
(1,2) + (3+4, 5)
(1,2) + (3)
(1,2) + (3,)
```

```
lis = [1,2,3] + [4,5,6]
lis[:]
lis[:0]
lis[-2]
lis[-2:]
([1,2,3] + [4,5,6])[2:4]
(lis[2],lis[3])
len(lis)
m = list("hallo"); m
m.reverse(); m
m.sort(); m
m.index("o")
{"a":1, "b":2}["b"]
d = {"x":1, "y":2, "z":3}
d["w"] = 0
d["x"] + d["w"]
d[(1,2,3)] = 4
d
d.keys()
list(d.keys())
list(d.values())
(1,2,3) in d
4 in range(10)
"dio" in "He's an idiot, but he's ok"
0 or "" or [] or () or {} or None or "Ende"
x, y = 1, 2
x, y = y, x
```

Aufgabe 2. Schreiben Sie ein einfaches Python-Programm mit `code`, das einen Taschenrechner von der Kommandozeile simuliert, der einfache arithmetische Ausdrücke berechnet. Das Programm erwartet als Kommandozeilenargumente drei Argumente, eine Gleitkommazahl gefolgt von einem Operator aus `{ +, -, x, /}` und wieder gefolgt von einer Gleitkommazahl. Ein Aufruf soll zum Beispiel so verlaufen

```
$ python3 calc.py 2.3 + 4.5
2.3 + 3.4 = 6.8
```

```
$ python3 calc.py 3 '*' 4
  3 * 4 = 12
```

Konvertieren Sie Strings in Gleitkommazahlen mit `float`. Geben Sie eine Meldung aus, wenn die Eingaben nicht passend sind. Trennen Sie die Berechnung von der Ein-/Ausgabe.

Testen Sie die Berechnung mit dem Unittest `test_calc`. Beachten Sie, dass dazu Ihre Datei `calc.py` heißen muss und eine Funktion `def calc(v1, op, v2):` existieren sollte. Die Funktion erwartet die drei Strings aus der Eingabe und gibt das Ergebnis zurück, falls die Parameter okay sind.

Hinweis 1. Nutzen Sie folgendes Schema für die Aufgaben.

```
#!/usr/bin/python3
import sys

# Hier weitere Funktionen

def main(args): # args sind die Kommandozeilen-Argumente
    print("Hier Ihr Code")

if __name__ == '__main__':
    main(sys.argv[1:])
```

Wir nutzen zunächst intuitiv Module und Funktionen und erläutern Details später.

Aufgabe 3. Simulieren Sie die `append`-Methode von Listen mit Hilfe von Slice-Operationen und Zuweisungen. Geben Sie für den Aufruf `lis.append(ele)` eine äquivalente Anweisung an, wobei `lis` eine Liste ist und `ele` ein beliebiges Objekt. Achten Sie darauf, dass `append` eine *destruktive* Methode ist, d.h. die Methode verändert das ursprüngliche Objekt. Kann man dies statt für Listen auch für Tupel oder Strings tun?

Aufgabe 4. Was passiert bei den beiden folgenden Anweisungsfolgen? Zeichnen Sie jeweils die Struktur von `lis`. Was ist der Unterschied?

```
lis = [1, 2]
lis.append(lis)
```

```
lis = [1, 2]
lis = lis + lis
```

Aufgabe 5. Arbeiten Sie im Python-Tutorial den Abschnitt 3 *"An Informal Introduction to Python"* durch.

Aufgabe 6. Schreiben Sie ein Programm `heron`, um die Wurzel einer Gleitkommazahl anzunähern mit der Methode von Heron. Die Methode berechnet iterativ eine numerische Sequenz a_i, die $\sqrt{x}$ annähert mit

$$a_0 = \frac{1+x}{2} \qquad\qquad a_{i+1} = \frac{a_i + \frac{x}{a_i}}{2}$$

Wir berechnen diese Sequenz bis zu einem Ergebniswert a_{i+1}. Wir stoppen, sobald sich a_i und a_{i+1} nur noch um einen Wert kleiner als ein gegebenes ε unterscheiden.

Wir übergeben die Werte von x und ε über die Kommandozeile in dieser Reihenfolge. Falls ε nicht angegeben ist verwenden Sie 10^{-6}, Vorgabe für x ist 10.0. Lassen Sie das Programm in `code` laufen über Default-Werte der Kommandozeilenparameter. Verwenden Sie den Debugger, um schrittweise die Berechnung der a_i zu verfolgen. Starten Sie das Programm auch auf der Konsole.

Hinweis 2. Checken Sie Ihre Programme, auch Zwischenstände (wenn eine Entwicklungsphase abgeschlossen ist [20-90 min]), immer in Ihren Git-Account ein. So können Sie einfach und überall daran arbeiten. Da Sie alleine arbeiten bleiben Sie immer im Branch `master`.

moodle Python_Barth