Übungsblatt 6

Willkommen zum Praktikum zu Programmieren 3.

Aufgabe 1. Schreiben Sie folgende **for**-Schleife als **while**-Schleife um.

```
1  l = [1,2,3]
2  for e in l:
3    print e
```

Schreiben Sie folgende **while**-Schleife als **for**-Schleife um.

```
1  d = {1:"eins", 2:"zwei", 3:"drei"}
2  l = d.keys()
3  i = 0
4  while i < len(l):
5    s = l[i]
6    print s, d[s]
7    i = i+1
```

Schreiben Sie die Programme so kompakt wie möglich.

Aufgabe 2. Schreiben Sie eine rekursive Funktionen `ggTr`, die den größten gemeinsamen Teiler von zwei Zahlen berechnet. Erinnern Sie sich daran, dass

$$\forall x, y : \texttt{ggT}(x, y) = \texttt{ggT}(y, x) \wedge \forall x > y : \texttt{ggT}(x, y) = \texttt{ggT}(x - y, y)$$

Testen Sie Ihre die Funktionen mit einigen Beispielen wie `(10,30)`, `(20,30)`, `(2,5)`, `(8,6)`, `(7,3)`. Die naive rekursive Version ist leider nicht sehr effizient. Erinnern Sie sich, dass auch gilt

$$\forall x > y, x\%y! = 0 : \texttt{ggT}(x, y) = \texttt{ggT}(x\%y, y) \tag{1}$$

Schreiben Sie eine iterative Funktion `ggT`, die den größten gemeinsamen Teiler von zwei Zahlen mit Hilfe von (1) berechnet. Lesen Sie die Zahlen aus der Datei `ggts.dat` ein. Jede Zeile repräsentiert eine Zahl. Berechnen Sie den `ggT` aller Zahlen der Zeilen i und $i + 1$ für gerade $i \geq 0$. Was ist der Mittelwert aller `ggT`s?

Schreiben Sie darauf aufbauend eine Funktion `ggTl`, die den größten gemeinsamen Teiler einer Liste von Zahlen berechnet. Beachten Sie, dass der `ggT` dreier Zahlen `x,y` und `z` gleich dem `ggT` von dem `ggT` zweier Zahlen (zum Beispiel `ggT(x,y)`) und der dritten Zahl (zum Beispiel `ggT(ggT(x,y),z)`) ist. Was ist der `ggT` von 10, 80, 20 und 75? Definieren Sie die Funktion `ggTl` unter Verwendung von `ggT` noch einmal, aber verwenden Sie weder Rekursion noch **while** noch **for**.

Aufgabe 3. Abgleich von Argumenten. Definieren Sie die folgenden Funktionen und werten Sie die folgenden Ausdrücke aus. Erklären Sie das jeweilige Ergebnis anhand des Argumentenabgleichs wie in der Vorlesung vorgestellt.

```
def f(a,b,c=1):
  print a,b,c
def g(a,b,*c,**d):
  print a,b,c,d
def h(a,b,c=1,*d,**e):
  print a,b,c,d,e
```

```
f(1,2,3), f(1), f(1,2), f(1,2,3,4)
g(1,2), g(1,2,3,4), g(1,2,3,4,bla="bla")
h(1,2,3,4,5,6,c=7), h(1,2,3,4,5,6,x=7)
```

Aufgabe 4. Unter Linux können Sie sich mit dem Befehl `fortune` ein zufällig gewähltes Zitat ausgeben lassen Schreiben Sie ein Skript, das `fortune` nachbildet. Ohne Kommandozeilenparameter soll ein beliebiges Zitat ausgegeben werden. Mit der Option `-m <pat>` soll ein beliebiges Zitat ausgegeben werden, das `<pat>` enthält.

Die Zitate liegen unter `/usr/share/games/fortunes` in den Dateien, die weder die Endung `.dat` noch `.u8` haben. Die Zitate sind durch `\n%\n` voneinander separiert. Die Bibliotheken `os`, `random`, `sys`, `string` sind hilfreich.

Aufgabe 5. Schreiben Sie List-Comprehension-Ausdrücke zur Berechnung von: Geradzahlige Kubikzahlen der Zahlen 1 bis 10, Alle Teiler einer Zahl `z` außer 1 und `z` (testen Sie mit 123, 12345, 123456), Alle Primzahlen zwischen 10000 und 10100. Schreiben Sie die obigen Ausdrücke unter Verwendung der funktionalen Primitiven ohne List-Comprehension.

Aufgabe 6. Ramanujan – ein indischer Mathematiker – wurde einmal von Hardy – einem englischen Mathematiker – besucht. Hardy erwähnte, dass er ein Taxi mit der Nummer 1729 genommen hatte. Ramanujan antwortete, dass die 1729 bemerkenswert sei, da es die kleinste Zahl ist, die in zwei verschiedenen Arten als Summe von zwei Kubikzahlen dargestellt werden kann.

$$1729 = 1^3 + 12^3 = 9^3 + 10^3$$

Schreiben Sie einen einzigen Python-Ausdruck der Ihnen hilft diese Aussage nachzuvollziehen. Experimentieren Sie schrittweise mit List-Comprehension.

Aufgabe 7. Die Funktion `zip` gibt eine Liste von Tupeln zurück, so daß das i.te Tupel das i.te Element von jedem Sequenz-Argument enthält. Für Details lesen Sie die Dokumentation. Schreiben Sie eine Funktion `unzip`, die die Operation umdreht. Beispiel:

```
>>> unzip(zip((1,2,3,4), (5,6), (7,8))) == [(1,2), (5,6), (7,8)]
True
>>> t = [(1,2), (3,4), (5,6)]
>>> unzip(unzip(t)) == t
True
```

http://www.mi.hs-rm.de/~barth/hsrm/prog3