Übungsblatt 4

Willkommen zum Praktikum zu Programmieren in Python.

Aufgabe 1. 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 nochmals, aber nur unter Verwendung der funktionalen Primitiven (`map, filter, reduce`) ohne List-Comprehension.

Aufgabe 2. Ramanujan wurde von einmal Hardy besucht, der 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 Python-Ausdruck, der Ihnen hilft diese Aussage nachzuvollziehen. Experimentieren Sie schrittweise mit List-Comprehension. Suchen Sie in allen Kombinationen von Zahlen bis zu einer Grenze nach vier paarweise verschiedenen Zahlen, von denen die Summe der Kubikzahlen von je zwei gleich ist. Erhöhen Sie die Grenze (interaktiv) immer weiter, bis die Liste der 4-Tupel nicht leer ist, geben Sie davon das Minimum aus.

Aufgabe 3. 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:

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

Aufgabe 4. Schreiben Sie ein Programm `rot13`, das von Standardeingabe einen Text liest und diesen ROT13-verschlüsselt auf der Standardausgabe ausgibt. ROT13[1] (rotiere um 13 Stellen) ist ein einfacher Verschlüsselungsalgorithmus bei dem alle Zeichen im Bereich `'A'` bis `'Z'` und im Bereich `'a'` bis `'z'` um jeweils 13 Zeichen verschoben werden. Zum Beispiel wird aus `a` das Zeichen `n` und aus `X` wird `K`. Aus dem Text `Python ist toll!` wird `Clguba vfg gbyy!`.

Testen Sie Ihr Programm indem Sie den Inhalt einer Datei als Eingabe des Programms verwenden und das Ergebnis in eine andere Datei speichern. Beachten Sie, dass die zweimalige Anwendung von ROT13 wieder den ursprünglichen Text ergeben sollte. Mit dem Unix-Tool `diff` können Sie Unterschiede zwischen zwei Texten feststellen.

```
$ cat datei.txt | ./rot13.py | ./rot13.py > datei.orig
```

Sie sollten in `rot13` eine Funktion `rot13` haben, die einen String erwartet und davon den ROT13-String zurückgibt. Testen Sie Ihre Funktion mit dem Unittest `test_rot13`. Testen Sie Ihr Programm auch mit der Datei `/usr/share/dict/words`.

[1] EBG13 jheqr vz Hfrarg va qra seüura 80re Wnuera rvatrfrgmg. Mvry jne avpug qvr Irefpuyüffryhat, fbaqrea qnff zna avpug hatrjbyyg riraghryy "bssrafvir" (oryrvqvtraqr?) Grkgr yrfra zhffgr. Rf jne rva Gnfgraqehpx abgjraqvt qvrfr mh yrfra.

Aufgabe 5. Das Soundex-Verfahren wurde in den USA verwendet, um Volkszählungsdaten nach ähnlich klingenden Nachnamen (gleicher Soundex-Wert) zu gruppieren und so schreibfehlertolerant nach englischen Wörtern zu suchen. Der Soundex-Wert eines Wortes berechnet sich wie in folgender Tabelle dargestellt, dabei ist der erste Buchstabe des Soundex-Werts der erste Buchstabe des Worts.

Buchstabe	BFPV	CGJKQSXZ	DT	L	MN	R	AEIOUWYH
Ziffer	1	2	3	4	5	6	-

Jeder nachfolgende Buchstabe wird (unabhängig von Groß- oder Kleinschreibung) schrittweise anhand voriger Tabelle in eine Ziffer umgewandelt oder ignoriert. Falls die Soundex-Ziffer gleich der vorhergehenden Soundex-Ziffer ist, dann wird sie ignoriert. Der Soundex-Wert besteht aus maximal 6 Zeichen, weitere Zeichen werden ignoriert. Wenn der Soundex-Wert weniger als 6 Zeichen hat, dann wird der Soundex-Wert mit der Ziffer 0 aufgefüllt.

Schreiben Sie Python-Programm `soundex`, das beliebig viele Worte als Kommandozeilenparameter entgegen nimmt und den Soundex-Wert jedes Wortes berechnet. Beispiel:

```
$ ./soundex.py soundex soundeggs flurbel
  1              soundex : s53200
  2            soundeggs : s53200
  3              flurbel : f46140
```

Sie sollten in Ihrer Modul `soundex` eine Funktion `soundex` haben, die einen String erwartet und davon den Soundex-Wert als String zurückgibt. Testen Sie Ihre Funktion mit dem Unittest `test_soundex.py`. Schreiben Sie Ihre Funktion so, dass Sie mit wenig Aufwand den Soundex-Wert eines Worts berechnen. Bereiten Sie dazu die Soundex-Tabelle mit Dictionaries passend auf. Nutzen Sie, wenn möglich, Dictionary-Expressions, die es mit geschweiften statt eckigen Klammern erlaubt ein Dictionary zu erzeugen, zum Beispiel:

```
>>> {i:i**2 for i in range(1, 5)}
{1: 1, 2: 4, 3: 9, 4: 16}
```

Schreiben Sie ein Programm `similar`, das ein Wort und einen (optionalen) Dateinamen erwartet. Aus der Datei werden alle Worte eingelesen und die ausgegeben, die nach dem Soundex-Verfahren ähnlich zu dem ersten Argument sind. Testen Sie das Programm mit der Datei `/usr/share/dict/words`, was auch der Vorgabewert ist. Berücksichtigen Sie nur Worte, die ausschließlich aus ASCII-Buchstaben bestehen. Testen Sie mit den Worten: `hello, Python, Guido, Obstsalat, flurbel`

Schreiben Sie ein Program `checksimilar` was nur einen (optionalen) Dateinamen erwartet und darin den Soundex einer größten gleich klingenden Gruppe and Worten bestimmt und ihn ausgibt. Zusätzlich sollen die Anzahl der Worte und die Anzahl der verschieden klingenden Worte ausgegeben werden. Testen Sie das Programm mit der Datei `/usr/share/dict/words`.

moodle Python_Barth