## Übungen zu Computergrundlagen WS 2010/2011

# Übungsblatt 6

23. November 2010

### Allgemeine Hinweise

Die Lösungen solltest Du in eine Kopie der Datei `/share/Courses/CG2010/blatt6/blatt6.txt` einfügen.

Abgabetermin für die Lösungen ist

- **Montag, 29.11., 13:00** für die Übungsgruppen am Mittwoch und Donnerstag
- **Donnerstag, 2.12., 13:00** für die Übungsgruppen am Montag und Dienstag

Zur Abgabe kannst Du **entweder** den Befehl `/share/Courses/CG2010/bin/abgabe <datei>` ausführen (dabei sollte `<datei>` die Lösungsdatei bezeichnen), **oder** Du schickst die Datei per Email an den jeweiligen Tutor.

### Aufgabe 6.1: Funktionen, Rekursionen, Schleifen und Listen (5 Punkte)

Die Fibonacci-Zahlen sind wie folgt definiert:

$$\text{fib}(n) = \begin{cases} 1 & \text{falls } n \leq 1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{sonst} \end{cases} \tag{1}$$

- 6.1.1 (1 Punkt) Schreibe eine Python-Funktion `fib1(n)`, die die $n$-te Fibonacci-Zahl wie in Gleichung 1 berechnet. Füge die Funktion in die Lösungsdatei ein.

  Berechne dann mit Hilfe der Funktion die Fibonacci-Zahlen für $n = 5, 10, 15, 20, 25, 30, 35, 40$. Miss die Laufzeit der einzelnen Berechungen und füge sie in die Lösungsdatei ein. **Hinweis:** Der Unixbefehl `time` kann zur Laufzeitmessung wie folgt verwendet werden:

```
> time python fibtest.py
121393

real    0m0.080s
user    0m0.070s
sys     0m0.000s
```

- 6.1.2 (1 Punkt) Eine äquivalente Definition der Fibonacci-Zahlen sieht so aus:

$$\text{fib}_{a,b}(n) = \begin{cases} a & \text{falls } n = 0 \\ b & \text{falls } n = 1 \\ \text{fib}_{b,(a+b)}(n-1) & \text{sonst} \end{cases} \tag{2}$$

$$\text{fib}(n) = \text{fib}_{1,1}(n) \tag{3}$$

  Schreibe eine Python-Funktion `fib2`, die die Fibonacci-Zahlen wie in Gleichung 3 (also rekursiv) berechnet. Füge die Funktion in die Lösungsdatei ein.

  Miss auch für diese Funktion die Laufzeit der Berechnung von Fibonacci-Zahlen wie in Aufgabe 6.1.1. Vergleiche die Laufzeiten. Wieso sind sie so unterschiedlich?

- 6.1.3 (1,5 Punkte) Schreibe eine Funktion `fib3`, die die Fibonacci-Zahlen mit Hilfe einer Schleife ohne den rekursiven Aufruf einer Funktion berechnet und füge sie in die Lösungsdatei ein. Miss auch für diese Funktion das Laufzeitverhalten.

- 6.1.4 (1,5 Punkte) Schreibe eine Funktion `fiblist`, die eine Liste der ersten $n$ Fibonacci-Zahlen berechnet. Diese soll natürlich *nicht* einfach die obigen Funktionen für jedes Listenelement erneut berechnen, sondern stattdessen die Liste in einem einzigen Durchgang entweder rekursiv oder in einer Schleife erzeugen.

## Aufgabe 6.2: Dictionaries, Strings und Module (5 Punkte)

- 6.2.1 (4 Punkte) Schreibe ein Skript `occurence.py`, das folgenden Spezifikationen folgt. Füge das Skript in die Lösungsdatei ein.

  Die Aufgabe des Skriptes ist es, einen Text von der Standardeingabe zu lesen und die Häufigkeit des Auftretens der verschiedenen Wörter zu zählen. Die Ausgabe soll dabei so formatiert sein, dass in jeder Zeile der Ausgabe die Häufigkeit eines Wortes auf 5 Zeichen Breite rechtsbündig formatiert werden soll, dann ein Leerzeichen folgt, und dann das Wort ausgegeben wird. **Hinweis:** Zum Testen des Skriptes kannst Du die Datei `/share/Courses/CG2010/blatt6/gpl-3.0.txt` wie folgt verwenden:

  ```
  > python occurence.py < gpl-3.0.txt
     11: free
     11: OR
     11: versions
      .
      .
  ```

  **Hinweis:** Benutze die Funktionen `sys.stdin.readlines` und `string.split`.

  Darüberhinaus soll das Skript die folgenden Eigenschaften haben:

  - Das Skript soll die Option `-m` bzw. `--min` verstehen. Der Wert dieser Option soll festlegen, wie häufig ein Wort in der Eingabe mindestens auftreten muss, damit seine Häufigkeit ausgegeben wird. **Beispiel:**

    ```
    >  python occurence.py --min=100 < gpl-3.0.txt
      102 you
      131 or
      165 a
      174 to
      208 of
      309 the
    ```

  - Die Ausgabe soll nach den Häufigkeiten sortiert sein, so dass die am häufigsten auftretenden Wörter am Ende ausgegeben werden. **Hinweis:** Verwende die Funktion `sorted`. Wie man die Funktion genau verwendet solltest Du im Internet nachlesen. Die Funktion `help` hilft in diesem Fall leider nicht sehr.

- 6.2.2 (1 Punkt) Verwende das Skript im Zusammenhang mit passenden Unixbefehlen, um festzustellen, welches die fünf häufigst auftauchenden Dateiname im Verzeichnis `/usr` und dessen Unterverzeichnissen sind. Wichtig dabei ist nur der Dateiname selbst, nicht die Namen der Unterverzeichnisse. Füge die Dateinamen und den dafür notwendigen Befehl in die Lösungsdatei ein.