5. Aufgabenblatt zur Vorlesung

Konzepte der Programmierung

WiSe 2024/25

Wolfgang Mulzer

Abgabe am Freitag, den 22. November 2024, 12 Uhr

Aufgabe 1 Blocksatz *10 Punkte*

Blocksatz ist eine Art, um Texte auf dem Papier auszurichten. Dabei wird das erste Wort einer Zeile am linken Rand ausgerichtet, und das letzte Wort am rechten Rand. Die restlichen Wörter werden gleichmäßig auf der Zeile verteilt.

(a) Schreiben Sie eine Python-Funktion `blocksatz`, die eine Liste von Wörtern erhält und diese im Blocksatz setzt. Ihre Funktion soll eine Liste von Zeilen liefern, die jeweils im Blocksatz sind. Gehen Sie dabei davon aus, dass eine Zeile aus 80 Zeichen besitzt. Wenn ein Wort mehr als 80 Zeichen hat, so wird es abgeschnitten.

(b) Erweitern Sie ihre Funktion so, dass die Anzahl der Zeichen in einer Zeile als Parameter übergeben werden kann.

(c) Informieren Sie sich über die Python-Funktionen zum Lesen von Dateien. Erweitern Sie Ihr Programm so, dass die Liste von Wörtern aus einer Textdatei eingelesen werden kann.

Aufgabe 2 Die Fibonacci-Zahlen *10 Punkte*

Die Fibonacci-Zahlen sind wie folgt definiert:

$$F_0 = 1; F_1 = 1; \text{ und } F_n = F_{n-1} + F_{n-2}, \text{ für } n \geq 2.$$

(a) Implementieren Sie die rekursive Definition der Fibonacci-Zahlen direkt in Python.

(b) Wie lange braucht Ihre Funktion aus (a), um F_{35} zu berechnen? Visualisieren Sie die Rekursion in einem Diagramm und erklären Sie, was die Ineffizienz verursacht.

(c) Modifizieren Sie Ihr Programm aus (a) so, dass bekannte Funktionsergebnisse in einem Wörterbuch gespeichert werden. Wie lange braucht es jetzt, um F_{35} zu berechnen?

(d) Schreiben Sie ein Programm, das F_n iterativ mit Hilfe einer Liste berechnet. Wie schnell können Sie F_{35} jetzt berechnen?

(e) Modifizieren Sie Ihr Programm aus (d) so, dass es möglichst wenig Speicher verwendet.

Aufgabe 3 Parameterübergabe *10 Punkte*

In der Vorlesung wurde erwähnt, dass es beim Aufruf von Funktionen unterschiedliche Möglichkeiten geben kann, wie sich formale und tatsächliche Parameter zueinander verhalten.

(a) Erklären Sie die Parameterübergabekonventionen *call by reference*, *call by value* und *call by name* jeweils kurz in zwei Sätzen.

(b) Betrachten Sie das folgende Codefragment:

```
def f(x, y)
    print("f: x = " + str(x) + ", y = " + str(y) + ".")
    x = x + y
    y = 2 * x
    print("f: x = " + str(x) + ", y = " + str(y) + ".")

a = 2
b = 4
print("main: a = " + str(a) + ", b = " + str(b) + ".")
f(a, b)
print("main: a = " + str(a) + ", b = " + str(b) + ".")
```

Was gibt das Programm aus, wenn `f` *call by value* verwendet? Was passiert bei *call by reference*?

(c) Betrachten Sie das folgende Codefragment:

```
def f():
  print("Hier ist f.")
  return 5

def g(a):
  x = a
  y = 2 * a
  return x + y

z = g(f())
print("main: z = " + str(z) + ".")
```

Was gibt das Programm aus, wenn `g` *call by value* verwendet? Was passiert bei *call by name*?