Einführung in die Programmierung

Prof. Dr. Franziska Matthäus
Prof. Dr. Matthias Kaschube
Dr. Karsten Tolle
Lukas Müller

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt ÜE-06

Ausgabe: 08.12.2021
Abgabe: 18.12.2021
16:00 Uhr

Rekursion

Hinweis:

- *Es sind grundsätzlich Rechenwege anzugeben, es sei denn es findet sich ein expliziter Hinweis, dass dies nicht nötig ist.*
- *Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.*
- *Digitale Abgaben, die nicht im Format **.pdf** oder **.txt** für Texte oder **.py** für Code erfolgen, werden nicht bewertet. Bei Abgaben mehrerer Dateien müssen diese als **.zip** zusammengefasst werden.*
- *Achten Sie darauf die Variable **__author__** in allen Quellcode Dateien korrekt zu setzen (am Anfang des Quellcodes):*
  __author__ = "<Matr-Nr>, <Nachname>"
- *Außerdem muss Ihr Name in jeder abgegebenen **.pdf** und **.txt** Datei zu finden sein.*
- *Abgaben, die per Hand geschrieben und eingescannt werden, sind **nicht** erlaubt (bzw. geben 0 Punkte und werden nicht korrigiert).*
- *Beim Programmieren und Kommentieren halten Sie sich die Regeln im Programmierhandbuch, siehe Moodle-Kurs (Programmierhandbuch WiSe 21/22 (Style Guide)). Im Zweifelsfall gilt PEP 8.*

∑ 8 (+1) Punkte

Aufgabe 1 – Rekursion — 2 Punkte

Unten sind zwei rekursive Funktionen abgebildet. Geben Sie die Ausgabe an, wenn diese Funktionen mit dem Wert 5 (also recursion_1(5) und recursion_2(5)) aufgerufen wird und **erklären** sie kurz, wie es zu dieser Ausgabe kommt.

```python
def recursion_1(n):
    if n == 0:
        return
    recursion_1 (n-1)
    print(n)
```

```python
def recursion_2(n):
    print(n)
    if n == 0:
        return
    recursion_2(n-1)
```

Aufgabe 2 – Endrekursion — 1 Punkte

Gegeben sei die unten dargestellte rekursive Funktion sum_over_n(n). Schreiben Sie die Funktion in eine endrekursive Funktion um, welche das identische Ergebnisse liefert.

```python
def sum_over_n(n):
    if n == 0:
        return 0
    else:
        return n + sum_over_n(n-1)
```

Aufgabe 3 – Der beste Weg 5 (+1) Punkte

Aufgabe ist für eine Matrix ($n \times m$) mit Zahlen einen Weg von Position (0, 0) nach (n-1, m-1) zu finden, wobei die Summe der Elemente auf dem Weg möglichst gering ist.

Randbedingungen:

- Jede Position darf nur einmal auf dem Weg liegen.
- Default: Man darf nur horizontal und vertikal gehen. Es werden nur positive Zahlen (>= 0) verwendet.
- Wer auch einbezieht Diagonal gehen zu können oder auch negative Zahlen zulässt, kann je einen halben Zusatzpunkt erhalten. Beides zusammen einen Punkt … also 6 statt 5 Punkte auf diese Aufgabe und somit max. 9 Punkte auf das Blatt.

Hier ein kleines Beispiel.

```
[[4 0 8]
 [3 4 7]
 [8 0 7]]
```

Der Start ist oben links bei (0, 0). Der erste Wert ist also 4. Geht man rechts kommt die 0 dazu, zweimal nach unten (+4 und +0) und dann rechts zum Ziel. Die Summe dieses Weges ist also: 4 + 0 + 4 + 0 + 7 = 15.
Der Weg selbst ist: (0,0), (0,1), (1,1), (2,1), (2,2).

Schreiben Sie eine Funktion, welche als Eingabe die Dimension der Matrix ($n \times m$) sowie einen Wertebereich (*min, max*) für mögliche Einträge in der Matrix angibt. Als Rückgabe eine entsprechende Matrix mit zufälligen Werten aus dem Wertebereich zurückgibt (*return*).

Schreiben Sie eine Funktion, welche als Parameter eine Matrix übergeben bekommt und einen entsprechenden optimalen Weg zurückgibt (*return*). Weiterhin soll die Funktion die Summe der Elemente auf dem Weg in der Konsole ausgeben (*print*).

Beachten Sie, dass es auch Situationen geben kann, wo mehrere Wege die gleiche Summe ergeben können. Wie Sie damit umgehen, sollte in den Kommentaren oder *docstring* zur Funktion erläutert werden (eine ausführliche extra Dokumentation wird nicht verlangt)!

Erstellen Sie für beide Funktionen mindestens drei Testfälle als „*doctest*“ und sorgen Sie dafür, dass diese nur ausgeführt werden, wenn das Modul als Hauptmodul (*main*) aufgerufen wird.

Für alle fehlenden Vorgaben treffen Sie selbst entsprechende Annahmen und begründen Sie diese. Geben Sie weiterhin (im *docstring* der Funktionen) an:

- ob und wenn ja welche Algorithmen-Entwurfsmuster sie verwendet haben
- ob ihre Lösung immer das globale beste Ergebnis findet (oder vielleicht nur ein lokales Optimum)