Prof. Dr. Stefan Göller
Florian Bruse
Dr. Norbert Hundeshagen

Einführung in die Informatik

WS 2018/2019

Übungsblatt 3
8.11.2018-15.11.2018

Abgabe: Bis zum 15.11. 18:00 Uhr über moodle. Reichen Sie pro Aufgabe, die Sie bearbeitet haben, genau eine Textdatei mit dem Namen `aufgabe_i.py`, (wobei i die Aufgabennummer ist) ein, welche die Lösung ihrer Gruppe enthält.

Aufgabe 1 (Listen, Tupel) (5*5 Punkte):

In dieser Aufgabe geht es um eine kleine Datenbank aus Filmen. Sie erhalten diese Datenbank als die ebenfalls bereitgestellte Datei `hollywood.csv`, welche im Comma-separated-values-Format vorliegt. Das bedeutet, dass jede Zeile genau einen Datensatz enthält. In unserem Fall ist ein Datensatz der Titel eines Films, ein Schauspieler, der darin mitgespielt hat, und der Regisseur. Diese Daten werden durch Kommas getrennt. Eine denkbare Zeile wäre also `The Fugitive, Harrison Ford, Andrew Davis` und kodiert, dass Harrison Ford im Film "The Fugitive" mitgespielt hat, bei dem Andrew Davis die Regie inne hatte. Der Einfachkeit halber können sie annehmen, dass keiner der Datensätze selbst ein Komma enthält. Jede Zeile enthält also genau zwei Kommas.

a) Schreiben Sie eine Funktion `lies_zeile`, welche eine Zeichenkette als Parameter entgegen nimmt und überprüft, ob diese Zeichenkette genau zwei Kommas enthält. Falls ja, soll die Zeichenkette in ein passendes Tupel mit drei Einträgen konvertiert werden. Aus der Zeichenkette `"The Fugitive, Harrison Ford, Andrew Davis"` soll das Tupel

```
("The Fugitive", "Harrison Ford", "Andrew Davis")
```

werden. Solche Tupel werden wir im Folgenden als *Filmtupel* bezeichnen. Falls die Zeichenkette nicht genau zwei Kommas enthält, soll eine Fehlermeldung ausgegeben werden, nach welcher das Programm terminiert.

b) Schreiben Sie nun eine Funktion `lies_datei`, welche eine Dateihandle zu einer Datei wie `hollywood.csv` (oder eine andere Datei vom gleichen Format) entgegennimmt, diese Datei ausliest, und die entsprechende Liste von Filmtupeln zurückgibt. Die Liste soll die aus der Datei vorgegebene Sortierung haben.

c) Schreiben Sie eine Funktion `hat_schauspieler`, welche eine Liste von Filmtupeln, sowie eine Zeichenkette entgegennimmt. Ihre Funktion soll nun überprüfen, ob es in der Liste einen Eintrag gibt, in dem der Schauspieler mit der Zeichenkette übereinstimmt. Das Ergebnis soll als `bool` zurückgegeben werden.

d) Schreiben Sie eine Funktion `schauspieler_zusammenarbeit`, welche wie vorher eine Liste von Filmtupeln entgegennimmt, und zusätzlich eine Zeichenkette. Ihre Funktion soll jetzt eine Liste von Schauspielern zurückgeben, die in einem Film mitgespielt haben, bei dem der in der Zeichenkette übergebene Schauspieler auch mitgespielt hat.

e) Es fällt auf, dass die Datei `hollywood.csv` nach Regisseuren sortiert ist. Schreiben Sie eine Funktion `fuege_ein`, welche wie vorher eine Liste von Tupeln entgegeben nimmt, und zuätzlich ein einzelnes Tupel. Ihre Funktion soll nun eine Liste zurück geben, welche sich dadurch ergibt, dass das neue Tupel an passender Stelle, ebenfalls nach Regisseuren in die alte Liste eingefügt wurde.

Hinweis: Es kann mehrere richtige Positionen zum Einfügen geben.

Aufgabe 2 (von Listen von Listen von ...) (10+10 Punkte):

Wie in der Vorlesung besprochen, können Sie in Python Listen `L` durch den Befehl `L[:]` kopieren. Dies kopiert jedoch nur die äußere Liste `L`, kopiert jedoch nicht die Elemente von `L`, die selbst auch veränderlich sind: Folgendes Python-Programm

```
L=[1,3]
K=[L,2]
U=K[:]
print(U)
```

kopiert zwar die Liste `K` und liefert die Ausgabe

`[[1,3],2]`.

Falls Sie diesem Programm jedoch die folgenden beiden Zeilen

```
L.append(4)
```

print`(U)`

anhängen würden, erhielten Sie die Ausgabe

```
[[1,3],2]
[[1,3,4],2].
```

Die Unterliste `L` hat also einen Seiteneffekt (der Tiefe 2) erzeugt, da die Unterliste `[1,3]` in `K` selbst nicht kopiert wurde. Genauso können u.U. Seiteneffekte in Tiefe 3 auftreten, indem Sie eine Liste modifizieren, die in einer Liste vorkommt, die selbst wiederum in einer Liste vorkommt (und so weiter).

Eine `int`-*Superliste* ist rekursiv folgendermaßen definiert:

- Jede Liste, deren Elemente alle vom Typ `int` sind, ist eine `int`-Superliste.
- Jede Liste deren Elemente alle `int`-Superlisten sind, ist eine `int`-Superliste.

a) Schreiben Sie eine rekursive Funktion `intSuperliste` mit einem Parameter, die überprüft, ob es sich bei dem übergebenen Parameter um eine `int`-Superliste handelt, also `True` zurückgibt, falls es sich um eine `int`-Superliste handelt und sonst `False`.

b) Schreiben Sie eine rekursive Funktion `Kopie`, die eine `int`-Superliste als Parameter erwartet und eine Kopie der `int`-Superliste zurückgibt, bei der die oben beschriebenen Seiteneffekte ausbleiben, d.h. es gibt keine Seiteneffekte beliebiger Tiefe.

Achtung: Für diese Aufgabe dürfen Sie nur Python-Befehle verwenden, welche Sie in der Vorlesung kennen gelernt haben. Insbesondere ist die Verwendung des Befehls `deepcopy` nicht erlaubt!

Hinweis: Falls `s` eine Variable ist, können Sie mittels **type**`(s)==list` (bzw. **type**`(s)==int` testen, ob `s` tatsächlich vom Typ `list` (bzw. vom Typ `int`) ist.

Aufgabe 3 (Rekursive Funktionen) (15 Punkte):

Die Försterin vom Habichtswald möchte die Höhe des kleinsten Baums in ihrem Wald berechnen. Sie kennt bereits die Höhe des größten Baums und weiß, dass folgende Regeln für den Habichtswald gelten:

- Wenn ein Baum der Höhe x existiert, dann auch einer der Höhe $x - 34$.
- Wenn ein Baum der Höhe x existiert, dann auch einer der Höhe $\frac{x-11}{2}$.

Schreiben Sie ein Programm, welches die Höhe des größten Baums im Habichtswald, gegeben als positive `float`-Zahl, als Eingabe erwartet und die Höhe des kleinsten Baums ausgibt. Nutzen Sie dafür eine Funktion, die rekursiv die kleinste Höhe berechnet. Beachten Sie, dass natürlich **keine Bäume der Höhe** ≤ 0 existieren.

Hinweis: Beachten Sie, dass Sie immer die Resultate beider Regelanwendungen untersuchen müssen. Für den Eingabewert $34,5$ liefert eine Anwendung der ersten Regel einen Baum der Höhe $0,5$. Der kleinste Baum im Habichtswald hat für den Startwert $34,5$ aber die Höhe $0,375$: Da wir wissen, dass es einen Baum der Höhe $34,5$ gibt, ergibt sich mit der zweiten Regel, dass es auch einen Baum der Höhe $\frac{34,5-11}{2}$ gibt, also einen Baum der Höhe $11,75$. Die erste Regel ist auf diesen Baum natürlich nicht mehr anwendbar. Eine weitere Anwendung der zweiten Regel liefert dann den Baum der Höhe $\frac{11,75-11}{2} = 0,375$, auf den keine Regel mehr anwendbar ist.