Prof. Dr. Stefan Göller
Daniel Kernberger

Einführung in die Informatik

WS 2019/2020

Übungsblatt 3
8.11.2019 - 14.11.2019

Abgabe: Bis zum 14.11. 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. Beachten Sie die speziellen Dateinamen, die in Aufgabe 2 gefordert sind.

Aufgabe 1 (Erweiterte Berechnung der Kubikwurzel) (10 Punkte):

Betrachten Sie noch einmal die näherungsweise Berechnung der Kubikwurzel einer Zahl mittels binärer Suche (Folie 183). Ihre Aufgabe ist es nun, das Verfahren so zu erweitern, dass es für alle Zahlen vom Typ `float` funktioniert, also auch Zahlen kleiner als 1.

Schreiben Sie dazu eine Funktion, die eine `float`-Zahl w und eine Fehlertoleranz $e > 0$ (auch als `float`) entgegen nimmt. Ihre Funktion soll nun prüfen, ob die Eingaben wie gewünscht sind. Danach soll ihre Funktion geeignete Startwerte für die Berechnung der Kubikwurzel mittels binärer Suche ermitteln (einer der Startwerte ist immer 0). Dann soll ihre Funktion, wie in der Vorlesung vorgestellt, einen Wert k für die Kubikwurzel suchen so dass k^3 nicht weiter als e von w entfernt ist. Dieser Wert soll dann ausgegeben werden.

Aufgabe 2 (Funktionen, Dateien lesen/schreiben) (25=10+10+5 Punkte):

Es soll ein Programm geschrieben werden, überprüft, ob ein gegebenes Wort in einer Wortliste, gegeben durch eine Datei, enthalten ist. So etwas ist z.B. für Rechtschreibkorrektur nützlich. Als Basis dafür benutzen wir hier das unter der GPL-Lizenz verfügbare de_DE Hunspell-Wörterbuch, welches Sie in der Datei `de_DE_frami.dic` finden.

a) Schreiben Sie eine Funktion `find_pos` innerhalb eines Programms mit dem Dateinamen `"aufgabe_1_1.py"`, die eine Zeichenkette und einen Buchstaben als Parameter erwartet und die erste Position des Buchstaben in der Zeichenkette zurück gibt, falls dieser existiert und ansonsten `-1`.

Beispiel: Bei Übergabe der Zeichenkette `"Abbild/JRTSm"` und des Buchstabens `"/"` soll die Funktion die Zahl `6` zurückgeben.

b) Schauen Sie sich zunächst den Inhalt der Datei `de_DE_frami.dic` an. Die Datei enthält neben einführenden Kommentarzeilen und einer Leerzeile (also einer, Zeile die nur einen Zeilenumbruch enthält) über 250000 Wörter der deutschen Sprache, wobei jedes Wort in einer eigenen Zeile eingetragen ist. Ferner werden Sie erkennen, dass den meisten Wortenden noch eine **Zeichenkette** beginnend mit `"/"` **angehängt** ist.

Ihr Programm mit dem Dateinamen `"aufgabe_1_1.py"` soll nun folgendermaßen funktionieren. Erstellen Sie nun mit Hilfe der in Teilaufgabe a) beschriebenen Funktion aus `de_DE_frami.dic` eine neue Datei `dictionary.txt`, welche nur noch zeilenweise die in `de_DE_frami.dic` vorkommenden Wörter ohne angehängte Zeichenkette enthält (also auch nicht die Kommentare und Leerzeilen).

Hinweis: Folgende Befehle benötigen Sie um Dateien zu manipulieren:

`f = open('test.txt','r')`	Öffnet eine Datei mit dem Namen `test.txt` im Lesemodus. Auf die Datei kann dann mit Hilfe des Bezeichners `f` zugegriffen werden.
`f = open('test.txt','w')`	Öffnet eine Datei mit dem Namen `test.txt` im Schreibmodus. Sollte die Datei nicht im Verzeichnis des Programms existieren, wird diese angelegt. Existierende Dateien werden überschrieben.
`f.close()`	Schließt die Datei die unter dem Bezeichner `f` geöffnet wurde. Dateien die nicht mehr benötigt werden, sollten aus Gründen des Speichermanagements immer geschlossen werden.
`f.read()`	Gibt den kompletten Inhalt der Datei als Zeichenkette zurück.
`f.readline()`	Gibt die erste Zeile der Datei als Zeichenkette zurück. Ein weiterer Aufruf von `f.readline()` gibt dann die nächste Zeile der Datei als Zeichenkette zurück, usw.. Sie erkennen das Ende der Datei daran, dass `f.readline()` eine leere Zeichenkette (also auch ohne Zeilenumbruch) zurückliefert.
`f.write("abc")`	Hängt die Zeichenkette `"abc"` an die letzte Zeile der Datei an, die unter dem Bezeichner `f` im Schreibmodus geöffnet wurde.

c) Schreiben Sie nun ein Programm "`aufgabe_1_2.py`", welches als Eingabe ein beliebiges Wort erwartet und die entsprechende Zeilennummer in `dictionary.txt` ausgibt, falls dieses Wort in der Datei enthalten ist.

Aufgabe 3 (Listen, Tupel) (25=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.