Übung 3: Listen, For-Schleifen und Dictionaries

Programmiertechniken in der Computerlinguistik I, HS 15

15. Oktober 2015

Hinweise zur Abgabe

- Abgabeformat: Bitte gebt jedes Python-Programm in einer eigenen Datei ab, die die Dateiendung *.py hat und ausführbaren Python-Code enthält
- Textdateien gebt ihr bitte als PDF oder Plain-Text ab.
- Um das Hochladen der Abgabe auf OLAT zu erleichtern, bitte die Dateien mit zip oder tar (oder einem anderen verbreiteten Format) komprimieren.
- Verseht jede Datei, die ihr abgebt mit eurem Namen und eurer Matrikelnummer
- Dateiname im Format olatbenutzername_pcl1_uebungsnr, z.B. muellerh_pcl1_uebung03
- Bitte nummeriert die Aufgaben auf dem Abgabeblatt gleich wie auf dem Aufgabenblatt
- Lernpartnerschaften zu zweit sind erlaubt und erwünscht. Bitte gebt in diesem Fall nur eine Lösung ab und benennt die Dateien wie folgt: benutzername1_benutzername2_pcl1_uebungnr
- Gebt auf jeden Fall pünktlich ab! Der Übungsbaustein ist nur bis am 28. Oktober um 18:00 offen.

1 Listen

1.1 Listen bearbeiten und manipulieren

Nutze für diese Aufgabe die Python-Dokumentation zu Listen (Abschnitt 5.1 More on Lists und 5.6.4. Mutable Sequence Types) und das Lehrvideo.
Für diese Aufgabe steht ein Programm `Aufgabe1.py` bereit, welches die Liste `text` beinhaltet. Ergänze das gegebene Skript so, dass es (nacheinander)

a) am Ende der Liste ein Ausrufezeichen anhängt,

b) der Liste die Wörter des Satzes 'Dies ist mein Satz' hinzufügt

c) die Anzahl der Vorkommen des Tokens 'ist' ermittelt und mitteilt,

d) das erste Vorkommen von '&' aus der Liste entfernt und

e) vor dem dritten Element (achte auf Index!) das Wort 'so' einfügt,

f) das erste Element mit dem Wort 'Computerlinguistik' ersetzt (d.h. Anzahl Elemente bleibt gleich),

g) und zum Schluss das erste Vorkommen von 'sehr' mit 'unglaublich' ersetzt (schwieriger).

Am Ende soll die modifizierte Liste als Text ausgegeben werden. Benutze `' '.join(list_name)` um die Ausgabe zu formatieren.
Versuche die Lösung so allgemein wie möglich zu halten, z.B. sollte d) auch noch korrekt funktionieren, wenn am Anfang der Liste zusätzliche Elemente eingefügt werden.
Kommentiere dein Programm sinnvoll. Abzugeben ist das komplette von dir ergänzte, ausführbare Skript. (Bitte benenne das Dokument wie oben definiert.)

1.2 Listen nutzen

Für diese Aufgabe steht die `.txt`-Datei `kafka_trial.txt` bereit. Diese Datei beinhaltet das Buch 'The Trial' von Franz Kafka.

a) Schreibe ein Programm, welches von der BenutzerIn Endungen einliest, welche mindestens vier Buchstaben lang sind. Es soll dann im Buch von Kafka nach Wörtern suchen, welche diese Endung haben. Das Programm soll so lange nach neuen Endungen fragen, bis mindestens 150 Wörter gefunden worden sind. Beispielsausgabe:

```
Please enter a 4-letter ending you want to look for:
tter
The ending 'tter' was found 97 times.
Please enter a 4-letter ending you want to look for:
illy
The ending 'illy' was found 3 times.
...
You have found more than 150 words!
```

Wenn eine eingegebene Endung kürzer ist als 4 Buchstaben, nicht nur Buchstaben enthält oder schonmal eingegeben wurde, soll eine Fehlermeldung kommen und die BenutzerIn nochmals nach einer Endung fragen.
Beachte: Wörter die kürzer sind als 4 Buchstaben sollen nicht überprüft werden.

b) Erweitere das Programm nun so, dass es am Schluss ausgibt, wie viele und welche Endungen eingegeben und wie viele unterschiedliche Wörter (Hinweis: `set()`) gefunden wurden. Auch soll es mitteilen, welche und wie lange die 5 längsten gefundenen Wörter sind.

c) Zusatz: Das Programm soll nur Endungen akzeptieren, welche in mehr als 5 Wörtern vorkommen.

Kommentiere dein Programm sinnvoll. Abzugeben ist das komplette von dir ergänzte, ausführbare Skript. (Bitte benenne das Dokument wie oben definiert.)

2 For-Schleifen und Strings

2.1 Filtern

Für diese Übung steht das angefangene Programm `Aufgabe2_1.py` bereit. `numbers` ist eine Liste, welche 300 randomisierte Zahlen zwischen 1 und 600 enthält. (Den Rest musst du nicht verstehen, kannst es aber gerne hier nachlesen.)
Erweitere das Programm nun so, dass es folgendes ausführt:

- die Anzahl der ungeraden sowie der geraden Zahlen ausgeben
- von den ungeraden sowie den geraden Zahlen die fünf grössten absteigend sortiert ausgeben
- die Summe der fünf grössten Zahlen von je den ungeraden und den geraden Zahlen berechnen und die grössere ausgeben. Teile mit, ob es die Summe der ungeraden oder der geraden Zahlen ist.

Tipp: Die hier verwendete Liste von Zahlen ist ziemlich gross, so auch die Zahlen. Die Richtigkeit deines Codes zu kontrollieren ist also erschwert. Es macht hier Sinn, eine zusätzliche Liste mit weniger und kleineren Zahlen (aber allen möglichen Fällen) zu kreieren, um dein Programm zu testen.

2.2 Buchstabensuppe

Nimm den String „gesundheitswiederherstellungsmittelmischungsverhaeltniskundiger“, das das längste Wort des Text+Berg Korpus (SAC-Jahrbuch 1905) ist, und lass dieses mit Hilfe einer for-Schleife

ausgeben. In jedem Durchlauf der for-Schleife soll jeweils nur ein einziger Buchstabe des Wortes ausgegeben werden, und zwar so oft wie seine Position im Alphabet. Zum Beispiel tritt „g“ im Alphabet an 6ter Stelle auf. Darum wird es 6 mal ausgegeben. Wenn beispielsweise das Programm nur die Zeichen „gesundheit“ liest, sieht die Ausgabe so aus:

```
gggggg
eeee
ssssssssssssssssss
uuuuuuuuuuuuuuuuuuuu
nnnnnnnnnnnnn
dddd
hhhhhhhh
eeeee
iiiiiiiii
ttttttttttttttttttt
```

Tipp: Du erhältst die Position eines Zeichens im Alphabet mit der Funktion `string.lowercase.index()`:

```
import string
letter_pos = string.lowercase.index(a_letter)
```

Kommentiere deine Programme sinnvoll. Abzugeben sind die kompletten, von dir ergänzten, ausführbaren Skripte. (Bitte benenne die Dokumente wie oben definiert.)

3 Dictionaries

3.1 Lieblingsberge

Der Schweizer Alpen-Club SAC hat die Lieblingsberge der Mitglieder als Python Dictionary registriert. Verwende die Datei `Aufgabe3_1.py` und das darin definierte Dictionary `SAC` und schreibe ein Python-Skript, welches folgendes ausführt:

a) Füge dem Dictionary ein weiteres Schlüssel-Wert-Paar hinzu, und gib die aktuelle Dictionary aus.

- Was passiert, wenn der neue Schlüssel mit einem bisherigen Schlüssel übereinstimmt, wenn beispielsweise in unser Dictionary 'Schoenenberger':'Rigi' eingefügt wird?
- Was passiert, wenn danach 'Schoenenberger':'Eiger' eingefügt wird?

b) Gib die Schlüssel-Wert-Paare zeilenweise aus, wobei Schlüssel und Wert jeweils durch ein Tabulatorzeichen (Repräsentation: '`\t`') getrennt sein sollen. Löse diese Aufgabe mit einer for- Schleife. Die Ausgabe soll so aussehen:

```
Schoenenberger    Rigi
Zueberbuehler    Pizol
Stepankova       Pizol
...
```

c) Gib die sortierten Nachnamen aus, deren Lieblingsberge einen Namen länger als fünf Zeichen haben. Ausgabe:

```
Bizirianis
Blickenstorfer
Helbling
Rutz
...
```

3.2 Bücher analysieren

Für diese Aufgabe steht das Buch 'Moby Dick; or The Whale' von Herman Melville als `.txt`-Datei `moby_dick.txt` bereit.
Schreibe nun ein Python-Programm, welches den Datei-Pfad oder den Namen (wenn sich die Datei im selben Ordner wie das Programm befindet) einer Datei als Kommandozeilenargument erwartet. Programm Aufruf soll so aussehen:
`$ python Aufgabe3_2.py Dateiname.txt`
Nutze für die Übung das Buch von Herman, aber dein Programm soll für jedes Buch in diesem Format funktionieren.
Folgende Operationen sollen ausgeführt werden:

a) Tokenisiere jede Zeile mit der Funktion `split()`. Speichere nun jedes Wort, welches grösser als 6 Buchstaben ist, als Schlüssel in ein Dictionary und setze für dessen Wert für den Moment eine beliebige Ganzzahl, z.B. 1. Was passiert, wenn ein Wort schon im Dictionary vorhanden ist?

b) Erweitere das Programm nun so, dass die Werte im Dictionary die Anzahl der Vorkommen jedes Wortes (die Schlüssel) im Buch, welches grösser als 6 Buchstaben ist, angeben. Vorschlag: Wenn ein Token zum ersten Mal auftaucht (d.h. es ist noch kein Schlüssel im Dictionary), füge es dem Dictionary als Schlüssel mit dem Wert 1 zu. Befindet sich das Token schon im Dictionary, dann erhöhe den aktuellen Wert um 1.

c) Lass das so aufgebaute Dictionary ausgeben, und zwar entweder sortiert nach dem Schlüssel oder – wenn du herausfindest, wie – nach dem Wert (der Worthäufigkeit). Wiederum soll jedes Schlüssel-Wert-Paar auf einer eigenen Zeile zu stehen kommen.

Reflexion/Feedback

a) Fasse deine Erkenntnisse und Lernfortschritte in zwei Sätzen zusammen.

b) Wie viel Zeit hast du in diese Übungen investiert?