Übung 5: Listenkomprehension, Funktionsdefinitionen und Häufigkeitsverteilungen

Programmiertechniken in der Computerlinguistik I, HS 15

12. November 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 25. November um 18:00 offen.

Vorbereitung

Installiere bzw. update NLTK, indem du den Anweisungen hier folgst. Da eine neue NLTK Version '3.1' herausgegeben wurde im Oktober 2015, gibst im deinem Terminal folgende Befehle ein:

```
$ sudo pip install --upgrade nltk
```

Überprüfe im Python-Interpreter, ob deine Version (es soll '3.1' sein) aktuell ist:

```
$ python
>>> import nltk
>>> nltk.__version__
'3.1'
```

Installiere zudem die NLTK-Daten des Pakets "book":

```
>>> nltk.download()
```

Nun sollte ein Fenster erscheinen, in dem du die Daten auswählen kannst. Achtung: Wenn du alle NLTK-Daten (**all** anstelle von nur **book** und **all-corpora**) auswählst, benötigt das sehr viel Platz und kann eine Weile dauern. Für jede Aufgabe, welche Zugriff auf NLTK-Bücher benötigt, soll folgende Initialisierung verwendet werden:

```
import nltk
from nltk.book import *
```

Da das Laden von allen Texten relativ viel Zeit beanspruchen kann, kann man auch gezielt Korpora als Sequenz von Tokens laden mit folgendem Ersatzkode.

```
from nltk.corpus import webtext
text6 = webtext.words('grail.txt')
```

1 Mengen vs. Listen

Starte den Python-Interpreter und teste folgende Eingaben. Begründe in wenigen Sätze die Unterschiede.

a)
```
>>> set('alphabet')
>>> set(['alphabet'])
```

b)
```
>>> set(['wenn','fliegen','hinter','fliegen','fliegen',',','fliegen','fliegen','nach','.'])
>>> ['wenn','fliegen','hinter','fliegen','fliegen',',','fliegen','fliegen','nach','.']
```

Gib deine Antworten in einer PDF-Datei ab.

2 Rückgabewerte von Funktionen

In dieser Aufgabe sollst du Funktionen betrachten und interpretieren. Beachte die Wirkung von Einrückungen.

a)
```
def wc1(textfile):
    c = 0
    for line in textfile:
        for word in line.split():
            c += 1
    return c
```

Fragen:

- Wann stoppt die Funktion `wc1()`?
- Was berechnet sie?

b)
```
def wc2(textfile):
    c = 0
    for line in textfile:
        for word in line.split():
            c += 1
        return c
```

Fragen:

- Wann stoppt die Funktion `wc2()`?
- Was berechnet sie?

c)
```
def wc3(textfile):
    c = 0
    for line in textfile:
        for word in line.split():
            c += 1
            return c
```

Fragen:

- Wann stoppt die Funktion `wc3()`?
- Was berechnet sie?

Gib deine Antworten in einer PDF-Datei ab.

3 Listenkomprehension

3.1 Listenkomprehension deuten

Gegeben sei folgender Python-Ausdruck:

```
sorted([word.lower() for word in set(text6) if word >= 4 and word[-3:] == 'ing'])
```

a) Beschreibe in natürlicher Sprache und in höchstens 3 Sätzen, zu welchem Wert dieser Ausdruck evaluiert.

b) Was ändert sich, wenn der Funktionsaufruf `set()` nicht auf `text6`, sondern auf `sorted([...])` angewendet wird?

c) Definiere nun eine Funktion, welche denselben Wert berechnet und diesen als Funktionswert (`return`-Anweisung) zurückgibt, aber keine Listenkomprehension verwendet. Initialisiere `text6` wie zu Beginn beschrieben.

Gib deine Antworten in einer PDF-Datei und ein ausführbares Python-Skript ab.

3.2 Listenkomprehension verwenden

Schreibe für jede folgende Funktion einen Listenkomprhensions-Ausdruck, dessen Wert dem Rückgabewert der jeweiligen Funktion entspricht mit `text6` als Eingabewert:

a)
```
def long_words(text):
    list_out = []
    for word in set(text):
        if len(word) > 7 and word[-3:] == 'ing':
            list_out.append(word.lower())
    return sorted(list_out, key = len)
```

b)
```
def tuples(text):
    list_out = []
    for word in set(text):
        length = len(word)
        list_out.append((word,length))
    return list_out
```

c)
```
def trigrams(text):
    list_out = []
    for i in range(2,len(text)):
        trigram = (text[i-2],text[i-1],text[i])
        list_out.append(trigram)
    return list_out
```

Gib ein ausführbares Python-Skript ab.

4 Reichweite von Variablen

Evaluieren folgende Funktionen zum selben Wert? Begründe deine Antwort in wenigen Sätzen.

```
from nltk.corpus import webtext

text6 = webtext.words('grail.txt')
```

```
word = 'GLOBAL'

def space1():
    word = text6[11]
    return word

def space2():
    word2 = word
    return word2

print space1()
print space2()
```

Gib deine Antworten in einer PDF-Datei ab.

5 Funktionen verwenden

Benutze für diese Aufgabe die beigefügte `.py`-Datei `Aufgabe5.py`. Erweitere das Programm nun so, dass es eine Wortlänge und einen Anfangsbuchstaben als Kommandozeilenargumente annimmt und folgendes für `text` berechnet:

- das häufigste Wort, welches so lange ist wie die Wortlänge
- das häufigste Wort, welches mit dem eingegebenen Buchstaben beginnt
- das häufigste Wort, welches so lange ist wie die eingegebene Wortlänge und mit dem eingegebenen Anfangsbuchstaben beginnt

In dieser Aufgabe sollen **Funktionen** gebraucht werden.
Beispielsausgabe:

```
$ python Aufgabe5.py 9 b

Häufigstes Wort mit 9 Buchstaben:
launcelot : 101
Häufigstes Wort mit Anfangsbuchstabe b :
boom : 45
Häufigstes Wort mit 9 Buchstaben und Anfangsbuchstabe b :
buggering : 2
```

Zusatz: Welche Funktionen könnte man auch mit Hilfe von Listenkomprehensions-Ausdrücke lösen? Wie? Schreibe die Alternativen als Kommentar in dein Python-Skript.

Gib ein ausführbares Python-Skript ab.

6 NLTK Brown Corpus

a) Lies das Kapitel 1.3 Brown Corpus im NLTK Online Book.

b) Für diese Aufgabe steht das `.py`-Program `Aufgabe6.py` bereit. Es soll dir als Grundgerüst dienen und kann beliebig angepasst werden. Dein Programm soll für die Wörter *money, duty, love* und *fun* die Häufigkeit pro Kategorie berechnen. Da es eine ziemlich grosse Datenmenge ist und die Berechnungen dadurch relativ lange dauern können, kannst du einfacherhalber die Auswahl an Kategorien im beigelegtem Programm verwenden.

c) Erweitere dein Programm nun so, dass es für jedes Wort mitteilt, welche Kategorien den grössten bzw. kleinsten **Anteil** haben. Beispielausgabe:

```
$ python Aufgabe6.py

[ money ]
Kategorie mit grösstem Anteil ist:
romance
Kategorie mit kleinstem Anteil ist:
science_fiction
...
```

d) Schreibe zum Schluss Docstring-Kommentare für jede definierte Funktion. Je grösser ein Programmier-Projekt wird, desto wichtiger wird die Dokumentation. Docstring-Kommentare dienen u.a. dazu, Funktionen zu erklären.
Beispiel:

```
def longwords(list, int):
    """
       Find words in list which are equal or greater int.
       Return value: list of words
    """
    out = []
    ...
    return out
```

Bonus: Benutze Funktionen und Listenkomprehensions-Ausdrücke, wo möglich und wo sinnvoll.

Gib ein ausführbares Python-Skript ab.

Reflexion/Feedback

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

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