PCL I – Loesungen Uebung 05
Lennart von Thiessen (lvthiessen, 11-185-790) und Roland Benz (rolben, 97-923-163)

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.

1a)
Python Input/Output:
>>> set('alphabet')
set(['a', 'b', 'e', 'h', 'l', 'p', 't'])
>>> set(['alphabet'])
set(['alphabet'])
Antwort:
Eine Set-Datenstruktur enthält jedes Element nur einmal, entsprechend der mathematischen Definition einer Menge.
Die Python Funktion: set(InputArgument), mit InputArgument vom Datentyp String erstellt eine Menge (set) von Buchstaben (characters). Die Elemente sind alphabetisch sortiert.
Ist das InputArgument vom Datentyp List wird eine Menge von Listenelementen erstellt. Hier enhält die Liste nur ein Element.
1b)
Python Input/Output:
>>> set(['wenn','fliegen','hinter','fliegen','fliegen',',','fliegen','fliegen','nach','.'])
set(['wenn', 'nach', ',', '.', 'hinter', 'fliegen'])
>>> ['wenn','fliegen','hinter','fliegen','fliegen',',','fliegen','fliegen','nach','.']
['wenn', 'fliegen', 'hinter', 'fliegen', 'fliegen', ',', 'fliegen', 'fliegen', 'nach', '.']
Antwort:
Eine Set-Datenstruktur enthält jedes Element nur einmal, entsprechend der mathematischen Definition einer Menge.
Hier ist das InputArgument der set(InputArgument) Funktion eine List-Datenstruktur. Jedes Element der Liste erscheint somit nur einmal und nicht mehrfach.
Die List-Datenstruktur im zweiten Fall ist eine Liste und kann gleiche Elemente mehrfach enthalten.

PCL I – Loesungen Uebung 05
Lennart von Thiessen (lvthiessen, 11-185-790) und Roland Benz (rolben, 97-923-163)

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?

Antwort 2a)

textfile = codecs.open(inFilename, 'r', encoding='utf-8')
Die Funktion stoppt, wenn das Objekt textfile in der for Schleife jede Zeile abgearbeitet hat und das Dateiende (EOF) erreicht hat.
Die Funktion wc1 zählt die Wörter des textfiles. Genauer summiert sie die Anzahl Elemente auf, die sich jeweils in der Liste line.split() befinden. Die Split Methode trennt den String line mit dem Delimiter Whitespace.

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?

Antwort 2b)

Die Funktion wc2 stoppt nach der Abarbeitung der ersten Zeile von textfile.
Sie zählt die Anzahl der Wörter in der ersten Zeile. Bezüglich dem Begriff Wörter gilt dasselbe wie unter 2a)

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.

Antwort 2c)
Die Funktion wc3 stoppt nach der Verarbeitung des ersten Wortes. Bezüglich dem Begriff Wort gilt dasselbe wie unter 2a)
Sie gibt enweder 0 oder 1 zurück. Wenn das Textfile leer ist 0, und sonst 1.

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

Antwort 3a)
Die Antwort auf diese Frage hängt davon ab, welche Datenstruktur oder welcher Datentyp die Variable text6 hat. Ist sie ein String gibt der Ausdruck eine leere Liste zurück. Ist sie eine Liste, wird daraus eine Menge (Set), dh. doppelte Elemente werden aus der Inputliste gelöscht. Dann wird der Set elementweise abgearbeitet und jedes Element darauf getestet ob es mehr als 3 Zeichen enthält und ob es mit ing endet. Die Elemente, welche die Bedingungen erfüllen, werden in eine Ergebnisliste geschrieben und noch zwei weitere Befehle (lowercase() und sorted()) darauf angewendet. Es gibt somit eine sortierte Liste mit kleingeschriebenen Wörtern die auf ing enden.
Antwort 3b)
Die Datenstruktur des Outputs ist nun keine Liste mehr, sondern ein Set. Die Elemente sind identisch zur Liste aus 3a) jedoch ist es möglich, dass sie einen anderen Index haben. Dh. die Funktion Set() auf eine sortierte Liste angewendet, kann die Sortierung durcheinander bringen.
Antwort 3c)

PCL I – Loesungen Uebung 05
Lennart von Thiessen (lvthiessen, 11-185-790) und Roland Benz (rolben, 97-923-163)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
lstText=[]
'''some code to fill lstText'''
lstOutput=findWordsWithSuffixIng(lstText)
def findWordsWithSuffixIng(lstText):
    lstTmp=[]
    for word in set(lstText):
        if word >= 4 and word[-3:] == 'ing':
            lstTmp.append(word.lower())
    return sorted(lstTmp)
```

4 Reichweite von Variablen

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

Gib deine Antworten in einer PDF-Datei ab.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

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()
#KING
print space2()
#GLOBAL
```

Antwort 4)
Der folgende Analyse mit dem pdb Debugger zeigt, dass text6 eine Liste mit 16967 Elementen ist. Das Element mit Index 11 hat den Wert KING und nicht GLOBAL.

```
(Pdb) len(text6)
16967
(Pdb) text6[0:20]
[u'SCENE', u'1', u':', u'[', u'wind', u']', u'[', u'clop', u'clop', u'clop', u']', u'KING', u'ARTHUR',
u':', u'Whoa', u'there', u'!', u'[', u'clop', u'clop']
```

Somit evaluieren die zwei print Statements nicht zum selben Wert.
Die Funktion space2 gibt den Wert der globalen Variable word aus, da sich in ihrem Scope keine deklarierte/definierte und initialisierte lokale Variable mit dem Namen word befindet.

(Nebenbemerkung: Wobei bei Python die Initialisierung der lokalen Variablen zwingend ist, um ihr gegenüber der globalen Variablen den Vorrang zu geben.)
Die Funktion space1 gibt den Wert der lokalen Variablen zurück, da nun die lokale Variable word gegenüber der globalen Variablen mit demselben Namen den Vorrang erhält.

Reflexion/Feedback

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

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

Antwort b)
Roland: Kennenlernen der Listenkomprehension und des nltk package. Und insbesondere, ich kann es kaum glauben, dass es in Python nur zwei Sichtbereiche (scopes) für Variablen gibt, nämlich global und local. D.h. es ist in Python nicht möglich, Variablen in einem Block (z.B. in einer for-Schleife, oder einem if-Statement) zu deklarieren/definieren und den Sichtbereich auf diesen Block zu beschränken. Die Variablen sind automatisch im ganzen Modul (file.py) bekannt.

Antwort a)
Roland: Für Aufgabe 1 bis 4 etwa 6 Stunden, sowie etwa 1/2 Stunde um Aufgabe 5, 6 von Lennart auszuprobieren und zu verstehen.