ELFTE ÜBUNG

ZUR EINFÜHRUNG IN DIE PROGRAMMIERUNG FÜR COMPUTERLINGUISTEN

Leonie Weißweiler

20.01.2017

TWEEDBACK

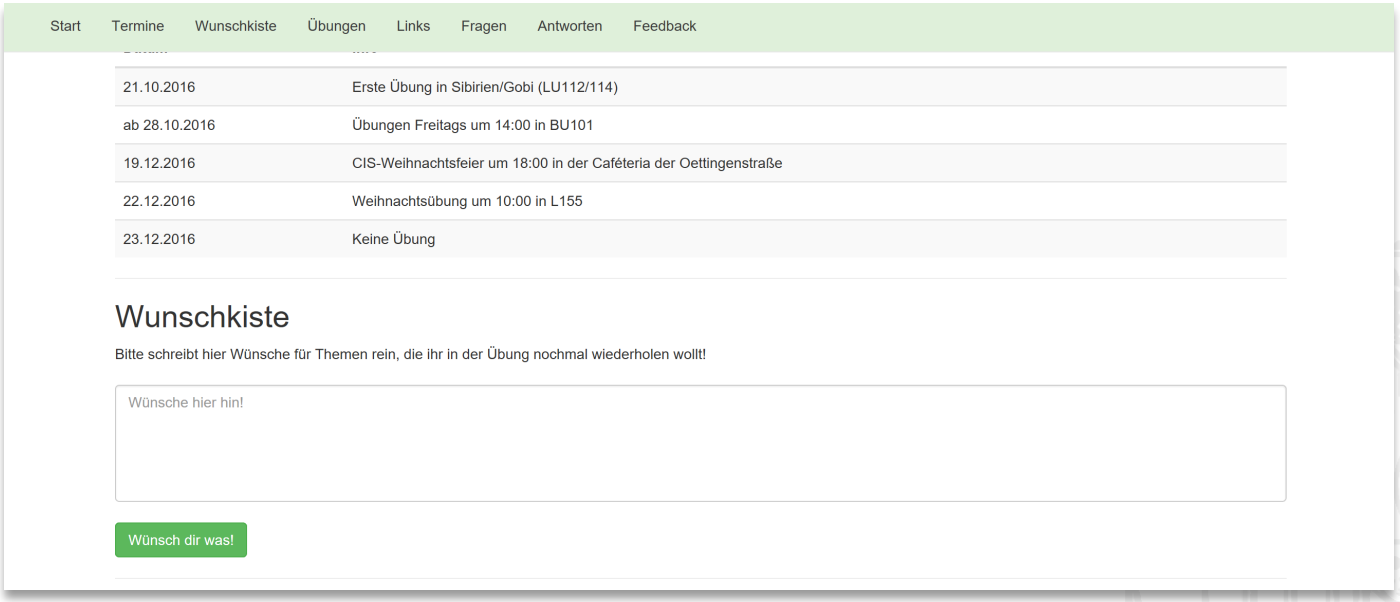
GY7

- lmu.twbk.de
- Lesson ID: **GY7**

WUNSCHKISTE

GY7

- Schickt mir Wiederholungswünsche
  http://www.cip.ifi.lmu.de/~weissweiler



WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(x):
    a = x*x
    return x

print(magic(5))
```

a) 5

b) 10

c) 25

d) 30

WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(x):
    a = x*x
    return x

print(magic(5))
```

a) 5
b) 10
c) 25
d) 30

WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(a,b):
    a = b
    return a+b

print(magic(4,5))
```

a) 8

b) 9

c) 10

d) 11

WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(a,b):
    a = b
    return a+b

print(magic(4,5))
```

a) 8

b) 9

c) 10

d) 11

WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(a):
    a = 5
    print(7)
    return 6

magic(9)
```

a) 5

b) 6

c) 7

d) 9

WIEDERHOLUNG: FUNKTIONEN

GY7

- Was gibt der Code aus?

```
def magic(a):
    a = 5
    print(7)
    return 6

magic(9)
```

a) 5

b) 6

c) 7

d) 9

WDH: ENCODINGS: ISO 8859

GY7

- ASCII enthält nur englische Buchstaben und Sonderzeichen
  - Was ist mit anderen Sprachen? äüÂøáË ĪKËÅõ
- Computer arbeiten mit 8-Bit ➔ Es sind noch 128 Möglichkeiten übrig
- $ = **0**0100100
- A = **0**1000001
- z = **0**1111010
- Ö = **1**??????
- å = **1**??????

WDH: ENCODINGS: ISO 8859

GY7

- ISO 8859 enthält 15 verschiedene Belegungen für die übrigen Plätze
  - ISO 8859-1 (Westeuropäisch)
  - ISO 8859-5 (Kyrillisch)
  - ISO 8859-11 (Thai)
- A = 01000001
- z = 01111010
- Ä = 11000100
- ü = 11111011

WDH: ENCODINGS: UNICODE

GY7

- ISO 8859 enthält jeweils nur 256 Zeichen
  - Was ist mit asiatischen Sprachen? ごみ 废话 
  - Was ist mit Dokumenten mit kyrillischen **und** deutschen „Sonderbuchstaben“?
- Es gibt mehr als $2^8 = 256$ Zeichen auf der Welt
- Es werden zwei Bit benötigt um alle Zeichen abzubilden
- In $2^{16} = 65.536$ ist genügend Platz für (fast) alle Zeichen

WDH: ENCODINGS: UTF 8

GY7

- Immer zwei Byte verwenden ist keine optimale Lösung
  - Platzverschwendung
  - Inkompatibel zu ASCII
  - Was ist wenn noch mehr Emojis erfunden werden...
  - ➢ Variable Länge
- Die ersten 127 Zeichen sind identisch zu ASCII und werden so gespeichert
  - **0**xxxxxxx = 00000000 0xxxxxxx
- Zeichen die mehr Platz benötigen werden in zwei/drei... Byte codiert
  - **110**xxxxx **10**xxxxxx = 00000xxx xxxxxxxx
  - **1110**xxxx **10**xxxxxx **10**xxxxxx = xxxxxxxx xxxxxxxx

WDH: ENCODINGS: UTF-16

GY7

- UTF-16 belegt pauschal 2 Byte (16 Bit) pro Zeichen
- Inkompatibel zu allen anderen Encodings
- Programmierer sind sich bis heute nicht einig welches Byte zuerst kommt
- Es gibt deswegen zwei „Varianten“ von UTF-16:
  - **UTF-16 L**ittle**E**ndian (zuerst das “hintere“/“niederwertige“ Byte)
  - **UTF-16 B**ig**E**ndian (zuerst das „vordere“/“hochwertige“ Byte)
- Manchmal wird als erstes ein **B**yte**O**rder**M**ark gespeichert: `11111111 11111110` (LE)
- Sonst muss man raten, aber da die meisten Texte größtenteils aus englischen Buchstaben bestehen ist das hochwertige Byte sehr häufig 00000000

FUNKTIONEN: ARGUMENTE

GY7

- Letztes Mal haben wir gesehen, dass man Funktionen Argumente übergeben kann:

```
def funktion(a,b):
    print(a+b)
```

- Nachteil: Die Funktion kann nur genau zwei Zahlen addieren
- Defaultwerte machen es möglich, einer Funktion unterschiedlich viele Argumente zu übergeben, weil für eventuell weggelassene Argumente ein Defaultwert hinterlegt ist:

```
def funktion(a,b,c=0,d=0):
    print(a+b+c+d)
```

IMMUTABLES VS MUTABLES

GY7

- Es gibt zwei Arten von Typen in Python: Mutables und Immutables
- Nur Variablen deren Typ mutable ist, können durch Methoden verändert werden
- Variablen deren Typ immutable ist, können nur neu belegt werden
- Zahlen sind immutable:
  - ```
    i = 3
    i = i + 4
    ```
- Listen sind mutable:
  - ```
    l = [0,7,2,5,1]
    l.sort()
    ```

IMMUTABLES VS MUTABLES

GY7

Immutable	Mutable
• int • float • string • boolean • range	• list • dict

CALL BY REFERENCE/ CALL BY VALUE GY7

- Es gibt zwei unterschiedliche Arten, wie ein Argument an eine Funktion übergeben werden kann:
- Call by reference
  - Das übergebene Objekt kann von der Funktion verändert werden
  - "Ich leihe dir meine Aufzeichnungen zum Abschreiben" → Du kannst auf mein Blatt schreiben
- Call by value
  - Es wird eine Kopie übergeben
  - "Ich kopiere dir meine Aufzeichnungen zum Abschreiben" → Meine Blatt bleibt wie es ist

CALL BY REFERENCE/ CALL BY VALUE GY7

- Man kann sich Call by Reference / Call by Value **nicht** aussuchen
- Immutables werden mit Call by Value aufgerufen
  - Weil sie nicht geändert werden können
- Mutables werden mit Call by Reference aufgerufen
  - Weil sie geändert werden können
  - Wenn man bei Mutables das Call by Reference umgehen will, kann man mit `list[:]` eine Kopie der Liste erstellen und mit `dict(frequenzliste)` eine Kopie des dictionarys

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(x):
    x = x + x

x = 3
magic(x)
print(x)
```

a) 3

b) 6

c) 9

d) 12

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(t):
   t.append(9)

z = [3,4,5]
magic(z)
print(z)
```

a) [3,4]

b) [4,5]

c) [3,4,5]

d) [3,4,5,9]

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(t):
    t.append(9)

z = [3,4,5]
magic(z)
print(z)
```

a) [3,4]

b) [4,5]

c) [3,4,5]

d) [3,4,5,9]

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(x):
    x = x + x

x = 3
magic(x)
print(x)
```

a) 3

b) 6

c) 9

d) 12

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(c):
    c = 7
    print(c)

t = 3
magic(t)
```

a) 3

b) 6

c) 7

d) 12

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
def magic(c):
    c = 7
    print(c)

t = 3
magic(t)
```

a) 3

b) 6

c) 7

d) 12

LOKALE UND GLOBALE VARIABLEN GY7

- Problem:
- ```
  x = 5
  def funktion():
      x += 1
  ```
- `UnboundLocalError: local variable 'x' referenced before assignment`


  ```

LOKALE UND GLOBALE VARIABLEN GY7

- Ein sog. *Namespace* ist der Bereich, in dem eine Variable existiert
- Es gibt dabei den sog. Globalen Namespace in dem "normale" Variablen existieren
- Eine Funktion hat aber ihren eigenen Namespace
  - Sie hat erstmal keinen Zugriff auf Variablen aus dem globalen Space
  - Sie kann nur durch ihre Argumente und "return" kommunizieren
- Mit `global x` baut man sich eine Verbindung zum globalen namespace auf
- **In 99% der Fälle braucht man keine solche Verbindung**

GLOBAL

GY7

```
x = 5
def funktion():
    x += 1
UnboundLocalError: local variable 'x' referenced before assignment

def richtige_funktion():
    global x
    x += 1

richtige_funktion()
x = 6

richtige_funktion()
x = 7
```


CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
x = 4
def magic():
    x = 3

magic()
print(x)
```

a) 3

b) 4

c) 7

d) 8

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
x = 4
def magic():
    x = 3

magic()
print(x)
```

a) 3

b) 4

c) 7

d) 8

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
x = 4
def magic():
    global x
    x = x + 3

magic()
magic()
print(x)
```

a) 3

b) 4

c) 7

d) 10

CALL BY REFERENCE/ CALL BY VALUE GY7

- Was gibt der Code aus?

```
x = 4
def magic():
    global x
    x = x + 3

magic()
magic()


print(x)
```

a) 3

b) 4

c) 7

d) 10

LAMBDA

GY7

- Lambda, diesesmal vernünftig erklärt:
- Ein Lambda-Ausdrucke definiert eine anonyme Funktion
- Beispiel:
- `for wort, frequenz in sorted(dict.items(),key=lambda x:x[1]):`
- Übersetzt:

```
def anonym(x):
    return x[1]
```

- Wird erstellt und an key übergeben

MUSTERLÖSUNG 11-1

GY7

- Holen Sie die ersten 4 Bücher der Bibel von Projekt Gutenberg (http://gutenberg.spiegel.de/buch/5560/i für [i=1,..,4]) mit dem UNIX Befehl wget

```
Leonie@Laptop $ wget "http://gutenberg.spiegel.de/buch/5560/1"-O 1.html
…
```

MUSTERLÖSUNG 11-2

GY7

- Verwenden sie lynx-dump um die Bücher, die in den .html Dateien gespeichert sind in eine Textdatei zu konvertieren.

```
Leonie@Laptop $ lynx –dummp –assume_charset=UTF-8 –hiddenlinks=ignore –nolist –
verbose 1.html > 1.txt
…
```

MUSTERLÖSUNG 11-3

GY7

- Fügen Sie alle Bücher zu einer Datei bibel.txt zusammen.

```
Leonie@Laptop $ cat 1.txt >  bibel.txt
Leonie@Laptop $ cat 2.txt >> bibel.txt
Leonie@Laptop $ cat 3.txt >> bibel.txt
Leonie@Laptop $ cat 4.txt >> bibel.txt
```

MUSTERLÖSUNG 11-4

GY7

Schreiben Sie eine Funktion, die eine Zeile bekommt und das längste Wort der Zeile zurückgibt.

MUSTERLÖSUNG 11-4

GY7

```
#!/usr/bin/python3
#Aufgabe 11-4
#Autorin: Leonie Weißweiler
import re
def größteswort(line):
    splitregex = re.compile(r'\w+')
    langeswort = ''
    for word in re.findall(splitregex,line):
        if(len(langeswort) < len(word)):
            langeswort = word
    return langeswort

testline = "spam, bacon, eggs and ham"
print ('Das längste Wort der Testzeile ist',größteswort(testline))
```

MUSTERLÖSUNG 11-5

GY7

Schreiben Sie eine Funktion, die eine Zeile bekommt und die Anzahl der Wörter zurückgibt.

MUSTERLÖSUNG 11-5

GY7

```
#!/usr/bin/python3
#Aufgabe 11-5
#Autorin: Leonie Weißweiler
import re
def anzahlwörter(line):
    splitregex = re.compile(r'\w+')
    anzahl = len(re.findall(splitregex,line))
    return anzahl

testline = "spam, bacon, eggs and ham"
print ('Die Testzeile ist',anzahlwörter(testline),'Wörter lang.')
```

MUSTERLÖSUNG 11-6A

GY7

Erzeugen Sie eine Frequenzliste aller Wörter aus der Datei bibel.txt.

a) Wieviele unterschiedliche Wörter kommen in der Datei vor?

MUSTERLÖSUNG 11-6A

GY7

```
#!/usr/bin/python3
#Aufgabe 11-6a
#Autorin: Leonie Weißweiler

import re

splitregex = re.compile(r'\w+')
bibel = open('bibel.txt', 'r')
```

MUSTERLÖSUNG 11-6A

GY7

```
frequenzliste = {}
for line in bibel:
    for word in re.findall(splitregex,line):
        word = word.strip()
        word = word.lower()
        if word in frequenzliste:
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1
bibel.close()

print('In der Datei kamen', len(frequenzliste),'unterschiedliche Wörter vor.')
```

MUSTERLÖSUNG 11-6B

GY7

Erzeugen Sie eine Frequenzliste aller Wörter aus der Datei bibel.txt.

a) Wieviele unterschiedliche Wörter kommen in der Datei vor?

b) Was sind die 10 häufigsten großgeschriebenen Wörter?

MUSTERLÖSUNG 11-6B

GY7

```
großregex = re.compile(r'^[A-ZÄÖÜ]')

i = 0

for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1], reverse=True):
    if i < 10 and re.search(großregex,wort):
        print (wort)
        i = i+1

bibel.close()
```

MUSTERLÖSUNG 11-6C

GY7

Erzeugen Sie eine Frequenzliste aller Wörter aus der Datei bibel.txt.

a) Wieviele unterschiedliche Wörter kommen in der Datei vor?

b) Was sind die 10 häufigsten großgeschriebenen Wörter?

c) Was sind die 10 häufigsten kleingeschriebenen Wörter?

MUSTERLÖSUNG 11-6C

GY7

```
kleinregex = re.compile(r'^[a-zäöü]')

i = 0
for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1], reverse=True):
    if i < 10 and re.search(kleinregex, wort):
        print (wort)
        i = i+1

bibel.close()
```

MUSTERLÖSUNG 11-7

GY7

Schreiben Sie ein Programm, das die ersten 3 Zeilen eines jeden Kapitels von jedem Buch ausgibt. Die Ausgabe soll folgende Form haben:

1. Buch Mose:

Kapitel 1

1. Am Anfang schuf Gott Himmel und Erde.

2. Und die Erde war wüst und leer, und es war finster auf der Tiefe; und der Geist Gottes schwebte auf dem Wasser.

3. Und Gott sprach: Es werde Licht! und es ward Licht.

MUSTERLÖSUNG 11-7

GY7

```
#!/usr/bin/python3
#Aufgabe 11-7
#Autorin: Leonie Weißweiler

import re

buchregex        = re.compile(r'^\d+. Buch \w+')
kapitelregex     = re.compile(r'^ Kapitel \d+')
versregex        = re.compile(r'^   (\d). .+')
emptylineregex   = re.compile(r'^\s*$')

bibel = open('bibel.txt', 'r')
vers_vervollständigen = False
```

MUSTERLÖSUNG 11-7

GY7

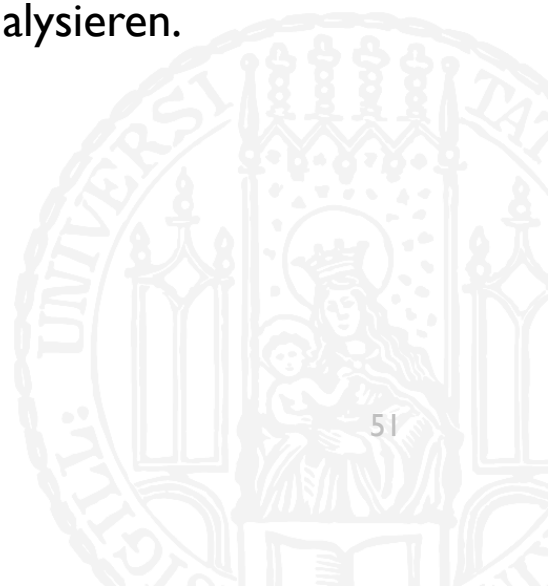
```
for line in bibel:
    if re.search(buchregex, line):
        print (line)
    elif re.search(kapitelregex, line):
        print (line)
    elif re.search(versregex, line):
        match = re.search(versregex, line)
        if int(match.group(1)) <= 3:
            print (line)
            vers_vervollständigen = True
    elif vers_vervollständigen:
        if re.search(emptylineregex, line):
            vers_vervollständigen = False
        else:
            print (line)

bibel.close()
```

MUSTERLÖSUNG 11-8

GY7

Finden sie heraus, welcher Vater die meisten Söhne hat, indem sie mit einer geeigneten Regex die Vorkommen von "Brian, Sohn des Nixus" finden und analysieren.



MUSTERLÖSUNG 11-8

GY7

```
#!/usr/bin/python3
#Aufgabe 11-8
#Autorin: Leonie Weißweiler

import re

bibel = open('bibel.txt', 'r').read()
sohn_regex = re.compile(r'(\w+), der Sohn (\w+)s')
anzahl_kinder = {}

for match in re.findall(sohn_regex, bibel):
    sohn = match[0]
    vater = match[1]
    anzahl_kinder[vater] = anzahl_kinder.get(vater, 0) + 1

meiste_kinder = 0
for vater, kinder in anzahl_kinder.items():
    if kinder > meiste_kinder:
        meiste_kinder = kinder
        kinderreichster_vater = vater
print("Der kinderreichste Vater ist", kinderreichster_vater, "mit", meiste_kinder, "Kindern")
```