ACHTE ÜBUNG

ZUR EINFÜHRUNG IN DIE PROGRAMMIERUNG FÜR COMPUTERLINGUISTEN

Leonie Weißweiler

19.12.16

TWEEDBACK

GZZ

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

WIEDERHOLUNG : ENCODINGS

GZZ

- Was ist hier passiert?

Anti terror Ã¤nderung

a) ISO-Latin als ASCII angezeigt

b) ASCII als ISO-Latin angezeigt

c) UTF-8 als ISO-Latin angezeigt

d) UTF-8 als ASCII angezeigt

WIEDERHOLUNG : ENCODINGS

GZZ

- Was ist hier passiert?

Anti terror Ã¤nderung

a) ISO-Latin als ASCII angezeigt

b) ASCII als ISO-Latin angezeigt

c) UTF-8 als ISO-Latin angezeigt

d) UTF-8 als ASCII angezeigt

QUIZ

GZZ

- Aus
  „Das ist nicht mein Traumjob“ ist
  „叔猟楳珖湩捨珖浥楮口牡睩楞“
  geworden. Was ist passiert?

a) UTF-8 als UTF-16BE angezeigt

b) UTF-16LE als UTF-8 angezeigt

c) UTF-16LE als UTF-16BE angezeigt

d) UTF-16 als ISO-Latin angezeigt

QUIZ

GZZ

- Aus
  „Das ist nicht mein Traumjob“ ist
  „叡猟楳珙湩捨珙浥楮口牡睬楞“
  geworden. Was ist passiert?

a) UTF-8 als UTF-16BE angezeigt

b) UTF-16LE als UTF-8 angezeigt

c) UTF-16LE als UTF-16BE angezeigt

d) UTF-16 als ISO-Latin angezeigt

QUIZ

GZZ

```
Leonie@Laptop $ hexdump -C datei.txt
00000000  41 6d 20 45 6e 64 65 20  69 73 74 20 65 73 20 61  |Am Ende ist es a|
00000010  62 65 72 20 45 67 61 6c  20 6f 62 20 64 69 65 20  |ber Egal ob die |
00000020  4c 65 75 74 65 20 22 73  69 6e 6e 20 6d 61 63 68  |Leute "sinn mach|
00000030  65 6e 22 20 6f 64 65 72  20 22 73 69 6e 6e 20 65  |en" oder "sinn e|
00000040  72 67 65 62 65 6e 22 20  73 61 67 65 6e 2c 20 64  |rgeben" sagen, d|
00000050  65 6e 6e 20 73 69 65 20  72 65 64 65 6e 20 6d 65  |enn sie reden me|
00000060  69 73 74 65 6e 73 20 73  6f 77 69 65 73 6f 20 6e  |istens sowieso n|
00000070  75 72 20 55 6e 73 69 6e  6e 21 0a                 |ur Unsinn!.|
0000007b
```

- Welches Encoding hat die Datei?

a) ISO-Latin

b) UTF-16LE

c) UTF-16BE

QUIZ

GZZ

```
Leonie@Laptop $ hexdump -C datei.txt
00000000  41 6d 20 45 6e 64 65 20  69 73 74 20 65 73 20 61  |Am Ende ist es a|
00000010  62 65 72 20 45 67 61 6c  20 6f 62 20 64 69 65 20  |ber Egal ob die |
00000020  4c 65 75 74 65 20 22 73  69 6e 6e 20 6d 61 63 68  |Leute "sinn mach|
00000030  65 6e 22 20 6f 64 65 72  20 22 73 69 6e 6e 20 65  |en" oder "sinn e|
00000040  72 67 65 62 65 6e 22 20  73 61 67 65 6e 2c 20 64  |rgeben" sagen, d|
00000050  65 6e 6e 20 73 69 65 20  72 65 64 65 6e 20 6d 65  |enn sie reden me|
00000060  69 73 74 65 6e 73 20 73  6f 77 69 65 73 6f 20 6e  |istens sowieso n|
00000070  75 72 20 55 6e 73 69 6e  6e 21 0a                 |ur Unsinn!.|
0000007b
```

- Welches Encoding hat die Datei?

a) ISO-Latin

b) UTF-16LE

c) UTF-16BE

WILDCARDS

GZZ

- Mit einer sog. Wildcard kann man Dateien nach ihrem Namen auswählen
- ```
  ls
  abc.txt
  defgh.txt
  xyz.py
  ```
- ```
  ls *.txt
  abc.txt
  defgh.txt
  ```
- ```
  ls ???.*
  abc.txt
  xyz.txt
  ```


  ```

REGEXES

GZZ

- In Python kann man Text ähnlich wie mit Wildcards durchsuchen
- RegExes (RegularExpression) können noch mehr als Wildcards
- Dies sind die möglichen Zeichen
  - `.` (Punkt) = ein beliebiges Zeichen
  - a = ein kleines a
  - `[a-z]` = ein kleiner Buchstabe
  - `[A-Z]` = ein großer Buchstabe
  - `[äöüÄÖÜ]` = ein Umlaut
  - `[^z]` = ein beliebiges Zeichen das kein kleines z ist

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'.o....'`

a) Goethe
b) Schiller
c) Kant
d) Lipschitz

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'.o....'`

a) Goethe
b) Schiller
c) Kant
d) Lipschitz

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'..[m-z][^n]'`

a) Seit

b) Sein

c) Kann

d) Muss

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'..[m-z][^n]'`

a) Seit

b) Sein

c) Kann

d) Muss

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'[A-Z][äöü]st'`

a) Übst

b) Obst

c) Löst

d) Dust

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'[A-Z][äöü]st'`

a) Übst

b) Obst

c) Löst

d) Dust

REGEXES

GZZ

- Man kann für Zeichen auch eine Anzahl festlegen
  - `*` = beliebig viele (0-∞)
  - `?` = eins oder keins (0-1)
  - `+` = ein oder mehr (1-∞)
  - `{1,3}` = eins bis drei (1-3)
  - `{5,}` = fünf bis beliebig (5-∞)
  - `{,2}` = keine bis zwei (0-2)

QUIZ

GZZ

- Welche der folgenden Strings werden von diesem Regex gematcht: „test“, „tttt“, “eeee“, „a“
- `‘[a-z]{2,4}'`

a) „a“

b) „tttt“, „eeee“

c) „test“

d) „test“, „tttt“, „eeee“

QUIZ

GZZ

- Welche der folgenden Strings werden von diesem Regex gematcht: „test“, „tttt“, “eeee“, „a“
- `'[a-z]{2,4}'`

a) „a“

b) „tttt“, „eeee“

c) „test“

d) „test“, „tttt“, „eeee“

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'.{1,3}'`

a) Das

b) Dies

c) Sonst

d) Selbst

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'.{1,3}'`

a) Das

b) Dies

c) Sonst

d) Selbst

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'a+.*'`

a) alle

b) Aal

c) Egaaal

d) Alles

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'a+.*'`

a) alle

b) Aal

c) Egaaal

d) Alles

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'x?B+.*'`

a) Xenial

b) Bauen

c) Xylophon

d) basteln

QUIZ

GZZ

- Was matcht der folgende Regex?
- `'x?B+.*'`

a) Xenial

b) Bauen

c) Xylophon

d) basteln

REGEXES

GZZ

- Es gibt besondere Zeichen für bestimmte Zeichen
  - `^` = Anfang des Strings
  - `$` = Ende des Strings
  - `\w` = ein „Word Character“ (Buchstabe)
  - `\s` = ein „Whitespace Character“ (Leerzeichen, Tab, etc.)
  - `\S` = kein „Whitespace Character“ (alles außer Leerzeichen, Tab, etc.)
  - `\d` = ein „digit“ (Zahl)
  - `\\` = ein tatsächliches „\“
  - `\.` = ein tatsächlicher “.“

QUIZ

GZZ

- Was matcht der folgende Regex in diesem Text?
- `'^\w+'`
-
```
"Wir verkaufen nicht nur Obst!
Auch Döner!"
```

a) „Wir“

b) „Döner!“

c) „Wir“ „verkaufen“ „nicht“ „nur“ „Auch“ „Döner“

d) „Wir“ „verkaufen“ „nicht“ „nur“ Obst“ „Auch“ Döner“

QUIZ

GZZ

- Was matcht der folgende Regex in diesem Text?
- `'^\w+'`
-
```
"Wir verkaufen nicht nur Obst!
Auch Döner!"
```

a) „Wir“

b) „Döner!“

c) „Wir“ „verkaufen“ „nicht“ „nur“ „Auch“ „Döner“

d) „Wir“ „verkaufen“ „nicht“ „nur“ Obst“ „Auch“ Döner“

QUIZ

GZZ

- Was matcht der folgende Regex in diesem Text?
- `'\w+@\w+\.\w{2,3}'`
- "Ihr könnt Fragen zum Tutorium, Aufgaben, Python, RegExs etc. jederzeit an leonie.weissweiler@campus.lmu.de, oder an kaiser@gmail.de (insbesondere Fragen zur Korrektur der Hausaufgaben) stellen."

a) nichts

b) „leonie.weissweiler@campus.lmu.de"

c) „kaiser@gmail.de"

d) „kaiser@gmail.de" „leonie.weissweiler@campus.lmu.de"

QUIZ

GZZ

- Was matcht der folgende Regex in diesem Text?
- `'\w+@\w+\.\w{2,3}'`
- "Ihr könnt Fragen zum Tutorium, Aufgaben, Python, RegExs etc. jederzeit an leonie.weissweiler@campus.lmu.de, oder an kaiser@gmail.de (insbesondere Fragen zur Korrektur der Hausaufgaben) stellen."

a) nichts

b) „leonie.weissweiler@campus.lmu.de"

c) „kaiser@gmail.de"

d) „kaiser@gmail.de" „leonie.weissweiler@campus.lmu.de"

REGEXES

GZZ

- Um in Python eine Regex zu benutzen muss man sie zuerst erzeugen
- `meine_regex = re.compile(r'b+.*')`
- Man Schreibt ein **r** vor den String um einen raw-string zu erzeugen, um Backslashes verwenden zu können
- Danach kann man sie mit verschiedenen Funktionen anwenden
  - `re.match(meine_regex, "Ein beliebig blöder String")`
    Überprüft ob die Regex am Anfang des Strings matcht
  - `re.search(meine_regex, "Ein beliebig blöder String")`
    Überprüft ob die Regex irgendwo im String matcht
  - `re.findall(meine_regex, "Ein beliebig blöder String")`
    Gibt eine Liste aller Teile des Strings zurück die matchen (ohne Überlappung)

REGEX ALS SPLIT

GZZ

Dies ist ein Text. Er enthält, manche, so 12, besondere Zeichen	
`for word in text.split(' ')`	`split_regex = re.compile(r'\w+'),` `for word in re.findall(split_regex, text)`
Dies ist ein Text. Er enthält, manche, so 12, besondere Zeichen	**Dies ist ein Text. Er enthält, manche, so 12, besondere Zeichen**

ERGÄNZUNG: REVERSE SORT

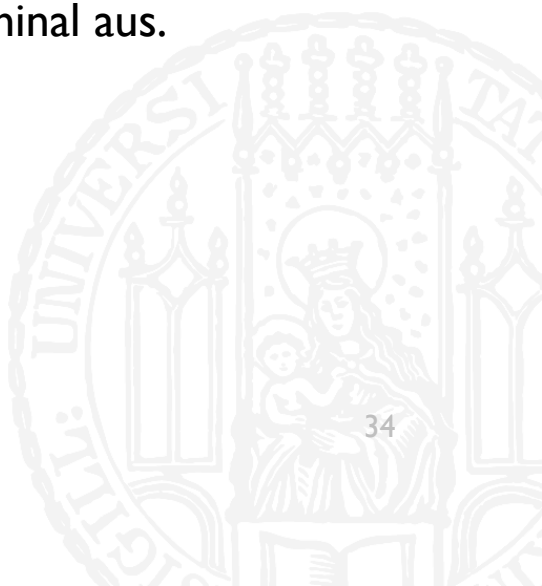
GZZ

- Man kann beim Sortieren einen „Rückwärtsgang“ einlegen:
- ```
  #nach Keys sortieren
  for key, value in sorted(dict.items(), reverse=True):
      print(key, value)
  ```
- ```
  #nach Value sortieren
  for key, value in sorted(dict.items(), key=lambda x: x[1], reverse=True):
      print(key, value)
  ```

MUSTERLÖSUNG 8-1

GZZ

Sortieren Sie die erzeugte Frequenzliste nach der Häufigkeit und geben Sie die nach der aufsteigenden Häufigkeit sortierte Frequenzliste auf dem Terminal aus.



MUSTERLÖSUNG 8-1

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-1
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}

    for word in line.split(' '):
        word = word.strip()
        word = word.lower()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

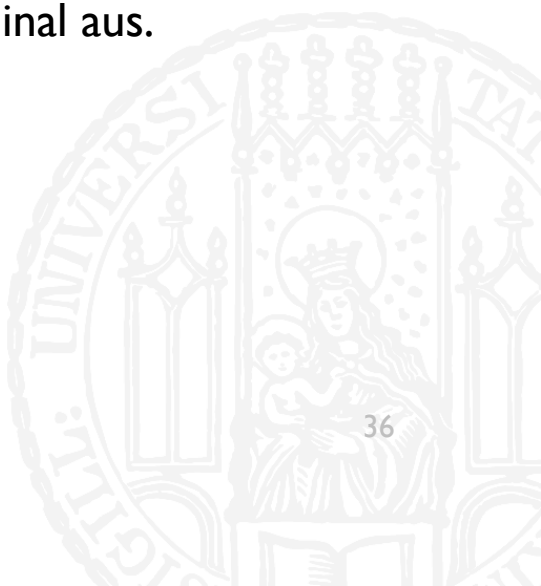
for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1]):
    print(wort, ':', frequenz)

verbrechen.close()
```

MUSTERLÖSUNG 8-2

GZZ

Sortieren Sie die erzeugte Frequenzliste nach der Häufigkeit und geben Sie die nach der absteigenden Häufigkeit sortierte Frequenzliste auf dem Terminal aus.



MUSTERLÖSUNG 8-2

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-2
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()
        word = word.lower()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1], reverse=True):
    print(wort, ':', frequenz)

verbrechen.close()
```

MUSTERLÖSUNG 8-3

GZZ

Geben Sie die zehn Wörter aus, die am häufigsten vorkommen.

MUSTERLÖSUNG 8-3

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-3
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}
allewoerter=[]

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()
        word = word.lower()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1],reverse=True):
    allewoerter.append(wort)

for x in range(0,10):
    print (allewoerter[x])

verbrechen.close()
```

MUSTERLÖSUNG 8-4

GZZ

Verwenden Sie für die nächsten Aufgaben eine Textdatei, die sie aus einer .html Datei extrahieren. Dazu folgende Hinweise:

a) Mit dem UNIX Befehl `wget : wget "http://de.wikipedia.org/wiki/Ludwig\_Wittgenstein" -O wittgenstein.html` speichern Sie von der Kommandozeile aus automatisch den Inhalt einer www-Seite in einer lokalen Datei. Mit dem obigen Befehl speichern Sie z.B. den Inhalt der Wikipedia Seite über Ludwig Wittgenstein in der Datei wittgenstein.html.

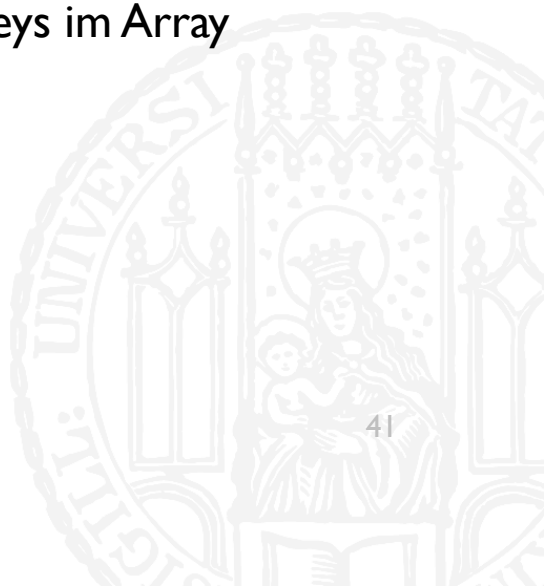
b) Mit dem UNIX Befehl lynx können Sie den Text aus der html Datei extrahieren und in einer Textdatei speichern.

```
lynx -dump -assume\_charset=UTF-8 -hiddenlinks=ignore -nolist -
verbose wittgenstein.html > wittgenstein.txt
```

MUSTERLÖSUNG 8-5

GZZ

Spalten Sie alle Zeilen der Datei wittgenstein.txt in Wörter auf und erzeugen Sie eine Frequenzliste der Wörter. Sortieren Sie die Frequenzliste absteigend nach der Häufigkeit und speichern Sie die nach der Häufigkeit sortierten keys im Array `alle_sortierten_woerter`



MUSTERLÖSUNG 8-5

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-5
#Autorin: Leonie Weißweiler

import re

splitregex = re.compile(r'\w+')
wittgenstein = open('wittgenstein.txt', 'r')
frequenzliste = {}
alle_sortierten_woerter = []

for line in wittgenstein:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1], reverse=True):
    alle_sortierten_woerter.append(wort)

print(alle_sortierten_woerter)
wittgenstein.close()
```

MUSTERLÖSUNG 8-6

GZZ

- Erweitern Sie das Programm, sodass es alle großgeschriebenen Wörter im Array `alle_sortierten_woerter` findet und die Wörter zusammen mit Ihrer Häufigkeit in der Datei "grosse_woerter.txt" speichert.

MUSTERLÖSUNG 8-6

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-6
#Autorin: Leonie Weißweiler

import re

splitregex = re.compile(r'\w+')
grosseregex = re.compile(r'^[A-ZÄÖÜ]')
wittgenstein = open('wittgenstein.txt', 'r')
grossewoerter = open('grosse_woerter.txt', 'w')
frequenzliste = {}
alle_sortierten_woerter = []

for line in wittgenstein:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1
```

MUSTERLÖSUNG 8-6

GZZ

```
for wort, frequenz in sorted(frequenzliste.items(), key=lambda
x: x[1], reverse=True):
    alle_sortierten_woerter.append(wort)

for wort in alle_sortierten_woerter:
    if(re.search(grosseregex, wort)):
        grossewoerter.write(wort)
        grossewoerter.write(' : ')
        grossewoerter.write(str(frequenzliste[wort]))
        grossewoerter.write('\n')

wittgenstein.close()
grossewoerter.close()
```

MUSTERLÖSUNG 8-7A

GZZ

Wie lauten die regulären Ausdrücke, die folgende Wörter im Array alle_sortierten_woerter finden (geben Sie nur den regulären Ausdruck an!). Testen Sie Ihre Ausdrücke mit einem Programm.

a) alle Wörter mit 3 Buchstaben

MUSTERLÖSUNG 8-7A

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-7a
#Autorin: Leonie Weißweiler

import re

splitregex = re.compile(r'\w+')
kurzregex = re.compile(r'^\w{1,3}$')
wittgenstein = open('wittgenstein.txt', 'r')
frequenzliste = {}

for line in wittgenstein:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

for wort, frequenz in sorted(frequenzliste.items(), key=lambda x: x[1]):
    if(re.search(kurzregex, wort)):
        print(wort)

wittgenstein.close()
```

MUSTERLÖSUNG 8-7A

GZZ

```
r'^\w{1,3}$'
```

MUSTERLÖSUNG 8-7B

GZZ

b) alle Wörter in denen ein Vokal vorkommt

```
r'[aeiouAEIOU]+'
```

MUSTERLÖSUNG 8-7C

GZZ

c) alle Wörter mit Umlauten

```
r'[äöüÄÖÜ]+'
```

MUSTERLÖSUNG 8-7D

GZZ

d) alle Wörter die mit 'net' enden

```
r'.*net$'
```

MUSTERLÖSUNG 8-7E

GZZ

e) alle Wörter die drei bis fünf Buchstaben lang sind und mit einem Vokal beginnen

```
r'^[aeiouAEIOU].{2,4}$'
```

MUSTERLÖSUNG 8-7F

GZZ

f) alle Wörter die mit einem Großbuchstaben beginnen und mit dem Suffix 'ung' enden

```
r'^[A-ZÄÖÜ].*ung$'
```

MUSTERLÖSUNG 8-7H

GZZ

h) alle Wörten in denen kein Vokal vorkommt

```
r'^[^aeiouöüAEIOUÄÖÜ]*$'
```

MUSTERLÖSUNG 8-8

GZZ

Schreiben Sie ein Programm, das die Anzahl aller kleingeschriebenen Wörter in der Datei wittgenstein.txt ermittelt. Geben Sie die Anzahl aus.

MUSTERLÖSUNG 8-8

GZZ

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

import re

splitregex = re.compile(r'\w+')
lowerregex = re.compile(r'^[a-zäöü]+$')
wittgenstein = open('wittgenstein.txt', 'r')
anzahllower = 0

for line in wittgenstein:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if(re.search(lowerregex, word)):
            anzahllower = anzahllower + 1

print('In der Datei waren', anzahllower, 'kleingeschriebene Wörter.')
wittgenstein.close()
```

MUSTERLÖSUNG 8-9

GZZ

Geben Sie alle Zeilen der Datei aus, die mit einem Großbuchstaben beginnen.

MUSTERLÖSUNG 8-9

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-9
#Autorin: Leonie Weißweiler

import re

großregex = re.compile(r'^[A-ZÄÖÜ]+')
wittgenstein = open('wittgenstein.txt', 'r')

for line in wittgenstein:
    line = line.strip()
    if(re.search(großregex, line)):
        print(line)

wittgenstein.close()
```

MUSTERLÖSUNG 8-10

GZZ

Geben Sie alle Zeilen der Datei aus, in denen "der", "die" oder "das" vorkommt.

MUSTERLÖSUNG 8-10

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-10
#Autorin: Leonie Weißweiler

import re

artikelregex = re.compile(r'\b(der|die|das|Der|Die|Das)\b')
wittgenstein = open('wittgenstein.txt', 'r')

for line in wittgenstein:
    line = line.strip()
    if(re.search(artikelregex, line)):
        print(line)

wittgenstein.close()
```

MUSTERLÖSUNG 8-11

GZZ

Geben Sie alle Wörter der Datei aus, die von einem Satzzeichen gefolgt sind.

MUSTERLÖSUNG 8-11

GZZ

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

import re

splitregex = re.compile(r'\w+[?!.,;;-_]')
wittgenstein = open('wittgenstein.txt', 'r')
frequenzliste = {}

for line in wittgenstein:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1
        else:
            frequenzliste[word] = 1

for key in frequenzliste.keys():
    print(key)

wittgenstein.close()
```

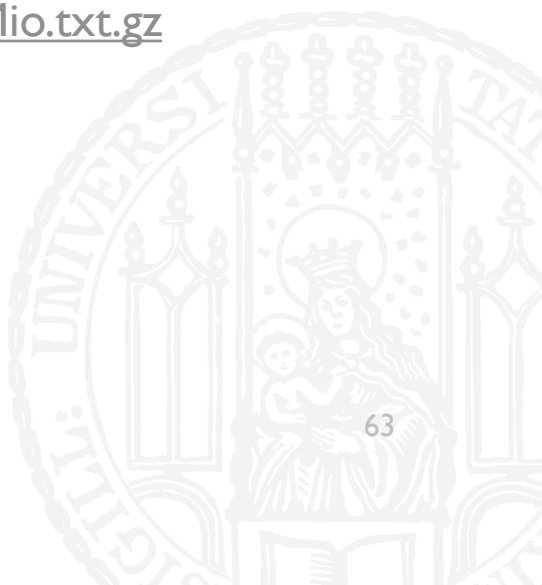
MUSTERLÖSUNG 8-12

GZZ

Schreiben Sie ein Programm, das ein Wort einliest und nachsieht, ob das von rückwärts gelesene Wortort als Teil eines Wortes in einer großen Datei vorkommt.

http://www.cis.uni-muenchen.de/kurse/max/einfprog/texte/sz_1Mio.txt.gz

Beispiel: "LOVE" => "EVOL" => Revolution => "R.EVOL.ution"



MUSTERLÖSUNG 8-12

GZZ

```
#!/usr/bin/python3
#Aufgabe 8-12
#Autorin: Leonie Weißweiler

import re

splitregex = re.compile(r'\w+')
sz = open('sz_1Mio.txt', 'r')
benutzerwort = input('Geben Sie ein Wort ein.\n')
wortrückwärts = benutzerwort[::-1]

for line in sz:
    for word in re.findall(splitregex,line):
        word = word.strip()
        if(wortrückwärts.lower() in word.lower()):
            print(word)

sz.close()
```