NEUNTE/ZEHNTE ÜBUNG

ZUR EINFÜHRUNG IN DIE PROGRAMMIERUNG FÜR COMPUTERLINGUISTEN

Leonie Weißweiler

13.01.2017

TWEEDBACK

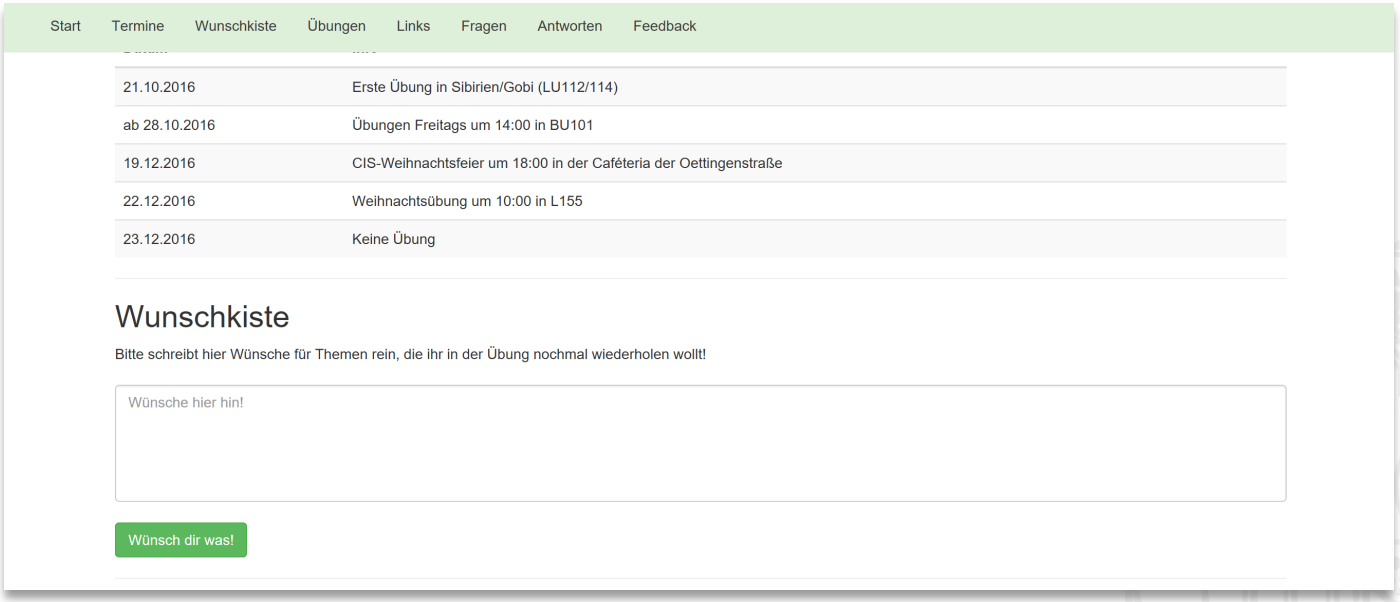
G35

- lmu.twbk.de
- Lesson ID: **G35**

WUNSCHKISTE

G35

- Schreibt gerne Wünsche zu Wiederholungen auf
  http://www.cip.ifi.lmu.de/~weissweiler



NUTZERRECHTE IN LINUX

G35

- In Linux kann jeder Nutzer verschiedenen Gruppen angehören
- Jede Datei hat einen User als Besitzer und eine Gruppe
- Der Zugriff auf die Datei wird in drei Ebenen kontrolliert:
  - Besitzer
  - Alle in der Gruppe
  - Alle
- Jede dieser Ebenen kann folgende Rechte haben:
  - Lesen (r)
  - Schreiben (w)
  - Ausführen (x)

NUTZERRECHTE IN LINUX

G35

```
rwxrwxrwx
rw-r-----
```

- Die Rechte einer Datei werden in einem langen ‘Wort’ dargestellt
- Jedes Trippel aus “rwx” steht für eine Ebene (Besitzer, Gruppe, Alle)
- In einer Ebene kann das r, w und x stehen, oder durch einen Strich ersetzt sein
- Buchstabe steht für “Recht vorhanden”, Strich für “nicht vorhanden”

NUTZERRECHTE IN LINUX

G35

```
rwxrwxrwx "alle dürfen alles"
rw-r----- "Nutzer: Lesen/Schreiben, Gruppe lesen"
```

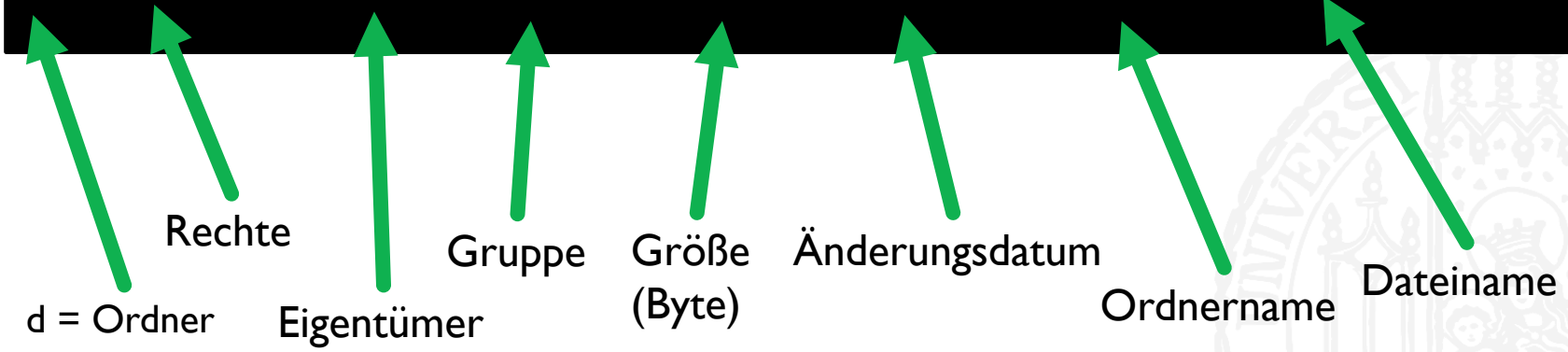
- Die Rechte einer Datei werden in einem langen 'Wort' dargestellt
- Jedes Trippel aus "rwx" steht für eine Ebene (Besitzer, Gruppe, Alle)
- In einer Ebene kann das r, w und x stehen, oder durch einen Strich ersetzt sein
- Buchstabe steht für "Recht vorhanden", Strich für "nicht vorhanden"

NUTZERRECHTE IN LINUX

G35

- Über den Command ls -l können diese Eigenschaften alle betrachtet werden

```
Leonie@Laptop $ ls -l
-rwxrw-r-- 1 Leonie EinfProg  191 Jan 12 23:14 programm.py
--w--w-rw- 1 Leonie Tutoren    28 Jun 02 16:33 schwarzes_brett.txt
drw-r----- 2 Leonie Tutoren     0 Dez 22 09:05 Website/
```



QUIZ

G35

- Was darf der Nutzer bei r-xrw--wx

a) Lesen

b) Lesen und Schreiben

c) Lesen und Ausführen

d) Schreiben und Ausführen

QUIZ

G35

- Was darf der Nutzer bei r-xrw--wx

a) Lesen

b) Lesen und Schreiben

c) Lesen und Ausführen

d) Schreiben und Ausführen

QUIZ

G35

- Was darf die Gruppe bei rwx-wxrwx

a) Lesen

b) Schreiben und Ausführen

c) Lesen und Ausführen

d) Lesen und Schreiben und Ausführen

QUIZ

G35

- Was darf die Gruppe bei rwx-wxrwx

a) Lesen

b) Schreiben und Ausführen

c) Lesen und Ausführen

d) Lesen und Schreiben und Ausführen

GREEDY / NON-GREEDY

G35

- Welchen Match eine Regex finden soll, ist nicht immer eindeutig
- Ein * oder + kann verschieden "weit" gehen
- `.*a` + `"Hey anna!"`
- `"Hey anna!"` oder `"Hey anna!"` ?
- Durch *Greedy* oder *Non-Greedy* kann der Unterschied spezifiziert werden!
- *Greedy* (englisch) **gierig**: wie gierig matcht der Regex?

GREEDY / NON-GREEDY

G35

- **Greedy**
  - Normales Verhalten
  - Matcht so weit wie **möglich**
  - `.*ein`
  - `Was für eine einsame Brücke?`
- **Non-Greedy**
  - Durch ein angehängtes Fragezeigen ausgelöst
  - Matcht nur so weit wie **nötig**
  - `.*?ein`
  - `Was für eine einsame Brücke?`

QUIZ

G35

- Was matcht 'x+?\w' auf "xxxxx" zuerst?

a) **xx**xxx

b) **xxx**xx

c) **xxxx**x

d) **xxxxx**

QUIZ

G35

- Was matcht 'x+?\w' auf "xxxxx" zuerst?

a) **xx**xxx

b) **xxx**xx

c) **xxxx**x

d) **xxxxx**

QUIZ

G35

- Was matcht '(\w+)+z' auf "Sowas ist zu zart" zuerst?

a) **Sowas ist** zu zart

b) **Sowas ist z**u zart

c) **Sowas ist zu** zart

d) **Sowas ist zu z**art

QUIZ

G35

- Was matcht '(\w+)+z' auf "Sowas ist zu zart" zuerst?

a) **Sowas ist** zu zart

b) **Sowas ist z**u zart

c) **Sowas ist zu** zart

d) **Sowas ist zu z**art

GROUPING

G35

- Manchmal sind die einzelnen Teile einer Regex interessant.
- `\d \w+`
- `Das sind die 8 Ritter!`
- Man kann sie mit Groups einschließen und danach auf diese zugreifen
- `(\d) (\w+)`
- `Das sind die 8 Ritter!`
- Es ist sogar möglich später in der Regex auf vorherige Gruppen zuzugreifen
- `(\d) (\w+) und \1 (\w+)`
- `Das sind die 8 Ritter und 8 Zauberer!`

QUIZ

G35

- Was matcht `'(\w+) \1'` auf "ein kleiner kleiner Wald ist ein Baum" zuerst?

a) **ein kleiner** kleiner Wald ist ein Baum

b) **ein** kleiner kleiner Wald ist **ein** Baum

c) **ein kleiner kleiner Wald ist ein** Baum

d) ein **kleiner kleiner** Wald ist ein Baum

e) Die Regex enthält einen Syntaxfehler

QUIZ

G35

- Was matcht `'(\w+) \1'` auf "ein kleiner kleiner Wald ist ein Baum" zuerst?

a) **ein kleiner** kleiner Wald ist ein Baum

b) **ein** kleiner kleiner Wald ist **ein** Baum

c) **ein kleiner kleiner Wald ist ein** Baum

d) ein **kleiner kleiner** Wald ist ein Baum

e) Die Regex enthält einen Syntaxfehler

FUNKTIONEN

G35

- Es ist möglich mehrfach verwendete Programmabschnitte zu gruppieren
- Man nennt diese Gruppen Funktionen und kann sie danach beliebig oft wieder aufrufen
- Eine Funktion wird mit dem Keyword **`def`**, einem Namen und `():` eingeleitet
- Danach kann sie durch `name()` beliebig aufgerufen werden
- ```
  def hallo_sagen():
      print('Hallo')

  hallo_sagen()
  ```
- **Faustregel:** anstatt Copy+Paste eine Funktion schreiben


  ```

FUNKTIONEN

G35

- Mit dem Keyword **return** kann eine Funktion auch etwas zurückliefern
- ```
  def fünf_fakultät():
      return 5*4*3*2*1

  ergebnis = fünf_fakultät()
  ```
- ```
  def aktuelles_jahr():
      return 2017

  jahr = aktuelles_jahr()
  ```

FUNKTIONEN

G35

- In den Klammern können die Funktionen Parameter erhalten
- In der Definition müssen hierzu der Reihe nach Namen vergeben werden
- Beim Aufrufen können dann entsprechend viele Parameter übergeben werden
- ```
  def vielfache_ausgeben(n):
      print(n, 2*n, 3*n, 4*n, 5*n)

  vielfache_ausgeben(2)
  >>> 2 4 6 8 10
  ```
- ```
  def multiplizieren(n,m):
      return n*m

  ergebnis = multiplizieren(5,6)
  print (ergebnis)
  >>> 30
  ```

QUIZ

G35

- Was gibt der folgende Code aus?
- ```
  def magic(n):
      print(n*3 - 5)

  magic(4)
  ```

a) 4

b) 7

c) -1;-2

d) 11


  ```

QUIZ

G35

- Was gibt der folgende Code aus?
- ```
  def magic(n):
      print(n*3 - 5)

  magic(4)
  ```

a) 4

b) 7

c) -1;-2

d) 11


  ```

QUIZ

G35

- Was gibt der folgende Code aus?
- ```
  def magic(x):
      print(x*7 - 2)
      return x*7 + 9

  a = magic(3)
  ```

a) 19

b) 30

c) 19; 30

d) 3


  ```

QUIZ

G35

- Was gibt der folgende Code aus?
- ```
  def magic(x):
      print(x*7 - 2)
      return x*7 + 9

  a = magic(3)
  ```

a) 19

b) 30

c) 19; 30

d) 3


  ```

MUSTERLÖSUNG 9-1

G35

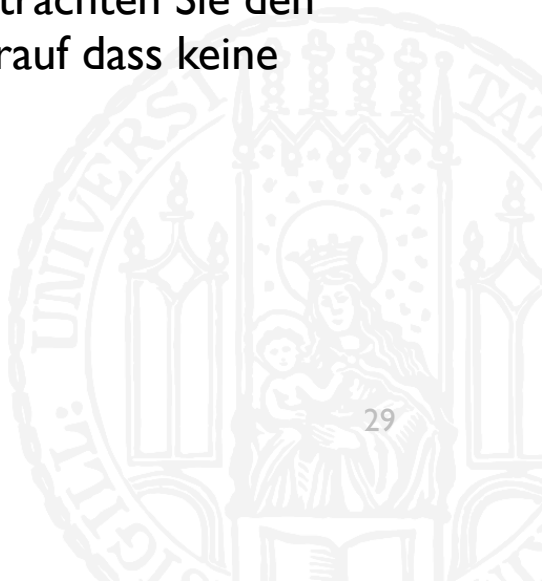
Basis	10	2	8	16
a)	15	1111	17	F
b)	22	10110	26	16
c)	256	100000000	400	100
d)	512	1000000000	1000	200
e)	1024	10000000000	2000	400
f)	2048	100000000000	4000	800

MUSTERLÖSUNG 9-2

G35

Wie sind folgende Zeichenketten in UTF-8 - und in ISO-LATIN-1 Kodierung abgespeichert?

Tipp: Speichern Sie die Wörter in einer utf8/isolatin Datei und betrachten Sie den hexdump der Datei mit dem unix-Befehl `hexdump`. Achten sie darauf dass keine newlines am Ende der Dateien sind.



MUSTERLÖSUNG 9-2A

G35

a) Zeichenkette: 'Weiß‘

```
Leonie@Laptop $ hexdump -C weiß_iso.txt
00000000  77 65 69 df
00000004

Leonie@Laptop $ hexdump -C weiß_utf8.txt
00000000  77 65 69 c3 9f
00000005
```

MUSTERLÖSUNG 9-2B

G35

a) Zeichenkette: ‘ÄäÜüÖö‘

```
Leonie@Laptop $ hexdump -C ääüüöö_iso.txt
00000000 c4 e4 dc fc d6 f6 0a
00000006

Leonie@Laptop $ hexdump -C ääüüöö_utf8.txt
00000000 c3 84 c3 a4 c3 9c c3 bc c3 96 c3 b6 0a
00000012
```

MUSTERLÖSUNG 9-3A

G35

Speichern sie das Wort "über" in einer Datei, einmal in Utf-8 (als utf8.txt) und einmal als Iso-Latin (als iso.txt).

a) Wie viele Bytes stehen in der Datei iso.txt und wie viele in der Datei utf8.txt?

```
Leonie@Laptop $ hexdump -C iso.txt
00000000 fc 62 65 72
00000004

Leonie@Laptop $ hexdump -C utf8.txt
00000000 c3 bc 62 65 72
00000005
```

MUSTERLÖSUNG 9-3B

G35

Speichern sie das Wort "über" in einer Datei, einmal in Utf-8 (als utf8.txt) und einmal als Iso-Latin (als iso.txt).

a) Wie viele Bytes stehen in der Datei iso.txt und wie viele in der Datei utf8.txt?

b) Welche Bytes sind identisch, welche nicht?

Die letzten drei Bytes welche die Zeichen "ber" repräsentieren sind identisch, davor steht bei ISO-Latin ein Byte und bei UTF-8 zwei andere Bytes.

MUSTERLÖSUNG 9-3C

G35

Speichern sie das Wort "über" in einer Datei, einmal in Utf-8 (als utf8.txt) und einmal als Iso-Latin (als iso.txt).

a) Wie viele Bytes stehen in der Datei iso.txt und wie viele in der Datei utf8.txt?

b) Welche Bytes sind identisch, welche nicht?

c) Schreiben sie ein Programm, das die Dateien mit dem Wort über in den unterschiedlichen Encodings einliest, und überprüft ob sie den gleichen Inhalt haben.

MUSTERLÖSUNG 3-C

G35

```
#!/usr/bin/python3
#Aufgabe 9-3c
#Autorin: Leonie Weißweiler

utf8 = open('utf8.txt', 'r', encoding='UTF-8')
iso = open('iso.txt', 'r', encoding='ISO-8859-1')

utf8_string = utf8.read()
iso_string = iso.read()

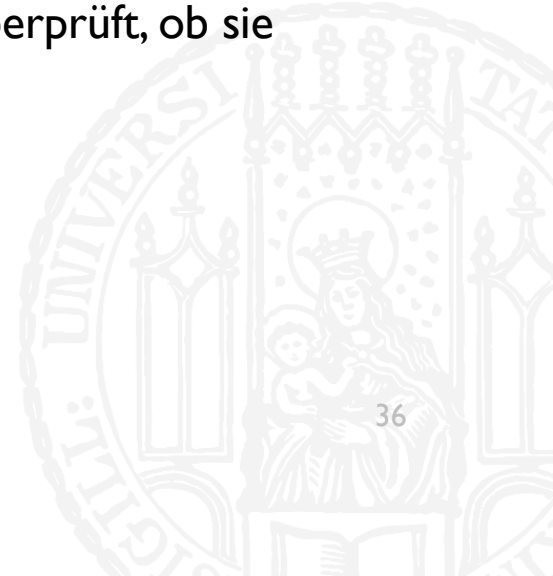
if (utf8_string == iso_string):
    print ('Der Inhalt der Dateien ist identisch.')
else:
    print ('Der Inhalt der Dateien ist nicht identisch.')
```

MUSTERLÖSUNG 9-4A

G35

Zwei Wörter sind Anagramme, wenn in ihnen die gleichen Buchstaben in beliebiger Reihenfolge vorkommen (oma ⇔ mao).

a) Schreiben Sie ein Programm, das zwei Wörter einliest und überprüft, ob sie Anagramme sind.



MUSTERLÖSUNG 9-4A

G35

```
#!/usr/bin/python3
#Aufgabe 9-4a
#Autorin: Leonie Weißweiler
input1 = input('Geben Sie das erste Wort ein\n')
input2 = input('Geben Sie das zweite Wort ein\n')
input1 = input1.lower()
input2 = input2.lower()
if(''.join(sorted(input1))==''.join(sorted(input2))):
    print('Die beiden Wörter sind Anagramme.')
else:
    print('Die beiden Wörter sind keine Anagramme.')
```

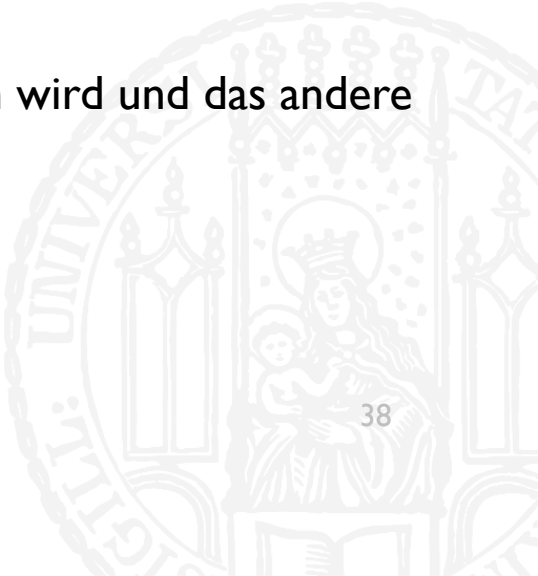
MUSTERLÖSUNG 9-4B

G35

Zwei Wörter sind Anagramme, wenn in ihnen die gleichen Buchstaben in beliebiger Reihenfolge vorkommen (oma<->mao).

a) Schreiben Sie ein Programm, das zwei Wörter einliest und überprüft, ob sie Anagramme sind.

b) Ändern Sie das Programm so ab, dass nur ein Wort eingelesen wird und das andere Wort intern gespeichert ist.



MUSTERLÖSUNG 9-4B

G35

```
#!/usr/bin/python3
#Aufgabe 9-4b
#Autorin: Leonie Weißweiler
benutzerwort = input('Geben Sie ein Wort ein\n')
benutzerwort = benutzerwort.lower()
internal = 'iamlordvoldemort'
if(''.join(sorted(benutzerwort))==''.join(sorted(internal))):
    print('Das Wort ist ein Anagramm zu IAmLordVoldemort')
else:
    print('Das Wort ist kein Anagramm zu IAmLordVoldemort')
```

MUSTERLÖSUNG 10-1

G35

- Suchen Sie alle Überschriften im Text. In Markdown-Dateien werden Überschriften mit # gekennzeichnet (#Große Überschrift, ##kleinere Überschrift etc.)
- ```
  #!/usr/bin/python3
  #Aufgabe 10-1
  #Autorin: Leonie Weißweiler

  import re
  markdown = open('Linux.md','r')
  überschriftenregex = re.compile(r'^#+')

  for line in markdown:
      if (re.search(überschriftenregex,line)):
          print (line)

  markdown.close()
  ```
  ```

MUSTERLÖSUNG 10-2

G35

- Finden sie alle Wörter oder Passagen, die als fett markiert sind (**fett** oder __fett__ steht für fett, aber *kursiv* oder _kursiv_ für kursiv)
-
```
#!/usr/bin/python3
#Aufgabe 10-2
#Autorin: Leonie Weißweiler

import re
markdown = open('Linux.md','r')
fettregex = re.compile(r'([_*]{2})(\w+)\1')

for line in markdown:
   for fett in re.findall(fettregex,line):
       print (fett[1])

markdown.close()
```

MUSTERLÖSUNG 10-3

G35

- Extrahieren Sie alle www-Links und speichern Sie sie in der Datei links.txt. WWW-Links sind in Markdown wie folgt gekennzeichnet: <http://www.cis.uni-muenchen.de>
- ```
  #!/usr/bin/python3
  #Aufgabe 10-3
  #Autorin: Leonie Weißweiler

  import re
  markdown = open('Linux.md','r')
  linkregex = re.compile(r'<http.*?>')

  for line in markdown:
     for link in re.findall(linkregex,line):
         print (link)

  markdown.close()
  ```


  ```

MUSTERLÖSUNG 10-4

G35

- Geben Sie alle Zeilen aus, in denen Linux erwähnt wird.
- ```
  #!/usr/bin/python3
  #Aufgabe 10-4
  #Autorin: Leonie Weißweiler

  import re
  markdown = open('Linux.md','r')
  linuxregex = re.compile(r'[Ll]inux')

  for line in markdown:
     if (re.search(linuxregex,line)):
         print (line)

  markdown.close()
  ```
  ```

MUSTERLÖSUNG 10-5

G35

- Suchen Sie im Text alle Wörter, in denen zwei Buchstaben doppelt hintereinander vorkommen, wie das Wort "vorko**mm**en" zum Beispiel.
- ```
  #!/usr/bin/python3
  #Aufgabe 10-5
  #Autorin: Leonie Weißweiler

  import re
  markdown = open('Linux.md','r')
  splitregex = re.compile(r'\w+')
  doppeltregex = re.compile(r'(\w)\1')

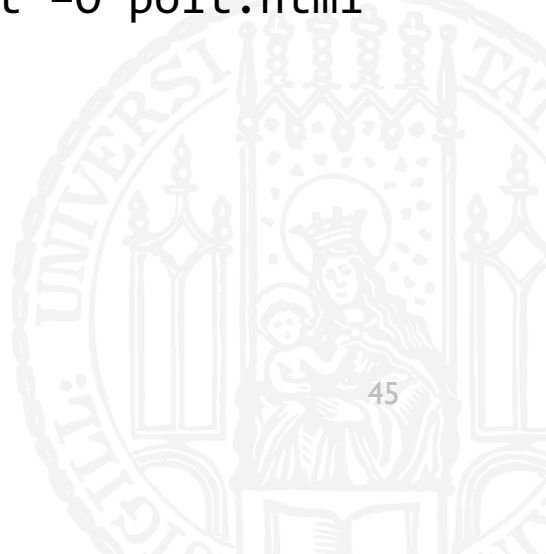
  for line in markdown:
      for word in re.findall(splitregex,line):
          if (re.search(doppeltregex,word)):
              print (word)

  markdown.close()
  ```
  ```

MUSTERLÖSUNG 10-6

G35

- Holen Sie sich mit wget den Wikipedia-Artikel über Gerhard Polt
- `wget "http://de.wikipedia.org/wiki/Gerhard_Polt"-O polt.html"`



MUSTERLÖSUNG 10-7

G35

- Extrahieren sie mit lynx dump den Text aus der html Datei
- ```
  lynx  -dump polt.html
        -assume_charset=UTF-8
        -hiddenlinks=ignore
        -nolist
        -verbose > polt.txt
  ```


  ```

MUSTERLÖSUNG 10-8

G35

- Erzeugen Sie eine Frequenzliste aller großgeschriebenen Wörter aus der Datei polt.txt, die länger als 5 Buchstaben sind und geben Sie die Anzahl der Keys aus.
- ```
  #!/usr/bin/python3
  #Aufgabe 10-8
  #Autorin: Leonie Weißweiler

  import re
  splitregex = re.compile(r'\w+')
  großregex = re.compile(r'^[A-ZÄÖÜ]\w{4,}')

  polt = open('polt.txt', 'r')
  frequenzliste = {}
  ```
  ```

MUSTERLÖSUNG 10-8

G35

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

print('In der Datei kamen', len(frequenzliste),
'unterschiedliche großgeschriebene lange Wo¨rter vor.')

polt.close()
```

MUSTERLÖSUNG 10-9

G35

- Schreiben Sie eine Funktion, die eine Zeile als Argument bekommt und die Wörter in umgekehrter Reihenfolge ausgibt, ohne slicing zu benutzen.
-
```
#!/usr/bin/python3
#Aufgabe 10-9
#Autorin: Leonie Weißweiler
import re

def reverseprint(line):
    splitregex = re.compile(r'\w+')
    linearray = re.findall(splitregex,line)
    i = 0

    while(i<len(linearray)/2):
        swap = linearray[i]
        linearray[i] = linearray[len(linearray)-i-1]
        linearray[len(linearray)-i-1] = swap
        i = i+1

    print (' '.join(linearray))

satz = input('Geben Sie einen Satz ein.\n')reverseprint(satz)
```

MUSTERLÖSUNG 10-10

G35

- Schreiben Sie eine Funktion, die eine Liste von Wörtern bekommt und jedes Wort zusammen mit der Position innerhalb der Wortliste ausdruckt. Ausgabe: Wort 1 = spam Wort 2 = and Wort 3 = eggs
- ```
  #!/usr/bin/python3
  #Aufgabe 10-10
  #Autorin: Leonie Weißweiler

  def printpositions(array):
      i = 1
      for word in array:
          print ('Wort', i, '=', word)
          i = i + 1

  testarray = ['Spam', 'and', 'Eggs']

  printpositions(testarray)
  ```


  ```

MUSTERLÖSUNG 10-11

G35

- Lesen Sie sich den Wikipedia-Artikel zu dem Gedicht "Fünfter Sein" von Ernst Jandl durch. Im Gedicht ändert sich nur ein einziges Wort, was es möglich macht, es mit wenigen Schleifenanweisungen von einem Programm ausgeben zu lassen. Schreiben Sie das Programm.
-
```
#!/usr/bin/python3
#Aufgabe 10-11
#Autorin: Leonie Weißweiler

zahlen = ['vierter', 'dritter', 'zweiter', 'nächster']

for zahl in zahlen:
    print ('tür auf')
    print ('einer raus')
    print ('einer rein')
    print (zahl,'sein\n')

print ('tür auf')
print ('einer raus')
print ('selber rein')
print ('tagherrdoktor')
```