ZWÖLFTE ÜBUNG

ZUR EINFÜHRUNG IN DIE PROGRAMMIERUNG FÜR COMPUTERLINGUISTEN

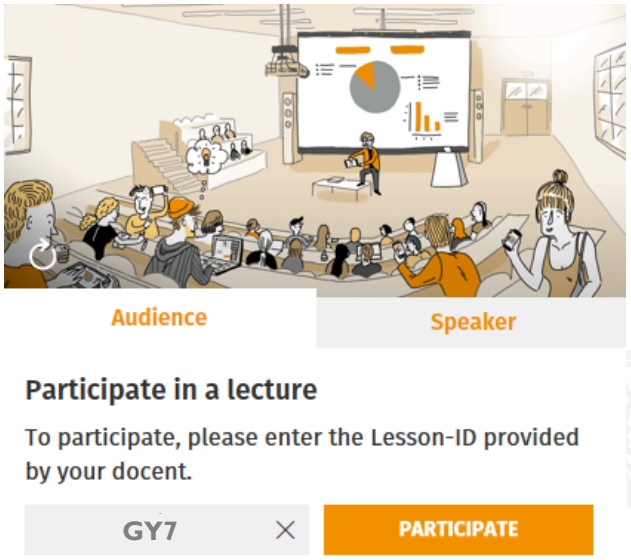
Leonie Weißweiler

29.01.17

TWEEDBACK

GVA

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



CALL BY REFERENCE/ CALL BY VALUE

GVA

- Was gibt der Code aus?

```
def brian(n):
    n = 2*n
    z = 4+n
    print(n)

x = 3
z = 4
brian(x)
print(z)
```

a) 3; 4

b) 6; 4

c) 3; 10

d) 6; 10

CALL BY REFERENCE/ CALL BY VALUE

GVA

- Was gibt der Code aus?

```
def brian(n):
    n = 2*n
    z = 4+n
    print(n)

x = 3
z = 4
brian(x)
print(z)
```

a) 3; 4

b) 6; 4

c) 3; 10

d) 6; 10

CALL BY REFERENCE/ CALL BY VALUE

GVA

- Was gibt der Code aus?

```
def youneverknow(t):
    t.append(1)
    print(t)

z = [7,6,5]
youneverknow(z)
print(z)
```

a) [7,6,5], [7,6,5]

b) [7,6,5,1], [7,6,5]

c) [7,6,5], [7,6,5,1]

d) [7,6,5,1], [7,6,5,1]

CALL BY REFERENCE/ CALL BY VALUE

GVA

- Was gibt der Code aus?

```
def youneverknow(t):
    t.append(1)
    print(t)

z = [7,6,5]
youneverknow(z)
print(z)
```

a) [7,6,5], [7,6,5]

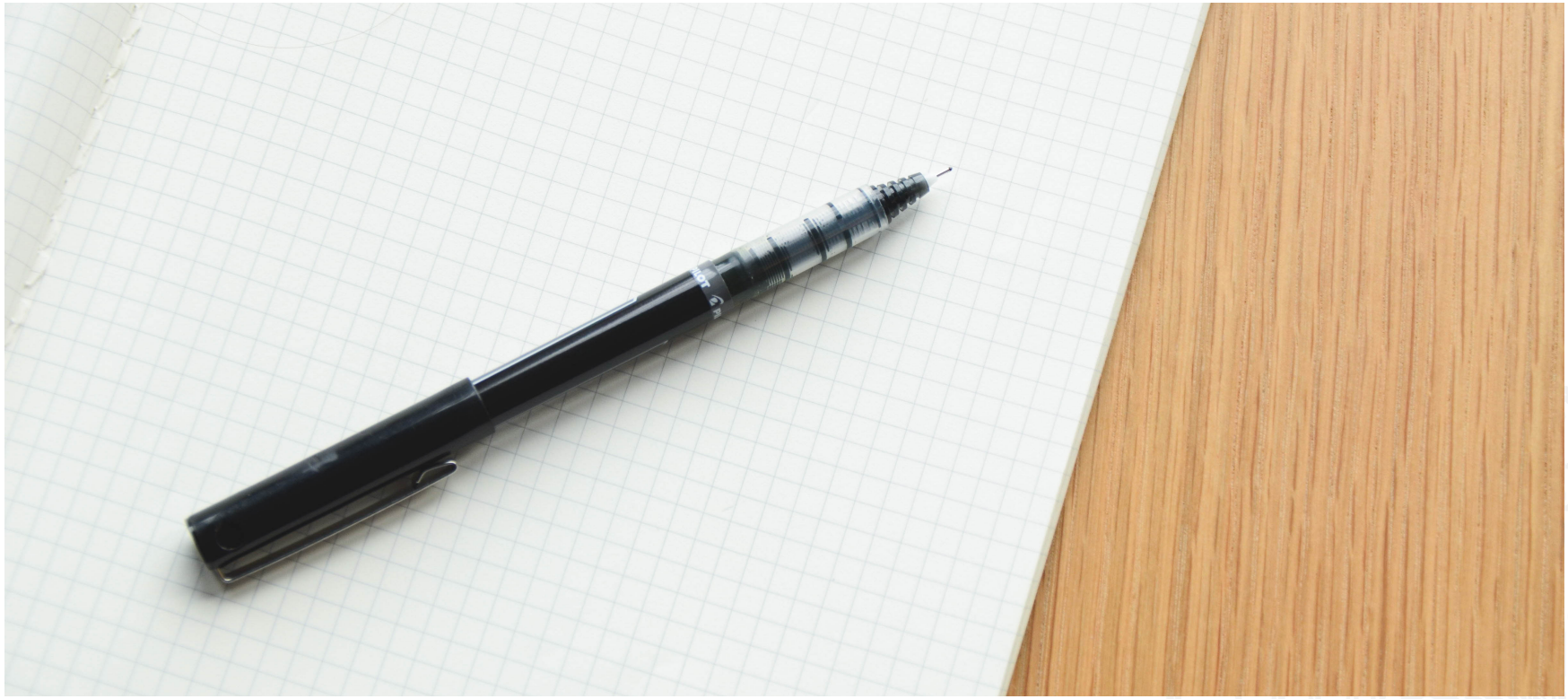
b) [7,6,5,1], [7,6,5]

c) [7,6,5], [7,6,5,1]

d) [7,6,5,1], [7,6,5,1]

BITTE DURCHGEBEN!

GVA



WIEDERHOLUNG: REGEX

GVA

Finde alle Wörter aus genau drei Kleinbuchstaben

Zeichen	Bedeutung
`.`	Beliebiges Zeichen
`f`	Ein kleines f
`\w`	„word character“ (Buchstaben, Zahlen etc)
`\d`	„digit“ (Zahl)
`[xyz]`	x **oder** y **oder** z
`(Der|Die|Das)`	„Der“ „Die“ **oder** „Das“
`(\w+)`	Die Klammern markieren eine Gruppe

Quantifier	Bedeutung
`a+`	**Ein** oder **mehr** (greedy)
`a+?`	**Ein** oder **mehr** (non-greedy)
`a*`	**Null** oder **mehr** (greedy)
`a*?`	**Null** oder **mehr** (non-greedy)
`a?`	**Null** oder **Eins**
Das **a** ist nur ein Beispiel - Quantifier kann hinter **alles** aus der Tabelle links geschrieben werden	

WIEDERHOLUNG: REGEX

GVA

Finde alle Wörter mit einem Artikel davor!

Zeichen	Bedeutung
`.`	Beliebiges Zeichen
`f`	Ein kleines f
`\w`	„word character“ (Buchstaben, Zahlen etc)
`\d`	„digit“ (Zahl)
`[xyz]`	x **oder** y **oder** z
`(Der|Die|Das)`	„Der“ „Die“ **oder** „Das“
`(\w+)`	Die Klammern markieren eine Gruppe

Quantifier	Bedeutung
`a+`	**Ein** oder **mehr** (greedy)
`a+?`	**Ein** oder **mehr** (non-greedy)
`a*`	**Null** oder **mehr** (greedy)
`a*?`	**Null** oder **mehr** (non-greedy)
`a?`	**Null** oder **Eins**
Das **a** ist nur ein Beispiel - Quantifier kann hinter **alles** aus der Tabelle links geschrieben werden	

WIEDERHOLUNG: REGEX

GVA

Finde alle Zahlen! (7 5 8,3 20.09)

Zeichen	Bedeutung
`.`	Beliebiges Zeichen
`f`	Ein kleines f
`\w`	„word character“ (Buchstaben, Zahlen etc)
`\d`	„digit“ (Zahl)
`[xyz]`	x **oder** y **oder** z
`(Der|Die|Das)`	„Der“ „Die“ **oder** „Das“
`(\w+)`	Die Klammern markieren eine Gruppe

Quantifier	Bedeutung
`a+`	**Ein** oder **mehr** (greedy)
`a+?`	**Ein** oder **mehr** (non-greedy)
`a*`	**Null** oder **mehr** (greedy)
`a*?`	**Null** oder **mehr** (non-greedy)
`a?`	**Null** oder **Eins**

Das **a** ist nur ein Beispiel - Quantifier kann hinter **alles** aus der Tabelle links geschrieben werden

WIEDERHOLUNG: REGEX

GVA

Finde alle Wörter mit einem Umlaut (ä ü ö)!

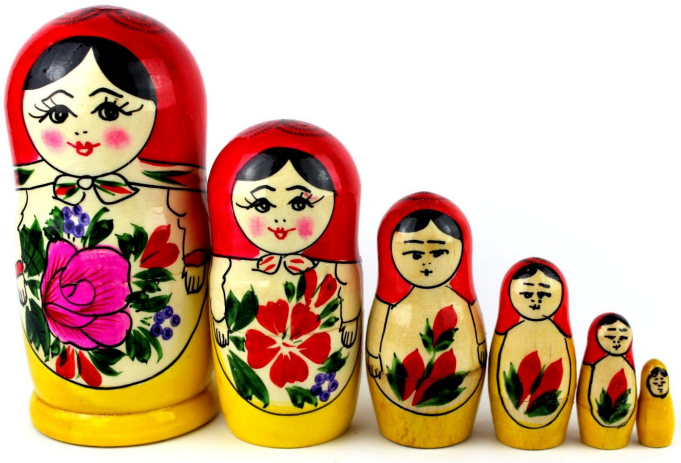
Zeichen	Bedeutung
`.`	Beliebiges Zeichen
`f`	Ein kleines f
`\w`	„word character“ (Buchstaben, Zahlen etc)
`\d`	„digit“ (Zahl)
`[xyz]`	x **oder** y **oder** z
`(Der|Die|Das)`	„Der“ „Die“ **oder** „Das“
`(\w+)`	Die Klammern markieren eine Gruppe

Quantifier	Bedeutung
`a+`	**Ein** oder **mehr** (greedy)
`a+?`	**Ein** oder **mehr** (non-greedy)
`a*`	**Null** oder **mehr** (greedy)
`a*?`	**Null** oder **mehr** (non-greedy)
`a?`	**Null** oder **Eins**
Das **a** ist nur ein Beispiel - Quantifier kann hinter **alles** aus der Tabelle links geschrieben werden	

REKURSION

GVA

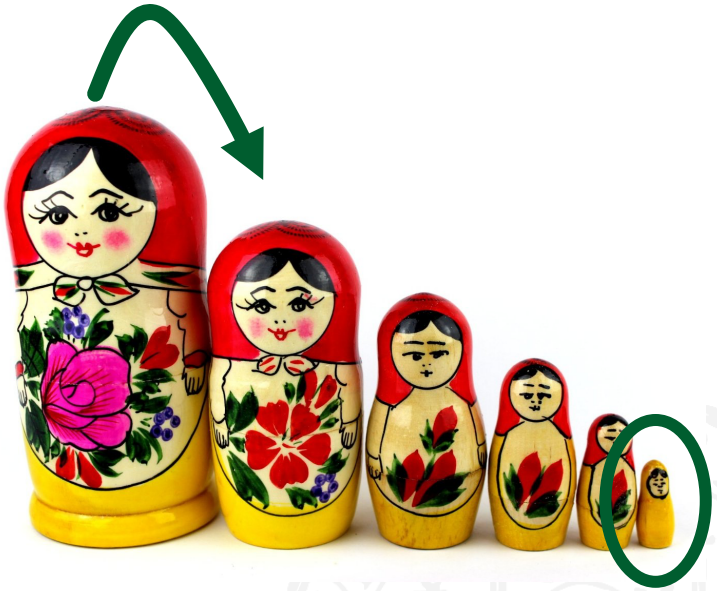
- Eine Funktion kann sich auch selbst aufrufen!
- z.B. Fakultät berechnen
- `5! = 5*4*3*2*1`
- `10! = 10*9*8*7*6*5*4*3*2*1`
- oder:
- `5! = 5 * 4!`
- `10! = 10 * 9!`
- Eine Rekusion braucht auch ein defininiertes Ende
- `1! = 1`



REKURSION

GVA

- Eine Rekursion lässt sich in zwei Fälle teilen:
  - Rekursion
  - Abbruch
- Ohne Abbruch würde die Rekursion ewig laufen
- Rekursion ist ideal um ein Problem in einfachere Teilprobleme zu zerlegen bis der einfachste Fall eintritt



REKURSION

GVA

- 4! =
- 4 * 3! =
- 4 * 3 * 2! =
- 4 * 3 * 2 * 1! =
- 4 * 3 * 2 * 1 =
- 24

REKURSION

GVA

```
def fakultät(n):
    if n == 1:
        return 1
    else:
        return n * fakultät(n-1)
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * fakultät(3)
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * if 3 == 1:
                   return 1
               else:
                  return 3 * fakultät(2)
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * if 3 == 1:
                   return 1
               else:
                  return 3 * if 2 == 1:
                                 return 1
                             else:
                                return 2 * fakultät(1)
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * if 3 == 1:
                   return 1
               else:
                  return 3 * if 2 == 1:
                                 return 1
                             else:
                                return 2 * if 1 == 1:
                                               return 1
                                           else:
                                              return 1 * fakultät
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * if 3 == 1:
                   return 1
               else:
                  return 3 * if 2 == 1:
                                 return 1
                             else:
                                return 2 * 1
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * if 3 == 1:
                   return 1
               else:
                  return 3 * 2
```

REKURSION

GVA

```
fakultät(4) =
if 4 == 1:
    return 1
else:
    return 4 * 6
```

REKURSION

GVA

```
fakultät(4) = 24
```

QUIZ

GVA

- Was gibt der Code aus?

```
def magic(u):
    if u == 0:
        return 1
    else:
        return 2 * magic(u-1)
print( magic(5) )
```

a) 1

b) 2

c) 16

d) 32

QUIZ

GVA

- Was gibt der Code aus?

```
def magic(u):
    if u == 0:
        return 1
    else:
        return 2 * magic(u-1)
print( magic(5) )
```

a) 1

b) 2

c) 16

d) 32

QUIZ

GVA

- Was gibt der Code aus?

```
def magic(t):
    if t == 1:
        return 1
    else:
        return magic(t/2) + magic(t/2)
print( magic(8) )
```

a) 2

b) 4

c) 8

d) 12

QUIZ

GVA

- Was gibt der Code aus?

```
def magic(t):
    if t == 1:
        return 1
    else:
        return magic(t/2) + magic(t/2)
print( magic(8) )
```

a) 2

b) 4

c) 8

d) 12

MUSTERLÖSUNG 12-1

GVA

- Schreiben sie eine Funktion, die rekursiv die Fakultät einer übergebenen Zahl berechnet

```
def fakultät(zahl):
    if (zahl < 0):
        return -1 * fakultät(-1*zahl)
    elif (zahl <= 1):
        return 1;
    else:
        return zahl * fakultät(zahl - 1)

testzahl = int(input('Geben sie eine Zahl ein.\n'))
print('Die Fakultät von‘ , testzahl , 'ist‘ , fakultät(testzahl))
```

MUSTERLÖSUNG 12-2

GVA

- Schreiben sie eine Funktion, die für eine Zahl x rekursiv die Summe x + (x-1) + (x-2) + ... + 1 berechnet. (z.B. summe(4) = 4+3+2+1 = 10

```
def summe(zahl):
   if (zahl == 0):
      return 0;
   elif (zahl < 0):
      return zahl + summe(zahl + 1)
   else:
      return zahl + summe(zahl - 1)

testzahl = int(input('Geben sie eine Zahl ein.\n'))
print('Die Summe von‘ , testzahl , 'ist‘ , summe (testzahl))
```

MUSTERLÖSUNG 12-3

GVA

- Schreiben Sie eine Funktion, die eine Zahl und eine Liste übergeben bekommt, rekursiv überprüft, ob die Zahl in der Liste vorhanden ist und das Ergebnis ausgibt.

```
def suche(zahl,array,pos):
    if (pos>=len(array)):
        print('Die eingegebene Zahl kam im Array nicht vor.')
    elif (array[pos]==zahl):
        print('Die eingebene Zahl kam im Array auf Position',pos,'vor.')
    else:
        return suche(zahl,array,pos+1)
testzahl = int(input('Geben Sie eine Zahl ein.\n'))
testarray = [1,2,3,4,5,6,42]
suche(testzahl ,testarray ,0)
```

MUSTERLÖSUNG 12-4

GVA

- Schreiben Sie eine Funktion, die rekursiv das Produkt zweier Zahlen berechnet, indem sie sie auf die Addition zurückführt: 2*3 = 2 + 2 + 2.

```
def produkt(zahl1,zahl2):
    if (zahl1==0 or zahl2==0):
        return 0
    elif (zahl1==1):
        return zahl2
    elif (zahl2==1):
        return zahl1
    else:
        return zahl1 + produkt(zahl1,zahl2-1)
zahl1 = int(input('Geben Sie die erste Zahl ein.\n'))
zahl2 = int(input('Geben Sie die zweite Zahl ein.\n'))
print ('Das Produkt der beiden Zahlen ist',produkt(zahl1,zahl2))
```

MUSTERLÖSUNG 12-5

GVA

- Schreiben Sie eine Funktion, die eine Liste bekommt und jedes zweite Element auf der Konsole ausgibt.

```
def jedeszweite(array):
    for zweites in array[::2]:
        print(zweites)

testarray = [1,2,3,4,5,6,42]
jedeszweite (testarray)
```

MUSTERLÖSUNG 12-6

GVA

- Schreiben Sie eine Funktion, die eine Liste bekommt, jede zweite Zahl daraus entfernt, die Reihenfolge der Zahlen umdreht und dann die Liste zurückgibt.

```
def jedeszweiteumdrehen(array):
    return array[::-2]

testarray = [1,2,3,4,5,6,42]
testarray = jedeszweiteumdrehen(testarray)

print(testarray)
```

MUSTERLÖSUNG 12-7

GVA

- Download auf http://www.cip.ifi.lmu.de/~weissweiler/#uebungen
- Fragen **jederzeit** an leonie.weissweiler@campus.lmu.de ☺