Funktionen

Ein kleines Beispiel einer Funktion:

```
def hello():
    print('Hallo')
    print('Hey!!!')
    print('Servus')
```

Hier wird unsere Funktion drei mal aufgerufen:

```
hello()
hello()
hello()
```

```
Hallo
Hey!!!
Servus
Hallo
Hey!!!
Servus
Hallo
Hey!!!
Servus
```

Zu Funktionen siehe auch: http://www.python-kurs.eu/python3_funktionen.php

Def-Anweisung mit Parametern

Wir haben schon len() oder print() als Funktion kennengelernt. Diese haben wir Argumente mitgegeben.

```
def hello(name):
    print('Hallo ' + name)

hello('Michl')
```

```
Hallo Michl
```

```
hello('Klaus')
```

```
Hallo Klaus
```

Der Rückgabewert

Mit `return` kann eine Funktion einen Wert zurückgeben.

```
import random
```

```
def getAnswer(answerNumber):
    if answerNumber == 1:
        return 'Es ist sicher'
    if answerNumber == 2:
        return 'Es ist ziemlich sicher'
    if answerNumber == 3:
        return 'Ja'
    if answerNumber == 4:
        return 'Es ist wahrscheinlich so'
    if answerNumber == 5:
        return 'Frag später nach'
    if answerNumber == 6:
        return 'Konzenteriere dich und frag noch mal nach'
    if answerNumber == 7:
        return 'Nein'
    if answerNumber == 8:
        return 'Sicherlich nicht'
    if answerNumber == 9:
        return 'Ziemlich sicher nicht'

r = random.randint(1,9)
fortune = getAnswer(r)
print(fortune)
```

```
Konzenteriere dich und frag noch mal nach
```

Der Wert None

None ist der return-Wert einer Funktion ohne Rückgabewert (klingt blöd ist aber so). Null ist so was wie nil, null etc.

```
spam = print('Hallo')
None == spam
```

```
Hallo
```

```
True
```

Schlüssewortargumente

Normalerweise bestimmt die Reihenfolge der Parameter ihrer Zuweisung.

```
r = random.randint(1, 10)
```

```
r = random.randint(10, 1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-9-00837243e5df> in <module>()
----> 1 r = random.randint(10, 1)


/opt/conda/lib/python3.5/random.py in randint(self, a, b)
    216         """
    217
--> 218         return self.randrange(a, b+1)
    219
   220    def _randbelow(self, n, int=int, maxsize=1<<BPF, type=type,


/opt/conda/lib/python3.5/random.py in randrange(self, start, stop, step, _int)
    194             return istart + self._randbelow(width)
    195         if step == 1:
--> 196         raise ValueError("empty range for randrange() (%d,%d, %d)" % (istart, istop, wid
    197
    198         # Non-unit step argument supplied.


ValueError: empty range for randrange() (10,2, -8)
```

Einige Funktionen wie print() können über optionale **Schlüsselwortangaben** der Funktion übergeben werden.

```
print('Hallo')
print('Welt')
```

```
Hallo
Welt
```

```
print('Hallo', end='')
print('Welt')
```

```
HalloWelt
```

```
print('cats', 'dogs', 'mice')
```

```
cats dogs mice
```

```
print('cats', 'dogs', 'mice', sep=',')
```

```
cats,dogs,mice
```

Mehr zu Parameter: http://www.python-kurs.eu/python3_parameter.php

Gültigkeitsbereiche

Zu globalen und lokale Variablen siehe auch: http://www.python-kurs.eu/python3_global_lokal.php

```
def spam():
    eggs= 4711

spam()
print(eggs)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-14-d19b56b5754b> in <module>()
      3
      4 spam()
----> 5 print(eggs)


NameError: name 'eggs' is not defined
```

```
def spam():
    eggs= 4711
    bacon()
    print(eggs)

def bacon():
    ham = 101
    eggs = 0

spam()
```

```
4711
```

```
def spam():
    eggs= 4711
    print(eggs)

eggs = 42
spam()
print(eggs)
```

```
4711
42
```

```
def spam():
    eggs= 'spam local'
    print(eggs)
```

```
def bacon():
    eggs= 'bacon local'
    print(eggs)
    spam()
    print(eggs)

eggs = 'global'
bacon()
print(eggs)
```

```
bacon local
spam local
bacon local
global
```

Amweisung global

```
def spam():
    global eggs
    eggs = 'spam'

eggs = 'global'
spam()
print(eggs)
```

```
spam
```

```
def spam():
    print(eggs)
    eggs= 'spam local'

eggs = 'global'
spam()
```

```
---------------------------------------------------------------------------

UnboundLocalError                         Traceback (most recent call last)

<ipython-input-19-910af5f0c812> in <module>()
      4
      5 eggs = 'global'
----> 6 spam()


<ipython-input-19-910af5f0c812> in spam()
      1 def spam():
```

```
----> 2     print(eggs)
      3     eggs= 'spam local'
      4
      5 eggs = 'global'
```

```
UnboundLocalError: local variable 'eggs' referenced before assignment
```

- Code im globalen Gültigkeitsbereich kann keine lokalen Variablen nutzen.
- Ein lokaler Gültigkeitsbereich kann dagegen auf globale Variablen zugreifen.
- Code im lokalen Gültigkeitsbereich einer Funktion kann keine Variablen aus anderen Gültigkeitsbereichen nutzen.
- Sie können für zwei Variablen den gleichen Namen wählen, sofern sie sich in unterschiedlichen Gültigkeitsbereichen befinden. Es kann also beispielsweise sowohl eine lokale als auch eine globale Variable namens spam geben.

Ausnahmebehandlung

Siehe auch: http://www.python-kurs.eu/python3_ausnahmebehandlung.php

```
def spam(divideBy):
    return 42 / divideBy

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

```
21.0
3.5
```

```
---------------------------------------------------------------------------
```

```
ZeroDivisionError                 Traceback (most recent call last)

<ipython-input-20-2d840b6c2347> in <module>()
      4 print(spam(2))
      5 print(spam(12))
----> 6 print(spam(0))
      7 print(spam(1))


<ipython-input-20-2d840b6c2347> in spam(divideBy)
```

```
      1 def spam(divideBy):
----> 2     return 42 / divideBy
      3
      4 print(spam(2))
      5 print(spam(12))


ZeroDivisionError: division by zero
```

```
def spam(divideBy):
    try:
        return 42 / divideBy
    except ZeroDivisionError:
        print('Error: Invalid atrgument.')

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

```
21.0
3.5
Error: Invalid atrgument.
None
42.0
```

Aufgabe Eine zufällige Zahl von 1 und 20 soll erraten werden. Die Anzahl der Versuche ist auf **sechs** begrenzt. Wird die Zahl erraten, soll die **Anzahl** der Versuche ausgegeben werden.

Aufgabe https://de.wikipedia.org/wiki/Collatz-Problem soll als ein Python-programm realisiert werden. Eingabe einer Zahl und Ausgabe der Reihe.