lektion2

October 18, 2018

1 Lektion 2

1.1 Boolsche Variablen und Operationen

```
In [1]: True
```

```
Out[1]: True
```

```
In [2]: False
```

```
Out[2]: False
```

```
In [3]: True and True, True and False # logisches und  'and'
```

```
Out[3]: (True, False)
```

```
In [4]: True & True # bitweises und
```

```
Out[4]: True
```

```
In [5]: True or False # logisches oder 'or'
```

```
Out[5]: True
```

```
In [6]: True | False # bitweises or
```

```
Out[6]: True
```

```
In [7]: not True  # Negation 'not'
```

```
Out[7]: False
```

1.2 Vergleiche

```
In [8]: 2 <= 2
```

```
Out[8]: True
```

```
In [9]: 2 < 2
```

```
Out[9]: False
```

```
In [10]: 1 == 2
```

```
Out[10]: False
```

```
In [11]: 'a' == 'a', [1,2] == [2,3]
```

```
Out[11]: (True, False)
```

```
In [12]: 1 != 2
```

```
Out[12]: True
```

1.3 If Anweisungen

```
In [13]: from numpy.random import rand
         c = -2+4*rand()
         print('c = {0} zu Anfang'.format(c))

         if c > 1:
             c = 1
             c = c*2
         elif c <-1:
             c =-1
         else:
             c = c**3

         print('c = {0} zu Ende'.format(c))
```

```
c = -0.8070469534908309 zu Anfang
c = -0.5256496835793638 zu Ende
```

1.4 Funktionen

```
In [14]: def mysqr2(x):
             """ Berechnet das Quadrat von x"""
             y = x**2
             return y
```

```
In [15]: ?mysqr2
```

```
In [16]: mysqr2(2)
```

```
Out[16]: 4
```

```
In [17]: def mypow(x):
             """ Berechnet x**n """
             y = x**n #n ist 'global'  NICHT verwenden
             return y

         n = 1/2
         mypow(2)
```

```
Out[17]: 1.4142135623730951
```

```
In [18]: def mypow(x,m=2):
             """ Berechnet x**m und falls m nicht gegeben ist das Quadrat von x"""
             y = x**m
             print('erstes m = {0}'.format(m))
             m = 42
             print('m = {0}'.format(m))
             return y,m

         y,m = mypow(2,3)
         m
```

```
erstes m = 3
m = 42
```

```
Out[18]: 42
```

```
In [19]: def f(a, L=[]):
             print('L in der Funktion {0}'.format(L))
             L.append(a)
             return L

         L = 23
         f(1)
         f([1,2])
```

```
L in der Funktion []
L in der Funktion [1]
```

```
Out[19]: [1, [1, 2]]
```

```
In [20]: def f(a, L=[]):
             print('L in der Funktion {0}'.format(L))
             L.append(a)
             return L

         #Der default Wert wird nur einmal ausgewertet und L ist hier veränderlich
         L = 'L ausserhalb'
         print('f(1) = {0} : {1}'.format(f(1),L))
         print('f(2) = {0} : {1}'.format(f(2),L))
         print(f(4,[2,3,1]))
```

```
L in der Funktion []
f(1) = [1] : L ausserhalb
L in der Funktion [1]
f(2) = [1, 2] : L ausserhalb
```

```
L in der Funktion [2, 3, 1]
[2, 3, 1, 4]
```

```
In [21]: def f(a, L=None):

             if L is None:
                 L = []
             L.append(a)
             return L

         print(f(1))
         print(f(2))
         print(f(3))
         print(f(4,[2,3,1]))
```

```
[1]
[2]
[3]
[2, 3, 1, 4]
```

1.5 for Schleife

```
In [22]: for i in [1,2,4]:
             print(i)
```

```
1
2
4
```

```
In [23]: for i in (1,2,3):
             print(i)
```

```
1
2
3
```

```
In [24]: for i in range(1,4):
             print(i)
```

```
1
2
3
```

```
In [25]: wb = {1:'a',2:'b',(1,2):'a+b',0:'Null'}
         for k in wb:
             print(k, wb[k])
```

```
1 a
2 b
(1, 2) a+b
0 Null
```

```
In [26]: m = {1,3,4,1,2}
         for k in m:
             print(k)
```

```
1
2
3
4
```

1.6 while Schleife

```
In [27]: a = 1
         while True:
             if a >10:
                 break
             a = a + (a+1)
             #print(a)
         a
```

```
Out[27]: 15
```

```
In [28]: from numpy.random import randint
```

```
In [29]: zaehler = 0
         summe = 0
         while summe < 100:
             summe += 1+randint(6)
             zaehler += 1
         print('Nach {0} mal würflen Augensumme {1}'.format(zaehler,summe))
```

```
Nach 28 mal würflen Augensumme 101
```

1.7 Generatoren

```
In [30]: a = []
         for _ in range(5):
             a.append([])
         a
```

```
Out[30]: [[], [], [], [], []]
```

```
In [31]: aa = [[] for _ in range(5)]
         aa

Out[31]: [[], [], [], [], []]

In [32]: { i: i**2 for i in range(5)}

Out[32]: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

1.8 Ein Beispiel: Minimum

```
In [33]: def mymin(a,b):
             """ Berechnet das Minimum der Eingabeparameter"""
             if a<b:
                 return a
             else:
                 return b

In [34]: mymin(3.2,2.1)

Out[34]: 2.1

In [35]: def mymin(*eingabewerte):
             """ Berechnet das Minimum der Betraege der Eingabeparameter"""
             print(eingabewerte)
             minimum = abs(eingabewerte[0])
             for a in eingabewerte:
                 print(a)
                 assert isinstance(a,(float,int,complex)), 'Eingabewerte muessen Zahlen sein'
                 if abs(a) < minimum:
                     minimum = abs(a)

             return minimum
         mymin(1,1+1j)

(1, (1+1j))
1
(1+1j)


Out[35]: 1

In [36]: mymin(2,2.1,3,1+2j)

(2, 2.1, 3, (1+2j))
2
2.1
3
(1+2j)
```

```
Out[36]: 2

In [37]: min([1,2,3,1.0,1+1j])
```

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

<ipython-input-37-1420914b6023> in <module>
----> 1 min([1,2,3,1.0,1+1j])


TypeError: '<' not supported between instances of 'complex' and 'int'
```