

Programmieren mit Python

Maturavorbereitung

Aufgabe 1

Beschreibe die Programmiersprache Python. Wer hat sie entwickelt? Was ist ihr Einsatzgebiet? Was worin bestehen die Unterschiede zu anderen Programmiersprachen?

Aufgabe 1

Hier habe ich mir erlaubt die Antwort von einer KI formulieren zu lassen und diese sowohl ergänzt als auch und gekürzt.

Python ist eine gut lesbare, interpretierte und objektorientiert Programmiersprache, die für ihre Einfachheit und Flexibilität bekannt ist.

Entwickler: Python wurde in den späten 1980er Jahren von Guido van Rossum entwickelt und erstmals 1991 veröffentlicht.

Einsatzgebiete: Python ist eine *universelle Programmiersprache* und wird daher in vielen Anwendungsbereichen eingesetzt:

- ▶ *Webentwicklung:* Mit Frameworks wie Django und Flask. Datenanalyse und Wissenschaftliches Rechnen: Bibliotheken wie NumPy, Pandas und SciPy sind weit verbreitet.
- ▶ *Künstliche Intelligenz und Maschinelles Lernen:* TensorFlow, Keras und scikit-learn sind beliebte Bibliotheken.
- ▶ *Automatisierung und Skripting:* Python wird häufig für Automatisierungsaufgaben und Skripting verwendet.

Unterschiede zu anderen Programmiersprachen:

- ▶ *Syntax:* Python hat eine klare und leicht verständliche Syntax, die es Anfängern erleichtert, die Sprache zu lernen. Im Vergleich zu Sprachen wie C++ oder Java ist der Code oft kürzer und prägnanter. Die Einrückung fördert eine klare und konsistente Struktur des Codes, was die Lesbarkeit bei kurzen Programmen erhöht. Umgekehrt können falsche Einrückungen zu schwer zu findenden Fehlern führen.
- ▶ *Dynamische Typisierung:* Python verwendet dynamische Typisierung, was bedeutet, dass Variablen nicht deklariert werden müssen und Typen zur Laufzeit zugewiesen werden. Dies unterscheidet sich von statisch typisierten Sprachen wie Java oder C/C++.

- ▶ *Interpretation statt Kompilation*: Python ist eine interpretierte Sprache, was bedeutet, dass der Code Zeile für Zeile ausgeführt wird. Im Gegensatz dazu werden Sprachen wie Java oder C/C++ in Maschinencode kompiliert, bevor sie ausgeführt werden.
- ▶ *Standardbibliothek*: Python bietet eine umfangreiche Standardbibliothek, die viele Funktionen und Module bereitstellt, die in anderen Sprachen möglicherweise nicht standardmässig verfügbar sind.

Aufgabe 2

```
print(1 + 2 * 3 - 4)
```

Aufgabe 2

3

Aufgabe 3

```
print(17 // 3)
```

Aufgabe 3

5

Aufgabe 4

```
print(17 % 3)
```

Aufgabe 4

2

Aufgabe 5

```
print(2**3)
```

Aufgabe 5

8

Aufgabe 6

```
print(4**0.5)
```

Aufgabe 6

2.0

Aufgabe 7

```
print(-3**2)
```

Aufgabe 7

-9

Aufgabe 8

Welcher Datentyp hat das Resultat des Ausdrucks in Python?
(ohne Berechnung)

```
print(type(4 * 9))  
print(type(5.0 + 3))  
print(type(8 / 2))  
print(type(8 // 3))  
print(type(7.0 // 4))  
print(type(9**0.5))
```

Aufgabe 8

```
<class 'int'>  
<class 'float'>  
<class 'float'>  
<class 'int'>  
<class 'float'>  
<class 'float'>
```

Aufgabe 9

```
a = -1
```

```
b = 7
```

```
a = b
```

```
b = a
```

```
print(a)
```

```
print(b)
```

Aufgabe 9

```
a = -1
```

```
b = 7
```

```
a = b
```

```
b = a
```

```
print(a)
```

```
print(b)
```

```
7
```

```
7
```

Aufgabe 10

```
a = 7
```

```
a += 3
```

```
a *= 9
```

```
print(a)
```

Aufgabe 10

```
a = 7
```

```
a += 3
```

```
a *= 9
```

```
print(a)
```

```
90
```

Aufgabe 11

```
a, b, c = 5, 3, 8
```

```
x, y, z = c, a, b
```

```
print(z)
```

Aufgabe 11

```
a, b, c = 5, 3, 8
```

```
x, y, z = c, a, b
```

```
print(z)
```

```
3
```

Aufgabe 12

```
x = y = z = 3  
print(x+y+z)
```

Aufgabe 12

```
x = y = z = 3
```

```
print(x+y+z)
```

9

Aufgabe 13

```
print(5 != 5)
```

Aufgabe 13

```
print(5 != 5)
```

False

Aufgabe 14

```
print(True and False)
```

Aufgabe 14

```
print(True and False)
```

```
False
```

Aufgabe 15

```
print((5 < 7) or (3 > 8))
```

Aufgabe 15

```
print((5 < 7) or (3 > 8))
```

True

Aufgabe 16

```
print(not (2 < 1) and (7 > 6))
```

Aufgabe 16

```
print(not (2 < 1) and (7 > 6))
```

```
True
```

Aufgabe 17

```
print(True and True and True and True and False)
```

Aufgabe 17

```
print(True and True and True and True and False)
```

```
False
```

Aufgabe 18

```
x = 3.1
```

```
print(2.8 < x < 7.6)
```

Aufgabe 18

```
x = 3.1
```

```
print(2.8 < x < 7.6)
```

```
True
```

Aufgabe 19

```
print(False or not True and False)
```

Aufgabe 19

```
print(False or not True and False)
```

```
False
```

Aufgabe 20

```
print(not not not not not not not not True)
```

Aufgabe 20

```
print(not not not not not not not not True)
```

```
True
```

Aufgabe 21

```
x = 7
if x != 1:
    x = x + 3
x = x + 1
print(x)
```

Aufgabe 21

```
x = 7
if x != 1:
    x = x + 3
x = x + 1
print(x)
```

11

Aufgabe 22

```
x = 7
if x != 5:
    x = x + 4
else:
    x = x + 3
x = x + 1
print(x)
```

Aufgabe 22

```
x = 7
if x != 5:
    x = x + 4
else:
    x = x + 3
x = x + 1
print(x)
```

12

Aufgabe 23

```
z = 9
if z < 2:
    z = z + 1
elif z < 4:
    z = z + 2
elif z < 6:
    z = z + 3
else:
    z = z + 4
print(z)
```

Aufgabe 23

```
z = 9
if z < 2:
    z = z + 1
elif z < 4:
    z = z + 2
elif z < 6:
    z = z + 3
else:
    z = z + 4
print(z)
```

13

Aufgabe 24

```
b = 7
if b == 7:
    if b > 5:
        b = b + 1
    else:
        b = b + 2
else:
    if b >= 4:
        b = b + 3
    else:
        b = b + 4
print(b)
```

Aufgabe 24

```
b = 7
if b == 7:
    if b > 5:
        b = b + 1
    else:
        b = b + 2
else:
    if b >= 4:
        b = b + 3
    else:
        b = b + 4
print(b)
```

8

Aufgabe 25

```
for i in range(3, 5):  
    print(2*i)
```

Aufgabe 25

```
for i in range(3, 5):  
    print(2*i)
```

6

8

Aufgabe 26

```
for k in range(1, 10, 4):  
    print(k)
```

Aufgabe 26

```
for k in range(1, 10, 4):  
    print(k)
```

1

5

9

Aufgabe 27

```
for j in range(10, 7, -1):  
    print(j)
```

Aufgabe 27

```
for j in range(10, 7, -1):  
    print(j)
```

10

9

8

Aufgabe 28

```
for e in [3, -8, 7, -4, 0]:  
    if e > 0:  
        print(e)
```

Aufgabe 28

```
for e in [3, -8, 7, -4, 0]:  
    if e > 0:  
        print(e)
```

3

7

Aufgabe 29

```
i = 0
while (i < 10):
    i = 2*i+1
    print(i)
```

Aufgabe 29

```
i = 0
while (i < 10):
    i = 2*i+1
    print(i)
```

1

3

7

15

Aufgabe 30

```
i = 0
while (i < 10):
    i = 2*i+1
print(i)
```

Aufgabe 30

```
i = 0
while (i < 10):
    i = 2*i+1
print(i)
```

15

Aufgabe 31

```
i = 4
s = 0
while (s < 20):
    s = s + i
    i = i + 3
    print(s)
```

Aufgabe 31

```
i = 4
s = 0
while (s < 20):
    s = s + i
    i = i + 3
    print(s)
```

```
4
11
21
```

Aufgabe 32

Handelt es sich beim folgenden Python-Programmfragment um eine Endlosschleife?

```
i = 0
while (i < 10):
    print(i)
```

Aufgabe 32

```
i = 0
while (i < 10):
    print(i)
```

Ja, denn die die Abbruchbedingung kann nicht erreicht werden.

Aufgabe 33

Handelt es sich bei dem folgenden Python-Programmfragment um eine Endlosschleife?

```
i = 10
while i != 0:
    i = i - 2
```

Aufgabe 33

```
i = 10
while i != 0:
    i = i - 2
```

Nein, denn die Abbruchbedingung $i=0$ wird nach 5 Schleifendurchläufen erreicht.

Aufgabe 34

```
for e in [4, -7, 6, 8]:  
    if e > 5:  
        break  
    else:  
        print(e)
```

Aufgabe 34

```
for e in [4, -7, 6, 8]:  
    if e > 5:  
        break  
    else:  
        print(e)
```

4

-7

Aufgabe 35

```
for i in range(1, 11):  
    if i % 3 == 0:  
        continue  
    else:  
        print(i)
```

Aufgabe 35

```
for i in range(1, 11):  
    if i % 3 == 0:  
        continue  
    else:  
        print(i)
```

1

2

4

5

7

8

10

Aufgabe 36

```
for i in range(2, 4):  
    for j in range(3, 5):  
        print(i+j)
```

Aufgabe 36

```
for i in range(2, 4):  
    for j in range(3, 5):  
        print(i+j)
```

5

6

6

7

Aufgabe 37

```
L = [3, -7, 1, 5, 8]  
print(L[2])
```

Aufgabe 37

```
L = [3, -7, 1, 5, 8]  
print(L[2])
```

1

Aufgabe 38

```
L = [[5, 3, 2], [1, 7], [9, 4, 8]]  
print(L[2][0])
```

Aufgabe 38

```
L = [[5, 3, 2], [1, 7], [9, 4, 8]]  
print(L[2][0])
```

9

Aufgabe 39

```
L = [9, 3, 4, 2, 7, 5]  
print(L[-2])
```

Aufgabe 39

```
L = [9, 3, 4, 2, 7, 5]  
print(L[-2])
```

7

Aufgabe 40

```
L = [9, 3, 4, 2, 7, 5]  
print(len(L))
```

Aufgabe 40

```
L = [9, 3, 4, 2, 7, 5]  
print(len(L))
```

6

Aufgabe 41

```
L = [[5, 3, 2], [1, 7], [9, 4, 8]]  
print(len(L))
```

Aufgabe 41

```
L = [[5, 3, 2], [1, 7], [9, 4, 8]]  
print(len(L))
```

3

Aufgabe 42

```
L = [8,1,4,9]
```

```
L[2] = 7
```

```
print(L)
```

Aufgabe 42

```
L = [8,1,4,9]
```

```
L[2] = 7
```

```
print(L)
```

```
[8, 1, 7, 9]
```

Aufgabe 43

```
L = [5, 2, 3]
```

```
L.append(7)
```

```
print(L)
```

Aufgabe 43

```
L = [5, 2, 3]
```

```
L.append(7)
```

```
print(L)
```

```
[5, 2, 3, 7]
```

Aufgabe 44

```
L = [5, 9, 8]  
L.insert(1, 2)  
print(L)
```

Aufgabe 44

```
L = [5, 9, 8]  
L.insert(1, 2)  
print(L)
```

```
[5, 2, 9, 8]
```

Aufgabe 45

```
L = [7, 9, 5, 8, 2]
```

```
L.pop()
```

```
print(L)
```

Aufgabe 45

```
L = [7, 9, 5, 8, 2]
```

```
L.pop()
```

```
print(L)
```

```
[7, 9, 5, 8]
```

Aufgabe 46

```
L = [7, 9, 5, 8, 2]
```

```
x = L.pop()
```

```
print(x)
```

Aufgabe 46

```
L = [7, 9, 5, 8, 2]
```

```
x = L.pop()
```

```
print(x)
```

2

Aufgabe 47

```
L = [7, 9, 5, 8, 2]
```

```
x = L.pop(2)
```

```
print(x)
```

Aufgabe 47

```
L = [7, 9, 5, 8, 2]
```

```
x = L.pop(2)
```

```
print(x)
```

5

Aufgabe 48

```
A = [9, 3, 2]
```

```
B = [4, 1]
```

```
print(A + B)
```

Aufgabe 48

```
A = [9, 3, 2]
```

```
B = [4, 1]
```

```
print(A + B)
```

```
[9, 3, 2, 4, 1]
```

Aufgabe 49

```
print(3 * [7,2])
```

Aufgabe 49

```
print(3 * [7,2])
```

```
[7, 2, 7, 2, 7, 2]
```

Aufgabe 50

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[2:5])
```

Aufgabe 50

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[2:5])
```

```
[5, 8, 2]
```

Aufgabe 51

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[:3])
```

Aufgabe 51

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[:3])
```

```
[7, 9, 5]
```

Aufgabe 52

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[6:])
```

Aufgabe 52

```
L = [7, 9, 5, 8, 2, 1, 4, 3]  
print(L[6:])
```

```
[4, 3]
```

Aufgabe 53

```
A = [6, 7, 2]
```

```
B = A
```

```
B[1] = 9
```

```
print(A)
```

Aufgabe 53

```
A = [6, 7, 2]
```

```
B = A
```

```
B[1] = 9
```

```
print(A)
```

```
[6, 9, 2]
```

Aufgabe 54

```
A = [6, 7, 2]
```

```
B = A[:]
```

```
B[1] = 9
```

```
print(A)
```

Aufgabe 54

```
A = [6, 7, 2]
```

```
B = A[:]
```

```
B[1] = 9
```

```
print(A)
```

```
[6, 7, 2]
```

Aufgabe 55

```
L = [6, 1, 2, 7, 9]
```

```
L.reverse()
```

```
print(L)
```

Aufgabe 55

```
L = [6, 1, 2, 7, 9]
```

```
L.reverse()
```

```
print(L)
```

```
[9, 7, 2, 1, 6]
```

Aufgabe 56

```
L = [6, 1, 2, 7, 9]
```

```
L.sort()
```

```
print(L)
```

Aufgabe 56

```
L = [6, 1, 2, 7, 9]
```

```
L.sort()
```

```
print(L)
```

```
[1, 2, 6, 7, 9]
```

Aufgabe 57

```
T = (9, 3, 4, 2, 7, 5)  
print(T[3])
```

Aufgabe 57

```
T = (9, 3, 4, 2, 7, 5)  
print(T[3])
```

2

Aufgabe 58

```
L = (9, 3, 4, 2, 7, 5)
```

```
L[3] = 8
```

```
print(L)
```

Aufgabe 58

```
L = (9, 3, 4, 2, 7, 5)
```

```
L[3] = 8
```

```
print(L)
```

```
...
```

```
    L[3] = 8
```

```
    ~~~~
```

```
TypeError: 'tuple' object does not support item assignment
```

Aufgabe 59

```
L = (9, 3, 4, 2, 7, 5)  
print(L[1:4])
```

Aufgabe 59

```
L = (9, 3, 4, 2, 7, 5)
```

```
print(L[1:4])
```

```
(3, 4, 2)
```

Aufgabe 60

```
L = [a for a in range(2, 5)]  
print(L)
```

Aufgabe 60

```
L = [a for a in range(2, 5)]  
print(L)
```

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

Aufgabe 61

```
A = [1, 2, 3]
```

```
B = [2*x for x in A]
```

```
print(B)
```

Aufgabe 61

```
A = [1, 2, 3]
```

```
B = [2*x for x in A]
```

```
print(B)
```

```
[2, 4, 6]
```

Aufgabe 62

```
Z = [[0 for i in range(2)] for j in range(3)]  
print(Z)
```

Aufgabe 62

```
Z = [[0 for i in range(2)] for j in range(3)]  
print(Z)
```

```
[[0, 0], [0, 0], [0, 0]]
```

Aufgabe 63

```
L = [0 if i % 2 == 0 else 1 for i in range(7)]  
print(L)
```

Aufgabe 63

```
L = [0 if i % 2 == 0 else 1 for i in range(7)]  
print(L)
```

```
[0, 1, 0, 1, 0, 1, 0]
```

Aufgabe 64

```
def f(x):  
    return 2*x + 4  
  
print(f(9))
```

Aufgabe 64

22

Aufgabe 65

```
def f(x):  
    return 19  
  
print(f(3))
```

Aufgabe 65

19

Aufgabe 66

```
def f(x):  
    x + 1  
  
print(f(6))
```

Aufgabe 66

None

Aufgabe 67

```
def f(a, b):  
    return 2*a + b  
  
print(f(5, 7))
```

Aufgabe 67

17

Aufgabe 68

```
def f(a, b):  
    return 2*a + b  
  
print(f(b=5, a=7))
```

Aufgabe 68

19

Aufgabe 69

```
def f(x):  
    return 2*x+1  
  
print(f(f(f(1))))
```

Aufgabe 69

15

Aufgabe 70

```
def f(x):  
    return 2*x - 1  
def g(x):  
    return x*x  
  
print(f(5) + g(3))
```

Aufgabe 70

18

Aufgabe 71

```
def f(x):  
    a = 4  
    print(a+x)
```

```
a = 3  
f(5)  
print(a)
```

Aufgabe 71

9

3

Variablen, die in einer Funktion definiert sind, sind nur im betreffenden Funktionsrumpf (eingerückten Bereich) gültig. Gleichzeitig „überschatten“ sie globale Variablen mit gleichem Namen. Nach der Ausführung werden die lokalen Variablen wieder gelöscht und globale Variablen mit gleichem Namen werden wieder sichtbar.

Aufgabe 72

```
def f(x=2, y=1):  
    return 2*x + 3*y  
  
print(f(3))
```

Aufgabe 72

9

Aufgabe 73

```
def f(L, x):  
    L.append(x)
```

```
L = [1, 2, 3]
```

```
f(L, 4)
```

```
print(L)
```

Aufgabe 73

[1, 2, 3, 4]

Aufgabe 74

```
def f(n):  
    if n == 0:  
        return 1  
    else:  
        return f(n-1)+n  
  
print(f(4))
```

Aufgabe 74

```
def f(n):  
    if n == 0:  
        return 1  
    else:  
        return f(n-1)+n
```

```
print(f(4))
```

$$\begin{aligned}f(4) &= f(3) + 4 \\ &= (f(2) + 3) + 4 \\ &= ((f(1) + 2) + 3) + 4 \\ &= (((f(0) + 1 + 2) + 3) + 4) \quad (\text{Base Case erreicht}) \\ &= (((1 + 1) + 2) + 3) + 4 \\ &= 11\end{aligned}$$

Aufgabe 75

```
def f(n):  
    if n < 2:  
        return n  
    else:  
        return 2*f(n-2) + 1  
  
print(f(5))
```

Aufgabe 75

```
def f(n):  
    if n < 2:  
        return n  
    else:  
        return 2*f(n-2) + 1  
  
print(f(5))
```

$$\begin{aligned}f(5) &= 2 * f(3) + 1 \\ &= 2 * (2 * f(1) + 1) + 1 \quad (\text{Base Case erreicht}) \\ &= 2 * (2 * 1 + 1) + 1 \\ &= 2 * 3 + 1 = 7\end{aligned}$$

7

Aufgabe 76

```
wort = 'HALLO'  
print(wort[1])
```

Aufgabe 76

A

Aufgabe 77

```
wort = "HALLO"  
print(wort[-1])
```

Aufgabe 77

0

Aufgabe 78

```
wort = 'hundert'  
print(wort[1:4])
```

Aufgabe 78

und

Aufgabe 79

```
satz = "Was soll das?"  
print(len(satz))
```

Aufgabe 79

13

Aufgabe 80

```
wort = 'Feeler'  
wort[2] = 'h'  
print(wort)
```

Aufgabe 80

Traceback (most recent call last):

File "python-09-ueb-05.py", line 2, in <module>

```
wort[2] = 'h'
```

TypeError: 'str' object does not support item assignment

Aufgabe 81

```
a = 'abc'  
b = 'xyz'  
print(a + b)
```

Aufgabe 81

abcxyz

Aufgabe 82

```
print('a' + 2 * 'n' + 'a')
```

Aufgabe 82

anna

Aufgabe 83

```
satz = 'Das\nist\nso.'  
print(satz)
```

Aufgabe 83

Das
ist
so.

Aufgabe 84

```
satz = '''Woher  
weisst du das?'''  
print(satz)
```

Aufgabe 84

Woher
weisst du das?

Aufgabe 85

```
satz = '''Das \  
ist vielleicht \  
sinnlos!'''  
print(satz)
```

Aufgabe 85

Das ist vielleicht sinnlos!

Aufgabe 86

```
text = '15'  
print(int(text)+5)
```

Aufgabe 86

15

Aufgabe 87

```
text = '7'  
print(int(text, 2))
```

Aufgabe 87

12

Aufgabe 88

```
satz = "Das ist das Zeichen \\. "  
print(satz)
```

Aufgabe 88

Das ist das Zeichen \.

Aufgabe 89

```
text = "Sag \"Hallo\""
print(text)
```

Aufgabe 89

Sag "Hallo"

Aufgabe 90

```
a = 'Das'  
b = 'ist'  
c = 'schlecht'  
print(a, b, c, sep='.')
```

Aufgabe 90

Das.ist.schlecht

Aufgabe 91

```
wort = 'Ragusa'  
print('usa' in wort)
```

Aufgabe 91

True

Aufgabe 92

```
text = '{0} + {1} = {2}'.format(3, 4, 7)
print(text)
```

Aufgabe 92

$$3 + 4 = 7$$

Aufgabe 93

```
text = '{1} Meter kosten {0} Fr.'.format(20, 30)
print(text)
```

Aufgabe 93

30 Meter kosten 20 Fr.

Aufgabe 94

```
text = 'gut gemacht!'
print(text.capitalize())
```

Aufgabe 94

Gut gemacht!

Aufgabe 95

```
wort = 'abracadabra'  
print(wort.count('ab'))
```

Aufgabe 95

2

Aufgabe 96

```
wort = 'Mississippi'  
print(wort.find('is'))
```

Aufgabe 96

1

Aufgabe 97

```
liste = ['25', '2', '2013']  
print('.'.join(liste))
```

Aufgabe 97

25.2.2013

Aufgabe 98

```
wort = "HAMMER"  
wort.lower()  
print(wort)
```

Aufgabe 98

HAMMER

Aufgabe 99

```
wort = "Hammer"  
wort = wort.replace('m', 'c', 1)  
wort = wort.replace('m', 'k', 1)  
print(wort)
```

Aufgabe 99

Hacker

Aufgabe 100

```
satz = 'Das ist gut.'  
satz = satz.strip('.')  
print(satz)
```

Aufgabe 100

Das ist gut

Aufgabe 101

```
satz = 'Das ist gut'  
abc = satz.split(' ')  
print(abc)
```

Aufgabe 101

```
['Das', 'ist', 'gut']
```

Aufgabe 102

```
wort = "ANANAS"  
wert = wort.split('N')  
print(wert)
```

Aufgabe 102

```
['A', 'A', 'AS']
```

Aufgabe 103

```
text = '7'  
text = text.zfill(3)  
print(text)
```

Aufgabe 103

007

Aufgabe 104

```
a = 'mit'  
b = list(a)  
b.reverse()  
c = ''.join(b)  
print(c)
```

Aufgabe 104

tim

Aufgabe 105

```
D = {3: 6, -3: -9, 6: -6}
print(D[1+2])
```

Aufgabe 105

```
D = {3: 6, -3: -9, 6: -6}
print(D[1+2])
```

6

Aufgabe 106

```
D = {'a':'d', 'e':'g', 'd':'e', 'g':'a'}  
print(D[D[D['e']]])
```

Aufgabe 106

```
D = {'a':'d', 'e':'g', 'd':'e', 'g':'a'}  
print(D[D[D['e']]])
```

d

Aufgabe 107

```
D = {'a': 3, 'e': 2, 'g': -3, 'f': 4}
```

```
D.pop('a')
```

```
print(len(D))
```

Aufgabe 107

```
D = {'a': 3, 'e': 2, 'g': -3, 'f': 4}
```

```
D.pop('a')
```

```
print(len(D))
```

3

Aufgabe 108

```
D = {'b': -2, 'e': 8, 'd': 2, 'g': -6}
del D['b']
print(len(D))
```

Aufgabe 108

```
D = {'b': -2, 'e': 8, 'd': 2, 'g': -6}
del D['b']
print(len(D))
```

3

Aufgabe 109

```
D = {'c': -6, 'd': 4, 'g': -9}
D.pop('b')
print(D)
```

Aufgabe 109

```
D = {'c': -6, 'd': 4, 'g': -9}
D.pop('b')
print(D)
```

Traceback (most recent call last):

...

D.pop('b')

KeyError: 'b'

Aufgabe 110

```
D = {'b': 6, 'g': 0, 'f': 8}
```

```
E = {'b': -2, 'g': 0, 'f': 8}
```

```
D.update(E)
```

```
print(D)
```

Aufgabe 110

```
D = {'b': 6, 'g': 0, 'f': 8}
```

```
E = {'b': -2, 'g': 0, 'f': 8}
```

```
D.update(E)
```

```
print(D)
```

```
{'b': -2, 'g': 0, 'f': 8}
```

Aufgabe 111

```
D = {'e': -4, 'g': 7, 'f': 3}
for x in D.keys():
    print(D[x])
```

Aufgabe 111

```
D = {'e': -4, 'g': 7, 'f': 3}
for x in D.keys():
    print(D[x])
```

-4

7

3

Aufgabe 112

```
D = {'a': 3, 'b': -2, 'e': -3}
for y in sorted(D.values()):
    print(y)
```

Aufgabe 112

```
D = {'a': 3, 'b': -2, 'e': -3}
for y in sorted(D.values()):
    print(y)
```

-3

-2

3

Aufgabe 113

```
D = {3:2, 2:0, 4:5}
for v, w in D.items():
    print(v-w)
```

Aufgabe 113

```
D = {3:2, 2:0, 4:5}
for v, w in D.items():
    print(v-w)
```

1

2

-1

Aufgabe 114

```
A = [3, 7, 9]
```

```
B = [8, 5, 2]
```

```
D = dict(zip(A, B))
```

```
for x, y in D.items():
```

```
    print(x+y)
```

Aufgabe 114

```
A = [3, 7, 9]
```

```
B = [8, 5, 2]
```

```
D = dict(zip(A, B))
```

```
for x, y in D.items():
```

```
    print(x+y)
```

```
11
```

```
12
```

```
11
```

Aufgabe 115

```
D = {5:4, 7:8, 3:2}
E = {y:x for x, y in D.items()}
print(sorted(E))
```

Aufgabe 115

```
D = {5:4, 7:8, 3:2}
E = {y:x for x, y in D.items()}
print(sorted(E))
```

[2, 4, 8]

Aufgabe 116

Gegeben ist das folgendes Python-Modul:

```
class Rechteck:
```

```
    anzahl = 0
```

```
    def __init__(self, laenge, breite):
        self.a = laenge
        self.b = breite
        Rechteck.anzahl += 1
```

```
    def umfang(self):
        return 2*(self.a + self.b)
```

```
    def inhalt(self):
        return self.a * self.b
```

```
    def add(self, other):
        return Rechteck(self.a+other.a, self.b+other.b)
```

```
    def get anzahl():
```

Aufgabe 116

```
1  class Rechteck:
2
3      anzahl = 0
4
5      def __init__(self, laenge, breite):
6          self.a = laenge
7          self.b = breite
8          Rechteck.anzahl += 1
9
10     def umfang(self):
11         return 2*(self.a + self.b)
12
13     def inhalt(self):
14         return self.a * self.b
15
16     def add(self, other):
17         return Rechteck(self.a+other.a, self.b+other.b)
18
19     def get_anzahl():
20         return Rechteck.anzahl
```

Aufgabe 117

Welche Ausgaben macht das folgende Python-Modul?

```
class Car:

    max_speed = 120

    def __init__(self):
        self.speed = 0

    def accelerate(self, delta_v):
        self.speed = min(self.speed+delta_v,
                          Car.max_speed)

    def brake(self, delta_v):
        self.speed = max(self.speed-delta_v, 0)

    def get_speed(self):
        return self.speed

c1 = Car()
c2 = Car()
```

Aufgabe 117

```
class Car:

    max_speed = 120

    def __init__(self):
        self.speed = 0

    def accelerate(self, delta_v):
        self.speed = min(self.speed+delta_v,
                          Car.max_speed)

    def brake(self, delta_v):
        self.speed = max(self.speed-delta_v, 0)

    def get_speed(self):
        return self.speed

c1 = Car()
c2 = Car()
```

Aufgabe 118

Gegeben ist das folgende Python-Modul.

```
class Image:

    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.img = [[0 for i in range(width)] for j in
                    range(height)]

    def set_pixel(self, x, y, color=1):
        self.img[y][x] = color

    def write(self, filename):
        fd = open(filename, mode='w')
        fd.write('P1\n')
        fd.write('{0} {1}\n'.format(self.w, self.h))
        for i in range(0, self.h):
            for j in range(self.w):
                fd.write('{0} '.format(self.img[i][j]))
        fd.close()
```

Aufgabe 118

```
class Image:

    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.img = [[0 for i in range(width)] for j in
                    range(height)]

    def set_pixel(self, x, y, color=1):
        self.img[y][x] = color

    def write(self, filename):
        fd = open(filename, mode='w')
        fd.write('P1\n')
        fd.write('{0} {1}\n'.format(self.w, self.h))
        for i in range(0, self.h):
            for j in range(self.w):
                fd.write('{0} '.format(self.img[i][j]))
        fd.close()
```

Aufgabe 119

Gegeben ist das folgende Python-Modul zum Rechnen mit Brüchen.

```
from math import gcd # greatest common divisor
```

```
class Bruch:
```

```
    def __init__(self, z, n):
```

```
        t = gcd(z, n)
```

```
        self.z = z // t
```

```
        self.n = n // t
```

```
        if self.n < 0:
```

```
            self.z = -self.z
```

```
            self.n = -self.n
```

```
    def __str__(self):
```

```
        if self.n == 1:
```

```
            return '{0.z}'.format(self)
```

```
        else:
```

```
            return '{0.z}/{0.n}'.format(self)
```

Aufgabe 119

```
from math import gcd # greatest common divisor
```

```
class Bruch:
```

```
    def __init__(self, z, n):
```

```
        t = gcd(z, n)
```

```
        self.z = z // t
```

```
        self.n = n // t
```

```
        if self.n < 0:
```

```
            self.z = -self.z
```

```
            self.n = -self.n
```

```
    def __str__(self):
```

```
        if self.n == 1:
```

```
            return '{0.z}'.format(self)
```

```
        else:
```

```
            return '{0.z}/{0.n}'.format(self)
```

```
    def __add__(self, other):
```

```
        z = self.z * other.n + self.n * other.z
```

Aufgabe 120

Schreibe eine Python-Funktion `list_max(L)`, die das grösste Element der Liste `L` zurückgibt.

Hinweis: Weise dem Maximum `m` den provisorischen Wert `float('-inf')`; d. h. $-\infty$ zu und überprüfe in einer Schleife über alle Listenelemente, ob sich das jeweils aktuelle Maximum durch ein noch grösseres Element ersetzen lässt. Falls ja, ersetze es.

Aufgabe 120

```
def list_max(L):
    '''Gibt das Maximum der Elemente von L zurück.'''
    m = float('-inf')
    for x in L:
        if x > m:
            m = x
    return m

# Testcode
LoL = [ [], [14], [3,1,-4, 19, 12], [3, 3, 3, 3, 3]]
for L in LoL:
    print(L, list_max(L))
```

Aufgabe 121

Schreibe eine Python-Funktion `list_find(item, L)`, die `True` zurückgibt, wenn `item` in der Liste `L` ist und `False` sonst.

Aufgabe 121

```
def list_find(item, L):
    '''Gibt True zurück, wenn "item" Element der Liste
       "L" ist.'''
    for x in L:
        if x == item:
            return True
    return False

# Testcode
print(list_find(5, [3, 5]))
print(list_find('x', ['ab', 'xz', 'pq']))
print(list_find(False, [True, True, True]))
```

Aufgabe 122

Schreibe eine Python-Funktion `mean(L)`, welche das arithmetische Mittel der Elemente in der Liste `L` berechnet und zurückgibt.

Welcher Spezialfall muss berücksichtigt werden und wie soll man ihn verarbeiten?

Aufgabe 122

```
def mean(L): # pythonic
    '''Gibt arithmetisches Mittel der Elemente in L
        zurück'''
    if L != []:
        return sum(L)/len(L)
    else:
        return 1

def mean_classic(L):
    '''Gibt arithmetisches Mittel der Elemente in L
        zurück'''
    summe = 0
    for i in range(len(L)):
        summe = summe + L[i]
    return summe/len(L)
```

Aufgabe 123

Schreibe eine Python-Funktion `is_sorted(L)`, die `True` zurückgibt, wenn die Liste `L` aufsteigend sortiert ist; d. h. wenn mit Ausnahme des letzten Elements kein Element grösser als sein Nachfolger ist. Andernfalls soll die Funktion `False` zurückgeben.

Überlege auch, welcher Rückgabewert bei Listen der Länge 0 oder 1 sinnvoll ist.

Aufgabe 123

```
def is_sorted(L):
    '''Gibt True zurück, wenn die Elemente in L
       aufsteigend sortiert sind'''
    n = len(L)
    if n < 2:
        return True
    for i in range(0, n-1):
        if L[i] > L[i+1]:
            return False
    return True

# Testcode
LoL = [[0], [31], [-4, 3, 5, 7, 12, 29], [2, 6, 9, 15,
11]]
for L in LoL:
    print(L, is_sorted(L))
```

Aufgabe 124

Schreibe eine Python-Funktion `random_list(n, a, b)`, die eine Liste mit `n` Elementen zurückgibt, wobei jedes Element `r` eine ganze Zufallszahl mit $a \leq r \leq b$ ist.

Hinweis: Importiere dazu die Funktion `randint` aus dem `random`-Modul, die mit dem Aufruf `randint(a,b)` gleichverteilte ganze Zufallszahlen `r` mit $a \leq r \leq b$ zurückgibt.

Aufgabe 124

```
from random import randint

def random_list(n, a, b): # pythonic
    '''Gibt eine Liste mit n zufälligen Zahlen a <= x <=
        b zurück'''
    return [randint(a,b) for in range(n)]

def random_list_classic(n, a, b):
    '''Gibt eine Liste mit n zufälligen Zahlen a <= x <=
        b zurück'''
    L = []
    for x in range(a, b):
        L.append(x)
    return L

print(random_list(5, 1, 9))
print(random_list_classic(1, 9))
```

Aufgabe 125

Schreibe eine Python-Funktion `gcd(a, b)`, welche den grössten gemeinsamen Teiler (*greatest common divisor*) der beiden ganzen Zahlen `a` und `b` berechnet und zurückgibt.

Aufgabe 125

```
def gcd(a, b):  
    '''Gibt den grössten gemeinsamen Teiler von a und b  
        zurück'''  
    while b != 0:  
        a, b = b, a % b  
    return a  
  
# Testcode  
print(gcd(24,15))  
print(gcd(17, 0))  
print(gcd(0, 29))  
print(gcd(1234, 1234))
```

Aufgabe 126

Schreibe eine Python-Funktion `zero_matrix(m, n)`, welche die Nullmatrix mit `m` Zeilen und `n` Spalten zurückgibt.

Beispiel: `zero_matrix(2, 3)` gibt als Wert `[[0, 0, 0], [0, 0, 0]]` zurück.

Aufgabe 126

```
def zero_matrix(m, n):  
    '''Gibt die Nullmatrix mit m Zeilen und n Spalten  
        zurück'''  
    return [[0 for i in range(n)] for j in range(m)]
```

Aufgabe 127

Schreibe eine Python-Funktion `reverse_string(string)`, welche den String `string` in umgekehrter Zeichefolge zurückgibt.

Beispiel: `reverse_string('ABC')` \Rightarrow `'CBA'`

Aufgabe 127

```
def reverse_string(string): # pythonic
    '''Gibt die Zeichen von "string" in umgekehrter
       Reihenfolge zurück'''
    return string[::-1]
```

```
def reverse_string_classic(string):
    new_string = ''
    n = len(string)
    for i in range(0, n):
        new_string += string[n-i-1]
    return new_string
```

```
string = 'ABCDEFGF'
print(reverse_string(string))
print(reverse_string_classic(string))
```


Aufgabe 128

```
def factorial(n):  
    '''Gibt n! = n*(n-1)*...*2*1 zurück'''  
    fact = 1  
    for k in range(1,n):  
        fact *= k  
    return fact
```

```
# Testcode
```

```
for k in range(0, 20):  
    print(f'{k}! = {factorial(k)}')
```

Aufgabe 129

Schreibe eine Python-Funktion `add_matrices(A,B)`, welche die Summe der Matrizen A und B mit gleicher Dimension berechnet und zurückgibt.

Aufgabe 129

```
def matrix_add(A, B):  
    '''Gibt die Summe der Matrizen A und B zurück'''  
    m, n = len(A), len(A[0])  
    return [[A[i][j]+B[i][j] for j in range(n)] for i in  
            range(m)]
```

```
A = [[1,2], [3,4], [5,6]]
```

```
B = [[3,1], [0,2], [4,1]]
```

```
C = matrix_sum(A, B)
```

```
print(C)
```

Aufgabe 130

Schreibe eine Python-Funktion `count_letters(string)`, welche die Häufigkeit aller Zeichen in der Zeichenkette `string` bestimmt und als Python-Dictionary zurückgibt.

Aufgabe 130

```
def count_letters(string):
    '''Gibt Dictionary mit Zeichenhäufigkeiten von
       'string' zurück'''
    D = dict()
    for char in string:
        if char not in D:
            D[char] = 1
        else:
            D[char] += 1
    return D

# Testcode
D = count_letters('MISSISSIPPI')
for key in sorted(D):
    print(key, D[key])
```

Aufgabe 131

Schreibe eine Python-Funktion `count_vowels(string)`, welche die Anzahl der Vokale (A,E,I,O,U) in der Zeichenkette `string` zählt und zurückgibt.

Beispiel: `count_vowels('Eremit')` \Rightarrow 3

Aufgabe 131

```
def reverse_string(string): # pythonic
    '''Gibt die Zeichen von "string" in umgekehrter
       Reihenfolge zurück'''
    return string[::-1]
```

```
def reverse_string_classic(string):
    new_string = ''
    n = len(string)
    for i in range(0, n):
        new_string += string[n-i-1]
    return new_string
```

```
string = 'ABCDEFGG'
print(reverse_string(string))
print(reverse_string_classic(string))
```

Aufgabe 132

Schreibe eine Python-Funktion `count_words(string)`, welche die Zeichenkette `string` mit der Methode `string.split()` in seine Wörter zerlegt und danach die Häufigkeit der Wörter als Python-Dictionary zurückgibt.

Aufgabe 132

```
def count_words(string):
    '''Gibt Dictionary mit der Häufigkeit aller Wörter in
       'string' zurück'''
    wordlist = string.split() # trennt an Leerzeichen und
                               Zeilenschaltungen
    tab = dict()
    for word in wordlist:
        if word in tab:
            tab[word] += 1
        else:
            tab[word] = 1
    return tab

# Testcode
text = '''Fischers Fritz fischt frische Fische
frische Fische fischt Fischers Fritz'''
tab = count_words(text)
for key in tab:
    print(key, tab[key])
```

Aufgabe 133

Schreibe eine Python-Funktion `is_palindrome(string)`, die `True` zurückgibt, wenn `string` ein Palindrom ist; d. h. wenn `string` vorwärts und rückwärts gelesen das gleiche Wort ergibt.

Aufgabe 133

```
def is_palindrome(string):  
    '''Gibt True zurück, wenn string ein Palindrom ist  
        und False sonst.'''  
    return string == string[::-1] # pythonic!
```

```
def is_palindrome_classic(string):  
    '''Gibt True zurück, wenn string ein Palindrom ist  
        und False sonst.'''  
    n = len(string)  
    for i in range(n):  
        if string[i] != string[n-i-1]:  
            return False  
    return True
```

Testcode

```
s1, s2 = 'ANNA', 'BEN'
```

```
print(s1, is_palindrome(s1)) # => True  
print(s2, is_palindrome(s2)) # => False
```

Aufgabe 134

- (a) Schreibe eine Python-Funktion `sum_of_n(n)`, welche die Summe der ersten natürlichen Zahlen von 1 bis `n` in $O(n)$ berechnet und zurückgibt.
- (b) Schreibe eine Python-Funktion `sum_of_n_fast(n)`, welche die Summe der ersten natürlichen Zahlen von 1 bis bis `n` in $O(1)$ berechnet und zurückgibt.

Aufgabe 134

```
def sum_of_n(n):  
    '''Gibt 1 + 2 + ... + n in  $O(n)$  zurück'''  
    return sum([i for i in range(1,n+1)])  
  
def sum_of_n_fast(n):  
    '''Gibt 1 + 2 + ... + n in  $O(1)$  zurück'''  
    return n*(n+1)//2
```

Aufgabe 135

Schreibe eine Python-Funktion `transpose_matrix(A)`, welche die Transponierte A^T der Matrix A zurückgibt.

Hinweis: Die Transponierte A^T einer Matrix A ist die Matrix, bei der die i -te Zeilen von A zur j -ten Spalte von A^T wird.

$$\text{Beispiel: } A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Aufgabe 135

```
def transpose_matrix(A):  
    '''Gibt die Transponierte  $A^T$  von A zurück'''  
    m, n = len(A), len(A[0])  
    return [[A[i][j] for i in range(m)] for j in range(n)]  
  
A = [[1,2], [3,4], [5,6]]  
B = transpose_matrix(A)  
rint(B) # => [[1,3,5],[2,4,6]]
```

Aufgabe 136

Schreibe eine Python-Funktion `write_integers(n, filename)`, welche die ganzen Zahlen $1, 2, \dots, n$ zeilenweise in die Datei mit dem Namen `filename` schreibt.

Aufgabe 136

```
def write_integers(n, filename):
    '''Schreibt die ersten n natürlichen Zahlen in eine
       Datei'''
    fd = open(filename, mode='w')
    for i in range(1, n+1):
        fd.write(f'{i}\n')
    fd.close()

write_integers(50, 'integers.txt')
```

Aufgabe 137

Schreibe eine Python-Funktion `add_floats_in_file(filename)`, welche die Summe der Zahlen in dieser Datei zurückgibt, wobei in jeder Zeile der Datei eine Gleitkommazahl steht.

Aufgabe 137

```
def add_floats_in_file(filename):  
    '''Addiert alle Gleitkommazahlen in der Datei  
       "filename".'''  
    fd = open(filename, mode='r')  
    s = 0  
    for line in fd:  
        s += float(line)  
    fd.close()  
    return s  
  
# Testcode  
summe = add_floats_in_file('messwerte.txt')  
print(summe)
```

Aufgabe 138

Schreibe eine Python-Funktion `fibonacci(n)`, welche die Liste mit den ersten n Fibonacci-Zahlen (1, 1, 2, 3, 5, 8, 13, ...) erzeugt und zurückgibt.

Aufgabe 138

```
def fibonacci(n):
    '''Gibt die Liste der ersten n Fibonacci-Zahlen
        zurück'''
    F = []
    if n > 0:
        F.append(1)
    if n > 1:
        F.append(1)
    for i in range(2,n+1):
        F.append(F[-2] + F[-1])
    return F

# Testcode
for k in range(1, 7):
    print(k, fibonacci(k))
```