### Aufgabe 1.1

Was für eine Art von Programm braucht es, um ein Python-Programm zu schreiben?

### Aufgabe 1.2

Was für eine Art von Programm braucht es, um ein Python-Programm auszuführen?

### Aufgabe 1.3

Zähle die drei grundsätzlichen Typen von Programmierfehlern auf und nenne jeweils ein Beispiel dazu.

### Aufgabe 1.4

Welche Dateiendung haben Python-Programme normalerweise?

### Aufgabe 2.1

```
print(3+2*9-1)
```

### Aufgabe 2.2

```
print(8+9*4-6)
```

### Aufgabe 2.3

```
print(7+4*1-8)
```

### Aufgabe 2.4

```
print(32 / 8)
```

### Aufgabe 2.5

```
print(26 % 7)
```

### Aufgabe 2.6

```
print(20 // 8)
```

### Aufgabe 2.7

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

### Aufgabe 3.1

```
a = 4
a = a + 6
a = a * 2
a = a - 9
print(a)
```

### Aufgabe 3.2

```
a = 3
b = a + 9
a = b + a - 8
print(a)
```

### Aufgabe 3.3

```
a, b, c = 9, 2, 8
b, c, a = a, b, c
print(c)
```

### Aufgabe 3.4

```
b = 5
b += 4
b *= -1
print(b)
```

### Aufgabe 3.5

Was gibt das folgenden Python-Programm nach der Ausführung von Zeile 3 auf der Shell aus, wenn die Benutzerin nach der Eingabeaufforderung auf die Taste mit der Ziffer 2 und anschliessend auf die ENTER-Taste drückt?

```
x = input('Eingabe: ')
y = 5*int(x)
print(y)
```

### Aufgabe 3.6

Sind die folgenden Bezeichner für Variablen syntaktisch korrekt?

(a) `anzahl_personen`

(b) `4you`

(c) `_1234`

(d) `lieferung-mai-2022`

(e) `else`

(f) `Or`

### Aufgabe 3.7

Was gibt das folgende Python-Programm auf der Shell aus?

```
print('{}{}{}{}'.format(3, 7, 'a', 'x'))
```

### Aufgabe 4.1

```
a=4
b=3
print(a != b)
```

### Aufgabe 4.2

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

### Aufgabe 4.3

```
print(False or False)
```

### Aufgabe 4.4

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

### Aufgabe 4.5

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

### Aufgabe 4.6

```
print(not(3 < 2) or 4 == 2)
```

### Aufgabe 4.7

```
print(7 < 4 <= 7)
```

### Aufgabe 5.1

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

### Aufgabe 5.2

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

### Aufgabe 5.3

```
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 5.4

```
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)
```

Wenn nichts anderes steht, ist die Ausgabe des Programmfragments zu notieren.

### Aufgabe 6.1

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

### Aufgabe 6.2

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

### Aufgabe 6.3

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

### Aufgabe 6.4

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

### Aufgabe 6.5

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

### Aufgabe 6.6

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

### Aufgabe 6.7

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

### Aufgabe 6.8

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

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

### Aufgabe 6.9

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

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

### Aufgabe 6.10

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

### Aufgabe 6.11

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

### Aufgabe 6.12

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

### Aufgabe 7.1

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

### Aufgabe 7.2

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

### Aufgabe 7.3

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

### Aufgabe 7.4

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

### Aufgabe 7.5

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

### Aufgabe 7.6

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

### Aufgabe 7.7

```
L = [5, 2, 3]
L.append(7)
print(L)
```

### Aufgabe 7.8

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

### Aufgabe 7.9

```
L = [7, 9, 5, 8, 2]
L.pop()
print(L)
```

### Aufgabe 7.10

```
L = [7, 9, 5, 8, 2]
x = L.pop()
print(x)
```

### Aufgabe 7.11

```
L = [7, 9, 5, 8, 2]
x = L.pop(2)
print(x)
```

### Aufgabe 7.12

```
A = [9, 3, 2]
B = [4, 1]
print(A + B)
```

### Aufgabe 7.13

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

### Aufgabe 7.14

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

### Aufgabe 7.15

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

### Aufgabe 7.16

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

### Aufgabe 7.17

```
A = [6, 7, 2]
B = A
B[1] = 9
print(A)
```

### Aufgabe 7.18

```
A = [6, 7, 2]
B = A[:]
B[1] = 9
print(A)
```

### Aufgabe 7.19

```
L = [6, 1, 2, 7, 9]
L.reverse()
print(L)
```

### Aufgabe 7.20

```
L = [6, 1, 2, 7, 9]
L.sort()
print(L)
```

### Aufgabe 7.21

```
L = [6, 4, 2, 8]
print(sum(L))
```

### Aufgabe 7.22

```
[y, x, z] = [5, 3, 1]
print(x)
```

### Aufgabe 7.23

```
L = [9, 2, 4, 1, 8]
s = 0
for e in L:
    s += e
print(s)
```

### Aufgabe 7.24

```
L = [9, 2, 4, 1, 8]
s = 0
for i in range(0, len(L)):
```

```
    s += L[i]
print(s)
```

### Aufgabe 7.25

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

### Aufgabe 7.26

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

### Aufgabe 7.27

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

### Aufgabe 7.28

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

### Aufgabe 7.29

```
A = [1, 2, 3]
B = [2*x for x in A]
print(B)
```

### Aufgabe 7.30

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

### Aufgabe 7.31

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

Wenn nichts anderes steht, ist die Ausgabe des Python-Programmfragments anzugeben.

### Aufgabe 8.1

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

print(f(9))
```

### Aufgabe 8.2

```
def f(x):
    return 19

print(f(3))
```

### Aufgabe 8.3

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

print(f(6))
```

### Aufgabe 8.4

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

print(f(5, 7))
```

### Aufgabe 8.5

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

print(f(b=5, a=7))
```

### Aufgabe 8.6

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

print(f(f(f(1))))
```

### Aufgabe 8.7

```
def f(x):
    return 2*x - 1

def g(x):
    return x*x

print(f(5) + g(3))
```

### Aufgabe 8.8

```
def f():
    return 9

print(f())
```

### Aufgabe 8.9

```
def f(x):
    if x < 0:
        return 1
    else:
        return 0

print(f(8))
```

### Aufgabe 8.10

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

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

### Aufgabe 8.11

Schreibe eine syntaktisch korrekte Funktion mit dem Namen `dreieckUmfang(...)`, die aus den drei Seitenlängen $a$, $b$ und $c$ den Dreiecksumfang berechnet und als Wert zurückgibt.

### Aufgabe 8.12

Schreibe eine syntaktisch korrekte Funktion mit dem Namen `trapezInhalt(...)`, die aus den gegebene parallelen Seiten $a$ und $c$ sowie der Höhe $h$ den Flächeninhalt des Tapzes berechnet und als Wert zurückgibt.

*Hinweis:* $A_{\text{Trapez}} = \dfrac{a+c}{2} \cdot h$

### Aufgabe 8.13

```
def f(x, y, z):
    x * y + z

print(f(2, 3, 4))
```

### Aufgabe 8.14

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

print(f(3))
```

### Aufgabe 8.15

```
def f(x):
    return 7

print(f(8))
```

### Aufgabe 8.16

```
def f(x):
    return 2*x - 1

print(f(f(2)))
```

### Aufgabe 8.17

```
def f(x, y):
    return x + y

def g(a, b):
    return a * b

print(f(1, 2) + g(3, 4))
```

### Aufgabe 8.18

```
def f(x):
    y = 2*x

y = 8
f(3)
print(y)
```

### Aufgabe 8.19

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

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

### Aufgabe 8.20

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

print(f(4))
```

### Aufgabe 8.21

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

print(f(5))
```

### Aufgabe 8.22

```python
def f(x, y, z):
    return x+y, z-y

a, b = f(7, 2, 5)
print(a+b)
```

### Aufgabe 9.1

```
s = '''Das ist
ein Text'''
print(s)
```

### Aufgabe 9.2

```
s = 'Das ist \nWahnsinn'
print(s)
```

### Aufgabe 9.3

```
s = 'C\'est la vie!'
print(s)
```

### Aufgabe 9.4

```
s = 'A\\B'
print(s)
```

### Aufgabe 9.5

```
s = 'Hund'
print(s[1:])
```

### Aufgabe 9.6

```
print('A' + 2*'B' + 'A')
```

### Aufgabe 9.7

```
s = 'Was ist das?'
print(len(s))
```

### Aufgabe 9.8

```
s = "hallo"
print(list(s))
```

### Aufgabe 9.9

```
print(ord('A'))
```

### Aufgabe 9.10

```
print(chr(66))
```

### Aufgabe 9.11

```
s = "Ananas"
print(s.count('a'))
```

### Aufgabe 9.12

```
s = 'Hallo'
s = s.lower()
print(s)
```

### Aufgabe 9.13

```
s = 'ch'
s.upper()
print(s)
```

### Aufgabe 9.14

```
s = 'abcde'
s = s.replace('a', 'b')
print(s)
```

### Aufgabe 9.15

```
L = ['x', 'y', 'z']
s = '+'.join(L)
print(s)
```

### Aufgabe 9.16

```
s = 'teller'
print(s.split('e'))
```

### Aufgabe 9.17

```
s = 'abcdxyz'
s = s.strip('abyz')
print(s)
```

### Aufgabe 9.18

```
s = 'abcxyz'
s = s.lstrip('abyz')
print(s)
```

### Aufgabe 9.19

```
s = 'abcxyz'
s = s.rstrip('abyz')
print(s)
```

### Aufgabe 9.20

```
s = 'a{}pb{}m'.format(3, 'e')
print(s)
```

### Aufgabe 9.21

```
s = 't{1}k{0}w{1}'.format(3, 'e')
print(s)
```

### Aufgabe 9.22

```
print(int("15" + "4") + 3)
```

### Aufgabe 9.23

```
print(float('15') + 3)
```

### Aufgabe 9.24

```
s = 'abracadabra'
print(s.find('ra'))
```

### Aufgabe 10.1

```
print('{0}/{1}'.format(3,4))
```

### Aufgabe 10.2

```
print('({0},{1})+({1},{0})'.format(2,7))
```

### Aufgabe 10.3

```
print('{},{},{}'.format(3,1,5))
```

### Aufgabe 10.4

```
L = [5,3,1,8,7,6]
print('{0[4]},{0[1]}'.format(L))
```

### Aufgabe 10.5

```
print('{0:+}'.format(123456))
print('{0:+}'.format(-123456))
print('{0: }'.format(123456))
print('{0: }'.format(-123456))
```

### Aufgabe 10.6

```
print('{0:*>7}'.format('a'))
print('{0:*<7}'.format('b'))
print('{0:*^7}'.format('c'))
```

### Aufgabe 10.7

```
print('{0:,}'.format(100000000))
print('{0:_}'.format(100000000))
```

### Aufgabe 10.8

```
a = 13
print('{0:b};{0:#b}'.format(a))
print('{0:o};{0:#o}'.format(a))
print('{0:d}'.format(a))
print('{0:x};{0:#x}'.format(a))
print('{0:X};{0:#X}'.format(a))
```

### Aufgabe 10.9

```
print('{0:.3f}'.format(2/3))
print('{0:.4f}'.format(2/3))
print('{0:.3e}'.format(2/3))
```

```
print('{0:.4e}'.format(2/3))
print('{0:.4e}'.format(2000/3))
```

### Aufgabe 10.10

```
print('079','999','99','99', sep='-')
```

### Aufgabe 10.11

```
print('abc', end='*')
print('uvw')
```

### Aufgabe 10.12

Was steht nach der Ausführung des folgenden Programms in der Datei `myfile.txt`?

```
fd = open('myfile.txt', mode='w')
fd.write('abc')
fd.write('xyz')
fd.close()
```

### Aufgabe 10.13

Welche Ausgabe macht das folgende Programm

```
fd = open('data.txt', mode='r')
L = []
for record in fd:
    L.append(int(record))
fd.close()
print(L)
```

für die Datei `data.txt` mit folgendem Inhalt:

```
5
8
7
9
```

### Aufgabe 11.1

```
D = {5: 7, 3: 5, 7: 3}
print(D[7])
```

### Aufgabe 11.2

```
D = {'a': 'c', 'b': 'a', 'c': 'b'}
print(D[D['a']])
```

### Aufgabe 11.3

```
D = {'a': -6, 'e': -2, 'd': -1, 'g': 5}
print(len(D))
```

### Aufgabe 11.4

```
D = {'u': 3, 'x': 2, 's': -3, 'p': 4}
print(D.pop('s'))
```

### Aufgabe 11.5

```
D = {'c': -2, 'e': 3, 'a': -1, 'h': 9}
for x in sorted(D.values()):
    print(x)
```

### Aufgabe 11.6

```
D = {'k': 4, 'm': 9, 'u': 8}
D.pop('k')
print(len(D))
```

### Aufgabe 11.7

```
personal = {
    987: {'name': 'Weber', 'gehalt': 80000},
    209: {'name': 'Schmid', 'gehalt': 65000},
    566: {'name': 'Huber',  'gehalt': 70000}
}
print(personal[209]['gehalt'])
```

### Aufgabe 11.8

```
D = {'a': 6, 'b': 0, 'c': 8}
E = {'b': 5, 'c': 4, 'd': 9}
E.update(D)
print(E['c'])
```

### Aufgabe 11.9

```
D = {'a': 7, 'b': 2, 'c': 4}
E = D
E['b'] = 99
print(D['b'])
```

### Aufgabe 11.10

```
D = {'u': 2, 'c': 8, 'm': 3}
for x in sorted(D):
    print(x)
```

### Aufgabe 11.11

```
D = {'a': 5, 'd': 4, 'd': 7}
E = dict()
for (x, y) in D.items():
    E[y] = x
print(sorted(E.keys()))
```

### Aufgabe 11.12

```
D = {'e': 4, 'h': 6, 'g': 5, 'f': 3}
if 'g' not in D:
    D['g'] = 1
else:
    D['g'] += 1
print(D['g'])
```

### Aufgabe 11.13

```
D = {'cxpks': 39480903, 'dusqf': 43892011, 'hbwed': 88314086}
D['cxpkt'] = D['cxpks']
del D['cxpks']
print(len(D))
```

*Hinweis:* Sofern Mengen vor der Ausgabe nicht sortiert werden, können Elemente von Mengen können in beliebiger Reihenfolge aufgezählt werden.

### Aufgabe 12.1

```
S = {5, 7, 3, 1, 5, 4}
print(len(S))
```

### Aufgabe 12.2

```
A = {1, 2, 4, 7}
B = {2, 4, 6}
print(B.issubset(A))
```

### Aufgabe 12.3

```
A = {1, 2, 4, 7}
B = {2, 3, 5}
print(A.union(B))
```

### Aufgabe 12.4

```
A = {1, 2, 4, 7}
B = {2, 3, 5}
print(A.intersection(B))
```

### Aufgabe 12.5

```
A = {1, 2, 4, 7}
B = {2, 3, 5}
print(A.difference(B))
```

### Aufgabe 12.6

```
A = {1, 5, 7}
B = {2, 4, 8, 9}
print(A.isdisjoint(B))
```

### Aufgabe 12.7

```
A = {1, 5, 7}
A.add(4)
print(A)
```

### Aufgabe 12.8

```
A = {1, 5, 7}
A.discard(5)
print(A)
```

### Aufgabe 12.9

```
A = {1, 5, 7}
print(sum(A))
```

### Aufgabe 12.10

```
A = {3, 2, 1, 7}
print(sorted(A))
```

### Aufgabe 12.11

```
A = {3, 2, 1, 7}
p = 1
for a in A:
    p *= a
print(p)
```

### Aufgabe 12.12

```
M = set([1, 3, 1, 3])
print(sorted(M))
```

### Aufgabe 12.13

```
M = set('PYTHON')
print(sorted(M))
```

Die Übungen beziehen sich auf die folgende Python-Datei

```
a = 3

def f(x):
    return 2*x
```

xyz.py

Sind die Codefragmente korrekt? Wenn ja, welche Ausgabe machen sie?

### Aufgabe 13.1

```
import xyz

print(a)
```

### Aufgabe 13.2

```
import xyz.py

print(xyz.a)
```

### Aufgabe 13.3

```
from xyz import f

def f(x):
    return 3*x

print(f(10))
```

### Aufgabe 13.4

```
import xyz

print(f(5))
```

### Aufgabe 13.5

```
import xyz as u

print(u.f(5))
```

### Aufgabe 13.6

```
from xyz import *

def f(x):
    return 4*x

print(f(10))
```

### Aufgabe 13.7

```
from xyz import f as g, a as b

a = 3

def f(x):
    return 4*x

print(g(a))
```

### Aufgabe 13.8

```
import math

print(math.sqrt(25))
```

### Aufgabe 13.9

```
import math

print('{0:.2f}'.format(math.pi))
```

### Aufgabe 15.1

Erkläre die folgenden Begriffe der objektorientierten Programmierung.

(a) Klasse

(b) Instanz (oder Objekt)

(c) Objekteigenschaft

(d) Objektmethode

(e) Klasseneigenschaft

(f) Klassenmethode

(g) Konstruktor

### Aufgabe 15.2

```
1  class MyClass:
2
3      def __init__(self, a, b):
4          self.a = a
5          self.b = b
6
7      def d(self, c):
8          return (self.a + self.b + c)
9
10 x = MyClass(3, 4)
11 print(x.d(5))
```

### Aufgabe 15.3

```
class Aufgabe():

    def __init__(self, a=3, b=2):
        self.x = b
        self.y = a

    def __str__(self):
        return '({0.y}, {0.x})'.format(self)

print(Aufgabe(4))
```

### Aufgabe 15.4

```
class Vektor():

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __str__(self):
        return '({0.x},{0.y})'.format(self)

    def __add__(u, v):
        return  Vektor(u.x+v.x, u.y+v.y)

    def __sub__(u, v):
        return Vektor(u.x-v.x, u.y-v.y)

    def __rmul__(u, k):
        return Vektor(k*u.x, k*u.y)

    def betrag(self):
        return (self.x**2 + self.y**2)**0.5

a = Vektor(2,3)
b = Vektor(4,7)
c = 2*a-b
print(c.betrag())
```

### Aufgabe 15.5

Welche Ausgabe macht das folgende Programm?

```
class Parent():
    def __init__(self, a):
        self.a = a
    def m(self, x):
        return (x*self.a)

class Child(Parent):
    def __init__(self, a, b):
        super().__init__(a)
        self.b = b
    def m(self, y):
        return (self.a + self.b + super().m(y))

c = Child(2,3)
print(c.m(10))
```

### Aufgabe 15.6

Implementiere die Klasse `Lampe`, welche das Verhalten einer Lampe aufgrund des folgenden Klassendiagramms modelliert.

| `Lampe` |
|---|
| `zustand: bool` |
| `Lampe()`<br>`schalten()`<br>`str(): str` |

Hinweise:

- Der Konstruktor erzeugt eine ausgeschaltete Lampe.
- Die Instanzvariable `zustand` kann nur die Werte `True` oder `False` annehmen.
- Die Spezialmethode `str()` wird mit Hilfe von `__str__` implementiert und hat je nach Zustand der Lampe den Rückgabewert `'Lampe ein'` bzw. `'Lampe aus'`.

### Aufgabe 15.7

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()

c1.accelerate(50)
c1.accelerate(30)
c1.brake(40)

c2.accelerate(40)
c2.brake(50)

print(c1.get_speed())
print(c2.get_speed())
```

### Aufgabe 15.8

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()

I = Image(3, 3)
I.set_pixel(0, 0)
I.set_pixel(1, 1)
I.set_pixel(2, 2)
I.write('test.pbm')
```

(a) Beschreibe, was dieses Modul macht.

(b) Gibt es Daten aus? Wenn ja, welche Daten und in welcher Form?

### Aufgabe 15.9

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

```
1  from math import gcd # greatest common divisor
2
3  class Bruch:
4
5      def __init__(self, z, n):
6          t = gcd(z, n)
7          self.z = z // t
8          self.n = n // t
9          if self.n < 0:
10             self.z = -self.z
11             self.n = -self.n
12
13     def __str__(self):
14         if self.n == 1:
15             return '{0.z}'.format(self)
16         else:
17             return '{0.z}/{0.n}'.format(self)
18
19     def __add__(self, other):
20         z = self.z * other.n + self.n + other.z
21         n = self.n * other.n
22         return Bruch(z, n)
23
24 a = Bruch(3,9)
25 b = Bruch(1,6)
26 c = a + b
27
28 print(a)
29 print(c)
```

*Hinweise:* Die Spezialmethode `__str__` überschreibt die `str()`-Methode und wird implizit bei der Ausgabe mit `print(...)` ausgeführt. Die Spezialmethode `__add__` überschreibt den Python-Operator „`+`“, wobei hier `self` für den linken und `other` für den rechten Operator steht.

(a) Beschreibe die einzelnen Teile des Moduls so genau wie möglich.

(b) Welche Ausgaben macht das Modul?

(c) Ergänze das Modul mit einer Methode zum Multiplizieren von Brüchen, indem du den Operator `__mul__` überschreibst.

### Aufgabe 15.10

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():
        return Rechteck.anzahl

r1 = Rechteck(2,5)
r2 = Rechteck(4,3)
r3 = r1.add(r2)

print(r1.umfang())
print(r2.inhalt())
print(r3.b)
print(Rechteck.get_anzahl())
```

(a) Welche Ausgabe(n) macht das Programm?

(b) Erkläre die folgenden Begriffe anhand des obigen Quellcodes.

- Klasse
- Instanz (oder Objekt)
- Objekteigenschaft (oder Objektattribut)
- Objektmethode
- Klasseneigenschaft (oder Klassenattribut)
- Konstruktor