Python

Prüfungsvorbereitung

Aufgabe 1.1

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

Aufgabe 1.1

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

Man benötigt einen Texteditor. Das ist ein Programm mit dem man Texte schreiben und bearbeiten kann.

Aufgabe 1.2

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

Aufgabe 1.2

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

einen Python-Intepreter (also ein Programm, das wiederum ein Programm ausführt)

Aufgabe 1.3

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

Aufgabe 1.3

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

- Syntax-Fehler
  - eine Variable falsch geschrieben,
  - Code nicht 4 Zeichen eingerückt,
  - Doppelpunkt vergessen,
  - ...
- Laufzeit-Fehler
  - Division durch Null,
  - eine Datei, die eingelesen werden soll, ist nicht vorhanden,
  - ...
- Logische Fehler
  - Fehler in einer Berechnung ,
  - Bedingungen nicht korrekt formuliert,
  - falsche Reihenfolge von Code,
  - ...

Aufgabe 1.4

Welche Dateiendung haben Python-Programme normalerweise?

Aufgabe 1.4

Welche Dateiendung haben Python-Programme normalerweise?

Die Endung `.py`

Aufgabe 2.1

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

Aufgabe 2.1

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

20

Aufgabe 2.2

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

Aufgabe 2.2

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

```
38
```

Aufgabe 2.3

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

Aufgabe 2.3

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

3

Aufgabe 2.4

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

Aufgabe 2.4

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

```
4.0
```

Aufgabe 2.5

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

Aufgabe 2.5

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

```
5
```

Aufgabe 2.6

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

Aufgabe 2.6

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

2

Aufgabe 2.7

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

Aufgabe 2.7

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

```
256
```

Aufgabe 3.1

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

Aufgabe 3.1

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

```
1  11
```

Aufgabe 3.2

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

Aufgabe 3.2

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

```
1 7
```

Aufgabe 3.3

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

Aufgabe 3.3

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

```
1  2
```

Aufgabe 3.4

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

Aufgabe 3.4

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

```
1  -9
```

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?

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

Aufgabe 3.5

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

Die Ziffer 2 wird als Zeichenkette ('2') in der Variablen mit dem Namen `x` gespeichert. Vor der Multiplikation mit der ganzen Zahl 5 wird die Zeichenkette '2' in die entsprechende Zahl 2 umgewandelt. Somit hat das Produkt den Wert 10, der in der Variablen mit dem Namen `y` gespeichert wird. Dieser Wert 10 wird in Zeile 3 auf der Shell ausgegeben.

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 3.6

(a) `anzahl_personen` ja

(b) `4you` nein (Ziffer an erster Stelle verboten)

(c) `_1234` ja

(d) `lieferung-mai-2022` nein (enthält Sonderzeichen -)

(e) `else` nein (Python-Schlüsselwort)

(f) `Or` ja (`Or` $\neq$ `or`)

Aufgabe 3.7

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

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

Aufgabe 3.7

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

Aufgabe 3.7

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

```
37ax
```

Aufgabe 4.1

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

Aufgabe 4.1

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

```
1 True
```

Aufgabe 4.2

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

Aufgabe 4.2

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

```
1 False
```

Aufgabe 4.3

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

Aufgabe 4.3

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

```
1 False
```

Aufgabe 4.4

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

Aufgabe 4.4

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

```
1 True
```

Aufgabe 4.5

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

Aufgabe 4.5

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

```
1 False
```

Aufgabe 4.6

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

Aufgabe 4.6

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

```
1  True
```

Aufgabe 4.7

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

Aufgabe 4.7

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

```
1 False
```

Aufgabe 5.1

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

Aufgabe 5.1

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

```
11
```

Aufgabe 5.2

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

Aufgabe 5.2

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

```
12
```

Aufgabe 5.3

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

Aufgabe 5.3

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

```
13
```

Aufgabe 5.4

```
1  b = 7
2  if b == 7:
3      if b > 5:
4          b = b + 1
5      else:
6          b = b + 2
7  else:
8      if b >= 4:
9          b = b + 3
10     else:
11         b = b + 4
12 print(b)
```

Aufgabe 5.4

```
1  b = 7
2  if b == 7:
3      if b > 5:
4          b = b + 1
5      else:
6          b = b + 2
7  else:
8      if b >= 4:
9          b = b + 3
10     else:
11         b = b + 4
12 print(b)
```

```
8
```

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

Aufgabe 6.1

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

Aufgabe 6.1

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

```
6
8
```

Aufgabe 6.2

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

Aufgabe 6.2

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

```
1
5
9
```

Aufgabe 6.3

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

Aufgabe 6.3

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

```
10
9
8
```

Aufgabe 6.4

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

Aufgabe 6.4

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

```
3
7
```

Aufgabe 6.5

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

Aufgabe 6.5

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

```
1
3
7
15
```

Aufgabe 6.6

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

Aufgabe 6.6

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

```
15
```

Aufgabe 6.7

```
1  i = 4
2  s = 0
3  while (s < 20):
4      s = s + i
5      i = i + 3
6      print(s)
```

Aufgabe 6.7

```
1  i = 4
2  s = 0
3  while (s < 20):
4      s = s + i
5      i = i + 3
6      print(s)
```

```
4
11
21
```

Aufgabe 6.8

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

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

Aufgabe 6.8

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

Ja, denn die die Abbruchbedingung kann nicht erreicht werden.

Aufgabe 6.9

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

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

Aufgabe 6.9

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

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

Aufgabe 6.10

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

Aufgabe 6.10

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

```
4
-7
```

Aufgabe 6.11

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

Aufgabe 6.11

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

```
1
2
4
5
7
8
10
```

Aufgabe 6.12

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

Aufgabe 6.12

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

```
5
6
6
7
```

Aufgabe 7.1

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

Aufgabe 7.1

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

```
1
```

Aufgabe 7.2

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

Aufgabe 7.2

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

```
9
```

Aufgabe 7.3

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

Aufgabe 7.3

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

```
7
```

Aufgabe 7.4

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

Aufgabe 7.4

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

```
6
```

Aufgabe 7.5

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

Aufgabe 7.5

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

```
3
```

Aufgabe 7.6

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

Aufgabe 7.6

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

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

Aufgabe 7.7

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

Aufgabe 7.7

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

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

Aufgabe 7.8

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

Aufgabe 7.8

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

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

Aufgabe 7.9

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

Aufgabe 7.9

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

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

Aufgabe 7.10

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

Aufgabe 7.10

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

```
2
```

Aufgabe 7.11

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

Aufgabe 7.11

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

```
5
```

Aufgabe 7.12

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

Aufgabe 7.12

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

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

Aufgabe 7.13

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

Aufgabe 7.13

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

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

Aufgabe 7.14

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

Aufgabe 7.14

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

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

Aufgabe 7.15

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

Aufgabe 7.15

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

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

Aufgabe 7.16

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

Aufgabe 7.16

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

```
[4, 3]
```

Aufgabe 7.17

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

Aufgabe 7.17

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

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

Aufgabe 7.18

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

Aufgabe 7.18

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

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

Aufgabe 7.19

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

Aufgabe 7.19

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

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

Aufgabe 7.20

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

Aufgabe 7.20

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

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

Aufgabe 7.21

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

Aufgabe 7.21

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

```
20
```

Aufgabe 7.22

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

Aufgabe 7.22

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

```
3
```

Aufgabe 7.23

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

Aufgabe 7.23

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

```
24
```

Aufgabe 7.24

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

Aufgabe 7.24

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

```
24
```

Aufgabe 7.25

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

Aufgabe 7.25

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

```
2
```

Aufgabe 7.26

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

Aufgabe 7.26

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

```
...
    L[3] = 8
    ~^^^
TypeError: 'tuple' object does not support item assignment
```

Aufgabe 7.27

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

Aufgabe 7.27

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

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

Aufgabe 7.28

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

Aufgabe 7.28

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

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

Aufgabe 7.29

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

Aufgabe 7.29

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

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

Aufgabe 7.30

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

Aufgabe 7.30

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

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

Aufgabe 7.31

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

Aufgabe 7.31

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

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

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

Aufgabe 8.1

```
1  def f(x):
2      return 2*x + 4
3
4  print(f(9))
```

Aufgabe 8.1

22

Aufgabe 8.2

```
1  def f(x):
2      return 19
3
4  print(f(3))
```

Aufgabe 8.2

19

Aufgabe 8.3

```
1  def f(x):
2      x + 1
3
4  print(f(6))
```

Aufgabe 8.3

```
None
```

Aufgabe 8.4

```
1  def f(a, b):
2      return 2*a + b
3
4  print(f(5, 7))
```

Aufgabe 8.4

17

Aufgabe 8.5

```
1  def f(a, b):
2      return 2*a + b
3
4  print(f(b=5, a=7))
```

Aufgabe 8.5

19

Aufgabe 8.6

```
1  def f(x):
2      return 2*x+1
3
4  print(f(f(f(1))))
```

Aufgabe 8.6

15

Aufgabe 8.7

```
1  def f(x):
2      return 2*x - 1
3
4  def g(x):
5      return x*x
6
7  print(f(5) + g(3))
```

Aufgabe 8.7

18

Aufgabe 8.8

```
1  def f():
2      return 9
3
4  print(f())
```

Aufgabe 8.8

9

Aufgabe 8.9

```
1  def f(x):
2      if x < 0:
3          return 1
4      else:
5          return 0
6
7  print(f(8))
```

Aufgabe 8.9

0

Aufgabe 8.10

```
1  def f(x):
2      a = 4
3      print(a+x)
4
5  a = 3
6  f(5)
7  print(a)
```

Aufgabe 8.10

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 gelichem Namen werden wieder sichtbar.

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 8.11

```
1  def dreieckUmfang(a, b, c):
2      return a + b + c
```

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: $A_{\text{Trapez}} = \frac{a+c}{2} \cdot h$

Aufgabe 8.12

```
1  def trapezInhalt(a, c, h):
2      m = (a+c)/2
3      return m*h
```

Aufgabe 8.13

```
1  def f(x, y, z):
2      x * y + z
3
4  print(f(2, 3, 4))
```

Aufgabe 8.13

```
None
```

Aufgabe 8.14

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

Aufgabe 8.14

9

Aufgabe 8.15

```
1  def f(x):
2      return 7
3
4  print(f(8))
```

Aufgabe 8.15

7

Aufgabe 8.16

```
1  def f(x):
2      return 2*x - 1
3
4  print(f(f(2)))
```

Aufgabe 8.16

5

Aufgabe 8.17

```
1  def f(x, y):
2      return x + y
3
4  def g(a, b):
5      return a * b
6
7  print(f(1, 2) + g(3, 4))
```

Aufgabe 8.17

15

Aufgabe 8.18

```
1  def f(x):
2      y = 2*x
3
4  y = 8
5  f(3)
6  print(y)
```

Aufgabe 8.18

8

Aufgabe 8.19

```
1  def f(L, x):
2      L.append(x)
3
4  L = [1, 2, 3]
5  f(L, 4)
6  print(L)
```

Aufgabe 8.19

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

Aufgabe 8.20

```
1  def f(n):
2      if n == 0:
3          return 1
4      else:
5          return f(n-1)+n
6
7  print(f(4))
```

Aufgabe 8.20

```
1  def f(n):
2      if n == 0:
3          return 1
4      else:
5          return f(n-1)+n
6
7  print(f(4))
```

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

```
11
```

Aufgabe 8.21

```
1  def f(n):
2      if n < 2:
3          return n
4      else:
5          return 2*f(n-2) + 1
6
7  print(f(5))
```

Aufgabe 8.21

```
1  def f(n):
2      if n < 2:
3          return n
4      else:
5          return 2*f(n-2) + 1
6
7  print(f(5))
```

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

7

Aufgabe 8.22

```
1  def f(x, y, z):
2      return x+y, z-y
3
4  a, b = f(7, 2, 5)
5  print(a+b)
```

Aufgabe 8.22

12

Aufgabe 9.1

```
1 s = '''Das ist
2 ein Text'''
3 print(s)
```

Aufgabe 9.1

```
1 s = '''Das ist
2 ein Text'''
3 print(s)
```

```
Das ist
ein Text
```

Aufgabe 9.2

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

Aufgabe 9.2

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

```
Das ist
Wahnsinn
```

Aufgabe 9.3

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

Aufgabe 9.3

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

```
C'est la vie!
```

Aufgabe 9.4

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

Aufgabe 9.4

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

```
A\B
```

Da der Backslash zum Maskieren der Stringbegrenzungszeichen ("..." und '...') sowie für die Bildung von Steuerzeichen (\n) verwendet wird, kann er nicht direkt in einer Zeichenkette auftreten und muss seinerseits durch einen zweiten Backslash maskiert werden.

Aufgabe 9.5

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

Aufgabe 9.5

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

```
und
```

Aufgabe 9.6

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

Aufgabe 9.6

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

```
ABBA
```

Aufgabe 9.7

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

Aufgabe 9.7

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

12

Jedes Zeichen (auch Leerzeichen, Zeichenschaltungen und Tabuloren) wird gezählt.

Aufgabe 9.8

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

Aufgabe 9.8

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

```
['h', 'a', 'l', 'l', 'o']
```

Die `list()`-Methode zerlegt eine Zeichenkette in eine Liste von Einzelzeichen (*characters*).

Aufgabe 9.9

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

Aufgabe 9.9

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

65

An Prüfungen zu diesem Thema steht eine ASCII-Tabelle zur Verfügung, so dass der Wert (die „Ordnungszahl“ des Zeichens) dort abgelesen werden kann.

Aufgabe 9.10

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

Aufgabe 9.10

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

```
B
```

Aufgabe 9.11

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

Aufgabe 9.11

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

2

Die String-Methode `str.count(<zeichenkette>)` zählt, wie oft `<zeichenkette>)` in `str` vorkommt. Man beachte, dass Gross- und Kleinschreibung unterschieden wird.

Aufgabe 9.12

```
1 s = 'Hallo'
2 s = s.lower()
3 print(s)
```

Aufgabe 9.12

```
1 s = 'Hallo'
2 s = s.lower()
3 print(s)
```

```
hallo
```

Aufgabe 9.13

```
1  s = 'ch'
2  s.upper()
3  print(s)
```

Aufgabe 9.13

```
1 s = 'ch'
2 s.upper()
3 print(s)
```

```
ch
```

Achtung: Da Zeichenketten unveränderlich (*immutable*) sind, können die nicht durch die String-Methoden verändert werden. Dafür liefern die Methoden einen Rückgabewert und es liegt in der Verantwortung des Programmierers, diesen Rückgabewert in einer Variablen zu speichern.

Aufgabe 9.14

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

Aufgabe 9.14

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

```
bbcde
```

Die Methode str.replace() ersetzt die Zeichenkette im ersten Parameter durch die Zeichenkette im zweiten Parameter. Man mann die Anzahl der Ersetzungen durch einen dritten Parameter beschränken aber das ist kein Prüfungsstoff.

Aufgabe 9.15

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

Aufgabe 9.15

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

```
x+y+z
```

Aufgabe 9.16

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

Aufgabe 9.16

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

```
['t', 'll', 'r']
```

Aufgabe 9.17

```
1  s = 'abcdxyz'
2  s = s.strip('abyz')
3  print(s)
```

Aufgabe 9.17

```
1 s = 'abcdxyz'
2 s = s.strip('abyz')
3 print(s)
```

```
cdx
```

`str.strip(<zeichenkette>)` entfernt links und rechts von `str` die in `<zeichenkette>` vorkommenden Zeichen. Die Reihenfolge ist dabei unwichtig. Fehlt der Parameter, so werden automatisch alle Formen von „Whitspaces“ (Leerzeichen, Tabulatoren, Zeilenschaltungen) entfernt.

`str.lstrip(<zeichenkette>)` und `str.rstrip(<zeichenkette>)` funktionieren analog, nur dass sie auf jeweils einer Seite (*left*, *right*) wirken.

Aufgabe 9.18

```
1 s = 'abcxyz'
2 s = s.lstrip('abyz')
3 print(s)
```

Aufgabe 9.18

```
1  s = 'abcxyz'
2  s = s.lstrip('abyz')
3  print(s)
```

```
cxyz
```

Aufgabe 9.19

```
1 s = 'abcxyz'
2 s = s.rstrip('abyz')
3 print(s)
```

Aufgabe 9.19

```
1 s = 'abcxyz'
2 s = s.rstrip('abyz')
3 print(s)
```

```
abcx
```

Aufgabe 9.20

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

Aufgabe 9.20

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

```
a3pbem
```

Aufgabe 9.21

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

Aufgabe 9.21

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

```
tek3we
```

Aufgabe 9.22

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

Aufgabe 9.22

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

```
157
```

Aufgabe 9.23

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

Aufgabe 9.23

```
18.0
```

Aufgabe 9.24

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

Aufgabe 9.24

2

Aufgabe 10.1

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

Aufgabe 10.1

3/4

Aufgabe 10.2

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

Aufgabe 10.2

```
(2,7)+(7,2)
```

Aufgabe 10.3

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

Aufgabe 10.3

```
3,1,5
```

Aufgabe 10.4

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

Aufgabe 10.4

7,3

Aufgabe 10.5

```
1 print('{0:+}'.format(123456))
2 print('{0:+}'.format(-123456))
3 print('{0: }'.format(123456))
4 print('{0: }'.format(-123456))
```

Aufgabe 10.5

```
+123456
-123456
 123456
-123456
```

Aufgabe 10.6

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

Aufgabe 10.6

******a

b******

c

Aufgabe 10.7

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

Aufgabe 10.7

```
100,000,000
100_000_000
```

Aufgabe 10.8

```
1 a = 13
2 print('{0:b};{0:#b}'.format(a))
3 print('{0:o};{0:#o}'.format(a))
4 print('{0:d}'.format(a))
5 print('{0:x};{0:#x}'.format(a))
6 print('{0:X};{0:#X}'.format(a))
```

Aufgabe 10.8

```
1101;0b1101
15;0o15
13
d;0xd
D;0XD
```

Aufgabe 10.9

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

Aufgabe 10.9

```
0.667
0.6667
6.667e-01
6.6667e-01
6.6667e+02
```

Aufgabe 10.10

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

Aufgabe 10.10

079-999-99-99

Aufgabe 10.11

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

Aufgabe 10.11

```
abc*uvw
```

Aufgabe 10.12

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

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

Aufgabe 10.12

```
abcxyz
```

Zeilenschaltungen müssen hier explizit mit dem Newline-Character `'\n'` geschrieben werden.

Aufgabe 10.13

Welche Ausgabe macht das folgende Programm

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

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

```
1 5
2 8
3 7
4 9
```

Aufgabe 10.13

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

Aufgabe 11.1

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

Aufgabe 11.1

```
1  3
```

Aufgabe 11.2

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

Aufgabe 11.2

```
1  b
```

Aufgabe 11.3

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

Aufgabe 11.3

```
1  4
```

Aufgabe 11.4

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

Aufgabe 11.4

```
1  -3
```

Aufgabe 11.5

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

Aufgabe 11.5

```
-2
-1
3
9
```

Aufgabe 11.6

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

Aufgabe 11.6

```
1  2
```

Aufgabe 11.7

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

Aufgabe 11.7

```
1 65000
```

Aufgabe 11.8

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

Aufgabe 11.8

```
1  8
```

Aufgabe 11.9

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

Aufgabe 11.9

```
1 99
```

Aufgabe 11.10

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

Aufgabe 11.10

```
1 c
2 m
3 u
```

Aufgabe 11.11

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

Aufgabe 11.11

```
1 [5, 7]
```

Aufgabe 11.12

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

Aufgabe 11.12

```
1  6
```

Aufgabe 11.13

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

Aufgabe 11.13

```
1  3
```

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

Aufgabe 12.1

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

Aufgabe 12.1

```
1  5
```

Aufgabe 12.2

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

Aufgabe 12.2

```
1 False
```

Aufgabe 12.3

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

Aufgabe 12.3

```
1  {1, 2, 3, 4, 5, 7}
```

Aufgabe 12.4

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

Aufgabe 12.4

```
1 {2}
```

Aufgabe 12.5

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

Aufgabe 12.5

```
1  {1, 4, 7}
```

Aufgabe 12.6

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

Aufgabe 12.6

```
1 True
```

Aufgabe 12.7

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

Aufgabe 12.7

```
1  {1, 4, 5, 7}
```

Aufgabe 12.8

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

Aufgabe 12.8

```
1  {1, 7}
```

Aufgabe 12.9

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

Aufgabe 12.9

```
1  13
```

Aufgabe 12.10

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

Aufgabe 12.10

```
1  [1, 2, 3, 7]
```

Aufgabe 12.11

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

Aufgabe 12.11

```
1  42
```

Aufgabe 12.12

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

Aufgabe 12.12

```
1 [1, 3]
```

Aufgabe 12.13

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

Aufgabe 12.13

```
1  ['H', 'N', 'O', 'P', 'T', 'Y']
```

Die Übungen beziehen sich auf die folgende Python-Datei

```
1  a = 3
2
3  def f(x):
4      return 2*x
```

xyz.py

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

Aufgabe 13.1

```
1  import xyz
2
3  print(a)
```

Aufgabe 13.1

```
1 Traceback (most recent call last):
2   File "python-13-pvor-01.py", line 3, in <module>
3     print(a)
4 NameError: name 'a' is not defined
```

Aufgabe 13.2

```
1  import xyz.py
2
3  print(xyz.a)
```

Aufgabe 13.2

```
1 Traceback (most recent call last):
2   File "python-13-pvor-02.py", line 1, in <module>
3     import xyz.py
4 ModuleNotFoundError: No module named 'xyz.py'; 'xyz' is
      not a package
```

Aufgabe 13.3

```
1  from xyz import f
2
3  def f(x):
4      return 3*x
5
6  print(f(10))
```

Aufgabe 13.3

```
1  30
```

Aufgabe 13.4

```
1  import xyz
2
3  print(f(5))
```

Aufgabe 13.4

```
1  Traceback (most recent call last):
2    File "python-13-pvor-04.py", line 3, in <module>
3      print(f(5))
4  NameError: name 'f' is not defined
```

Aufgabe 13.5

```
1  import xyz as u
2
3  print(u.f(5))
```

Aufgabe 13.5

```
1  10
```

Aufgabe 13.6

```
1  from xyz import *
2
3  def f(x):
4      return 4*x
5
6  print(f(10))
```

Aufgabe 13.6

```
1  40
```

Aufgabe 13.7

```
1  from xyz import f as g, a as b
2
3  a = 3
4
5  def f(x):
6      return 4*x
7
8  print(g(a))
```

Aufgabe 13.7

```
1  6
```

Aufgabe 13.8

```
1  import math
2
3  print(math.sqrt(25))
```

Aufgabe 13.8

```
1 5.0
```

Aufgabe 13.9

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

Aufgabe 13.9

```
1 3.14
```

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 15.1

(a) Ein Bauplan für Objekte (ein abstrakter Datentyp)

(b) Ein Objekt, das zur Laufzeit aus einer Klasse erzeugt wird

(c) Eine Variable, die zu einem Objekt gehört

(d) Eine Funktion, die zu einem Objekt gehört

(e) Eine Variable, die zu einer Klasse gehört

(f) Eine Funktion, die zu einer Klasse gehört

(g) Eine spezielle Methode, mit der ein Objekt initialisiert wird.

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 15.2

```
12
```

Aufgabe 15.3

```
1  class Aufgabe():
2
3      def __init__(self, a=3, b=2):
4          self.x = b
5          self.y = a
6
7      def __str__(self):
8          return '({0.y}, {0.x})'.format(self)
9
10 print(Aufgabe(4))
```

Aufgabe 15.3

```
(4, 2)
```

Aufgabe 15.4

```
1  class Vektor():
2
3      def __init__(self, x=0, y=0):
4          self.x = x
5          self.y = y
6
7      def __str__(self):
8          return '({0.x},{0.y})'.format(self)
9
10     def __add__(u, v):
11         return  Vektor(u.x+v.x, u.y+v.y)
12
13     def __sub__(u, v):
14         return Vektor(u.x-v.x, u.y-v.y)
```

```
15
16      def __rmul__(u, k):
17          return Vektor(k*u.x, k*u.y)
18
19      def betrag(self):
20          return (self.x**2 + self.y**2)**0.5
21
22  a = Vektor(2,3)
23  b = Vektor(4,7)
24  c = 2*a-b
25  print(c.betrag())
```

Aufgabe 15.4

```
1.0
```

Aufgabe 15.5

Welche Ausgabe macht das folgende Programm?

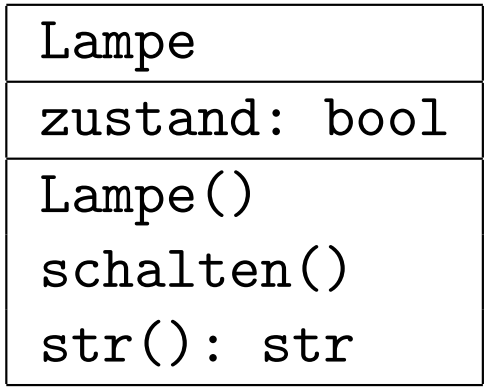
```
1  class Parent():
2      def __init__(self, a):
3          self.a = a
4      def m(self, x):
5          return (x*self.a)
6
7  class Child(Parent):
8      def __init__(self, a, b):
9          super().__init__(a)
10         self.b = b
11     def m(self, y):
12         return (self.a + self.b + super().m(y))
13
14 c = Child(2,3)
15 print(c.m(10))
```

Aufgabe 15.5

25

Aufgabe 15.6

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



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 15.6

```
1  class Lampe:
2
3      def __init__(self):
4          self.zustand = False
5
6      def __str__(self):
7          if self.zustand == True:
8              return 'Lampe an'
9          else:
10             return 'Lampe aus'
11
12     def schalten(self):
13         if self.zustand == False:
14             self.zustand = True
15         else:
16             self.zustand = False
```

```
18 # Test-Client: (nur zur Illustration)
19 L = Lampe()  # erzeugt einen neue (ausgeschaltete) Lampe
20 L.schalten() # Schaltet die Lampe ein
21 L.schalten() # Schaltet die Lampe aus
22 L.schalten() # Schaltet die Lampe ein
23 print(L)     # => 'Lampe ein'
```

Aufgabe 15.7

Welche Ausgaben macht das folgende Python-Modul?

```
1  class Car:
2
3      max_speed = 120
4
5      def __init__(self):
6          self.speed = 0
7
8      def accelerate(self, delta_v):
9          self.speed = min(self.speed+delta_v,
               Car.max_speed)
10
11     def brake(self, delta_v):
12         self.speed = max(self.speed-delta_v, 0)
13
14     def get_speed(self):
15         return self.speed
16
17 c1 = Car()
18 c2 = Car()
```

Aufgabe 15.7

```
1  class Car:
2
3      max_speed = 120
4
5      def __init__(self):
6          self.speed = 0
7
8      def accelerate(self, delta_v):
9          self.speed = min(self.speed+delta_v,
               Car.max_speed)
10
11     def brake(self, delta_v):
12         self.speed = max(self.speed-delta_v, 0)
13
14     def get_speed(self):
15         return self.speed
16
17 c1 = Car()
18 c2 = Car()
19
```

Aufgabe 15.8

Gegeben ist das folgende Python-Modul.

```
1  class Image:
2
3      def __init__(self, width, height):
4          self.w = width
5          self.h = height
6          self.img = [[0 for i in range(width)] for j in
               range(height)]
7
8      def set_pixel(self, x, y, color=1):
9          self.img[y][x] = color
10
11     def write(self, filename):
12         fd = open(filename, mode='w')
13         fd.write('P1\n')
14         fd.write('{0} {1}\n'.format(self.w, self.h))
15         for i in range(0, self.h):
16             for j in range(self.w):
17                 fd.write('{0} '.format(self.img[i][j]))
18         fd.close()
```

Aufgabe 15.8

```
1  class Image:
2
3      def __init__(self, width, height):
4          self.w = width
5          self.h = height
6          self.img = [[0 for i in range(width)] for j in
               range(height)]
7
8      def set_pixel(self, x, y, color=1):
9          self.img[y][x] = color
10
11     def write(self, filename):
12         fd = open(filename, mode='w')
13         fd.write('P1\n')
14         fd.write('{0} {1}\n'.format(self.w, self.h))
15         for i in range(0, self.h):
16             for j in range(self.w):
17                 fd.write('{0} '.format(self.img[i][j]))
18         fd.close()
19
```

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
```

Aufgabe 15.9

```
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
```

Aufgabe 15.10

Gegeben ist das folgendes Python-Modul:

```
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():
```

Aufgabe 15.10

```
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
```

Version vom 23. Oktober 2025