Aufgabe 1

```
1  class Quadrat():
2
3      n = 0
4
5      def __init__(self, a):
6          self.a = a
7          Quadrat.n += 1
8
9      def flaeche(self):
10         return self.a * self.a
11
12     def umfang(self):
13         return 4 * self.a

15 a = Quadrat(5)
16 b = Quadrat(4)
17 print(a.umfang())
18 print(b.flaeche())
19 print(Quadrat.n)
```

Ausgabe Zeile 17: 20

Ausgabe Zeile 18: 16

Ausgabe Zeile 19: 2

Aufgabe 2

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

Ausgabe Zeile 9: 8

Aufgabe 3

```
1  class Example:
2
3      c = 3
4
5      def __init__(self, x):
```

```
        self.a = x
        Example.c += x

e1 = Example(5)
e2 = Example(7)
print(e1.a)
print(e2.a)
print(Example.c)
```

```
5
7
15
```

Aufgabe 4

```
class Point:

    n = 0

    def __init__(self, x, y):
        self.x = x
        self.y = y
        Point.n += 1

    def translate(self, x, y):
        self.x += x
        self.y += y

    def scale(self, factor):
        self.x = factor * self.x
        self.y = factor * self.y

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

A = Point(2, 5)
B = Point(-4, 3)
A.translate(10, 20)
B.scale(-1)
print(A)
print(B)
print(Point.n)
```

```
(12,25)
(4,-3)
2
```

Aufgabe 5

```
class Dog():

    species = 'Canis familiaris'

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def speak(self, sound):
        print('{0} says {1}'.format(self.name, sound))


d1 = Dog('Miles', 5)
d2 = Dog('Jack', 4)
print(d1.age)
print(d2.name)
d1.speak('Woof')
d2.speak('Bow Bow')
```

```
5
Jack
Miles says Woof
Jack says Bow Bow
```

Aufgabe 6

```
class Kreis:

    pi = 3.14

    def __init__(self, radius):
        self.r = radius

    def inhalt(self):
        return self.r**2 * Kreis.pi

    def umfang(self):
        return 2 * self.r * Kreis.pi

k1 = Kreis(10)
k2 = Kreis(1)
print(k1.inhalt())
print(k2.umfang())
```

```
314.0
6.28
```

Aufgabe 7

```
class Textmodifier:

    def __init__(self, string):
        self.s = string

    def reverse(self):
        return ''.join(c for c in self.s[::-1])

    def stretch(self, n):
        return ''.join(n * c for c in self.s)

    def shift(self, pos):
        n = len(self.s)
        return ''.join(self.s[(i-pos)%n] for i in range(n))

    def unvowelize(self):
        return ''.join(c for c in self.s if c not in 'AEIOUaeiou')


x = Textmodifier('Hammer')
print(x.reverse())
print(x.stretch(3))
print(x.shift(2))
print(x.unvowelize())
```

Aufgabe 8

```
class Robot:

    def __init__(self, name, x=0, y=0):
        self.name = name
        self.x = x
        self.y = y
        self.energy = 100

    def go_north(self, steps):
        self.y += steps
        self.energy -= steps

    def go_south(self, steps):
        self.y -= steps
        self.energy -= steps

    def go_east(self, steps):
        self.x += steps
        self.energy -= steps

    def go_west(self, steps):
        self.x -= steps
        self.energy -= steps

```

```
    def __str__(self):
        txt  = '{0}\n'.format(self.name)
        txt += 'Position: ({0},{1})\n'.format(self.x, self.y)
        txt += 'Energy: {0}\n'.format(self.energy)
        return txt

r1 = Robot('R2-D2', 3, 5)
r2 = Robot('Asimov')
r1.go_south(4)
r1.go_west(10)
r1.go_north(12)
r2.go_east(7)
r2.go_north(3)
r2.go_west(5)

print(r1)
print(r2)
```

```
R2-D2
Position: (-7,13)
Energy: 74

Asimov
Position: (2,3)
Energy: 85
```

Aufgabe 9

```
class Rechteck:
    '''Klasse für Rechtecksberechnungen'''

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __str__(self):
        return 'a={0.a}, b={0.b}'.format(self)

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

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

if __name__ == '__main__':

    r1 = Rechteck(3, 2)
    r2 = Rechteck(1, 5)

```

```
    print(r1)
    print(r1.inhalt())
    print(r2.umfang())
```

Aufgabe 10

```
class Quader:

    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def volumen(self):
        return self.a * self.b * self.c

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

Aufgabe 11

```
class Fach:

    def __init__(self, name):
        self.name = name
        self.noten = []

    def __str__(self):
        for i in range(len(self.noten)):
            print(f'Note {i+1}: {self.notem[i]}')

    def neue_note(self, note):
        self.note.append(note)

    def mittelwert(self):
        mw = round(sum(self.noten)/len(noten), 3)
        print(f'Durchschnitt in {self.fach}: {mw}')
```

Aufgabe 12

```
class Linear_function:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __str__(self):
```

```
8           '''Returns a text representation'''
9           return f'{self.a}*x{self.b:+}'
10
11      def slope(self):
12          '''Returns the slope'''
13          return f'{self.a}'
14
15      def intercept(self):
16          '''Returns the intercept'''
17          return f'{self.b}'
18
19      def root(self):
20          '''Returns the root, if one exists'''
21          if self.a != 0:
22              return -self.b/self.a
23          elif self.b == 0:
24              return 'infinite roots'
25          else:
26              return 'no roots'
27
28      def table(self, start, end, step=1):
29          '''Prints a table of values'''
30          x = start
31          while x <= end:
32              print(f'{x}, {self.a*x+b}')
33
34  g = linear_function(2, -1)
35  print(g)              # 2*x-1
36  print(g.intercept()) # -1
37  print(g.root()       # 0.5
38  print(g.table(0, 3)  # 0, -1
39                       # 1, 1
40                       # 2, 3
41                       # 3, 5
```

Aufgabe 13

(a) Zeile 31: `(2,3,1)^T`

Zeile 32: `(5,9,-4)^T`

Zeile 33: `(30,60,-50)^T`

(b)
```
20      def __sub__(self, other):
21          x = self.x - other.x
22          y = self.y - other.y
23          z = self.z - other.z
24          return Vector(x, y, z)
```

(c)
```
26      def __abs__(self):
27          return (self.x**2 + self.y**2 + self.z**2)**0.5
```

(d)
```
    def dot(self, other):
        return self.x*other.x + self.y*other.y + self.z*other.y
```

(e)
```
a = Vector(2,3,1)
b = Vector(3,6,-5)
print(a)
print(a+b)
print(10*b)

c = Vector(2, -1, 2)
d = Vector(5, 4, -3)
print(c - d)
print(abs(c))
print(a.dot(b))
```