Python (OOP)

Übungen

Aufgabe 1

Was gibt das Python-Modul in den Zeilen 17–19 aus?

```
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)
```

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:

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

Ausgabe Zeile 17: 20

```
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:

```
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

```
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:

```
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

Welche Ausgabe macht das folgende Python-Modul?

```
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)
```

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:

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

Welche Ausgabe macht das folgende Python-Modul?

```
1  class Example:
2
3      c = 3
4
5      def __init__(self, x):
6          self.a = x
7          Example.c += x
8
9  e1 = Example(5)
10 e2 = Example(7)
11 print(e1.a)
12 print(e2.a)
13 print(Example.c)
```

Aufgabe 3

```
1  class Example:
2
3      c = 3
4
5      def __init__(self, x):
6          self.a = x
7          Example.c += x
8
9  e1 = Example(5)
10 e2 = Example(7)
11 print(e1.a)
12 print(e2.a)
13 print(Example.c)
```

```
5
7
15
```

Aufgabe 4

Welche Ausgaben macht das folgende Programm?

```
1  class Point:
2
3      n = 0
4
5      def __init__(self, x, y):
6          self.x = x
7          self.y = y
8          Point.n += 1
9
10     def translate(self, x, y):
11         self.x += x
12         self.y += y
13
14     def scale(self, factor):
15         self.x = factor * self.x
16         self.y = factor * self.y
17
18     def __str__(self):
```

Aufgabe 4

```
1  class Point:
2
3      n = 0
4
5      def __init__(self, x, y):
6          self.x = x
7          self.y = y
8          Point.n += 1
9
10     def translate(self, x, y):
11         self.x += x
12         self.y += y
13
14     def scale(self, factor):
15         self.x = factor * self.x
16         self.y = factor * self.y
17
18     def __str__(self):
19         return f'({self.x},{self.y})'
20
```

Aufgabe 5

Welche Ausgabe macht das folgende Python-Modul?

```
1  class Dog():
2
3      species = 'Canis familiaris'
4
5      def __init__(self, name, age):
6          self.name = name
7          self.age = age
8
9      def speak(self, sound):
10         print('{0} says {1}'.format(self.name, sound))
11
12
13 d1 = Dog('Miles', 5)
14 d2 = Dog('Jack', 4)
15 print(d1.age)
16 print(d2.name)
17 d1.speak('Woof')
18 d2.speak('Bow Bow')
```

Aufgabe 5

```
1  class Dog():
2
3      species = 'Canis familiaris'
4
5      def __init__(self, name, age):
6          self.name = name
7          self.age = age
8
9      def speak(self, sound):
10         print('{0} says {1}'.format(self.name, sound))
11
12
13 d1 = Dog('Miles', 5)
14 d2 = Dog('Jack', 4)
15 print(d1.age)
16 print(d2.name)
17 d1.speak('Woof')
18 d2.speak('Bow Bow')
```

Aufgabe 6

Welche Ausgabe macht das folgende Python-Modul?

```
1  class Kreis:
2
3      pi = 3.14
4
5      def __init__(self, radius):
6          self.r = radius
7
8      def inhalt(self):
9          return self.r**2 * Kreis.pi
10
11     def umfang(self):
12         return 2 * self.r * Kreis.pi
13
14 k1 = Kreis(10)
15 k2 = Kreis(1)
16 print(k1.inhalt())
17 print(k2.umfang())
```

Aufgabe 6

```
1  class Kreis:
2
3      pi = 3.14
4
5      def __init__(self, radius):
6          self.r = radius
7
8      def inhalt(self):
9          return self.r**2 * Kreis.pi
10
11     def umfang(self):
12         return 2 * self.r * Kreis.pi
13
14 k1 = Kreis(10)
15 k2 = Kreis(1)
16 print(k1.inhalt())
17 print(k2.umfang())
```

```
314.0
```

Aufgabe 7

Welche Ausgaben macht das folgende Programm?

```
1  class Textmodifier:
2
3      def __init__(self, string):
4          self.s = string
5
6      def reverse(self):
7          return ''.join(c for c in self.s[::-1])
8
9      def stretch(self, n):
10         return ''.join(n * c for c in self.s)
11
12     def shift(self, pos):
13         n = len(self.s)
14         return ''.join(self.s[(i-pos)%n] for i in
               range(n))
15
16     def unvowelize(self):
17         return ''.join(c for c in self.s if c not in
```

Aufgabe 7

```
1  class Textmodifier:
2
3      def __init__(self, string):
4          self.s = string
5
6      def reverse(self):
7          return ''.join(c for c in self.s[::-1])
8
9      def stretch(self, n):
10         return ''.join(n * c for c in self.s)
11
12     def shift(self, pos):
13         n = len(self.s)
14         return ''.join(self.s[(i-pos)%n] for i in
               range(n))
15
16     def unvowelize(self):
17         return ''.join(c for c in self.s if c not in
               'AEIOUaeiou')
18
```

Aufgabe 8

Welche Ausgabe macht das folgende Python-Modul?

```
1  class Robot:
2
3      def __init__(self, name, x=0, y=0):
4          self.name = name
5          self.x = x
6          self.y = y
7          self.energy = 100
8
9      def go_north(self, steps):
10         self.y += steps
11         self.energy -= steps
12
13     def go_south(self, steps):
14         self.y -= steps
15         self.energy -= steps
16
17     def go_east(self, steps):
18         self.x += steps
```

Aufgabe 8

```
1  class Robot:
2
3      def __init__(self, name, x=0, y=0):
4          self.name = name
5          self.x = x
6          self.y = y
7          self.energy = 100
8
9      def go_north(self, steps):
10         self.y += steps
11         self.energy -= steps
12
13     def go_south(self, steps):
14         self.y -= steps
15         self.energy -= steps
16
17     def go_east(self, steps):
18         self.x += steps
19         self.energy -= steps
20
```

Aufgabe 9

Schreibe gemäss der folgenden Vorlage eine Klasse mit dem Namen `Rechteck` für Rechtecksberechnungen, so dass der Testcode ganz unten in der Vorlage die angegebenen Ausgaben macht.

```
1  class Rechteck:
2      '''Klasse für Rechtecksberechnungen'''
3
4      def __init__(self, a, b):
5          ...
6
7      def __str__(self):
8          ...
9
10     def inhalt(self):
11         ...
12
13     def umfang(self):
14         ...
15
16 if __name__ == '__main__':
```

Aufgabe 9

```
1  class Rechteck:
2      '''Klasse für Rechtecksberechnungen'''
3
4      def __init__(self, a, b):
5          self.a = a
6          self.b = b
7
8      def __str__(self):
9          return 'a={0.a}, b={0.b}'.format(self)
10
11     def inhalt(self):
12         return self.a * self.b
13
14     def umfang(self):
15         return 2*(self.a + self.b)
16
17 if __name__ == '__main__':
18
19     r1 = Rechteck(3, 2)
20     r2 = Rechteck(1, 5)
```

Aufgabe 10

Implementiere gemäss dem folgenden Klassendiagramm eine Klasse für die Berechnung von Volumen und Oberfläche von Quader-Objekten.

Quader
a: number b: number c: number
Quader(a: number, b: number, c: number) volumen(): number oberflaeche(): number

Aufgabe 10

```
1  class Quader:
2
3      def __init__(self, a, b, c):
4          self.a = a
5          self.b = b
6          self.c = c
7
8      def volumen(self):
9          return self.a * self.b * self.c
10
11     def oberflaeche(self):
12         return 2*(self.a*self.b + self.b*self.c
13                   + self.c*self.a)
```

Aufgabe 11

Implementiere aufgrund des folgenden Klassendiagramms eine Klasse `Fach` zum Verwalten von Prüfungsnoten in einem Schulfach.

`Fach`
`name: str` `noten: list`

`Fach(name: str)`
`neue_note(note: float)`
`mittelwert(): float`
`ausgabe(): None`

- Der Konstruktor soll die Variable `noten` mit der leeren Liste initialisieren.
- Die Methode `neue_note()` soll die Notenliste um eine Note erweitern.
- Die Methode `mittelwert()` soll den Durchschnitt der darin enthaltenen Noten berechnen in der Shell ausgaben.
- Die Methode `ausgabe()` soll die Einzelnoten wie folgt in der

Aufgabe 11

```
1  class Fach:
2
3      def __init__(self, name):
4          self.name = name
5          self.noten = []
6
7      def __str__(self):
8          for i in range(len(self.noten)):
9              print(f'Note {i+1}: {self.notem[i]}')
10
11     def neue_note(self, note):
12         self.note.append(note)
13
14     def mittelwert(self):
15         mw = round(sum(self.noten)/len(noten), 3)
16         print(f'Durchschnitt in {self.fach}: {mw}')
```

Aufgabe 12

Schreibe eine Klasse mit dem Namen `Linear_function`, welche die folgenden Anforderungen erfüllt.

- Der Konstruktor nimmt die Werte von Steigung (`a`) und Ordinatenabschnitt (`b`) entgegen und speichert sie in den entsprechenden Objekteigenschaften.
- Definiere die Methode `__str__(self)`, welche die Zeichenkette `y=a*x+b` zurückgibt.

  Hinweis: `f'{x:+}'` gibt die Zahl `x` immer mit einem Vorzeichen aus; also `+4`, `+0` oder `-5`.
- Definiere die Methode `slope(self)`, welche die Steigung der zur Funktion gehörenden Gerade zurückgibt.
- Definiere die Methode `intercept(self)`, die den Ordinatenabschnitt der zur Funktion gehörenden Gerade zurückgibt.
- Definiere die Methode `root(self)` welche die Nullstelle der zur Funktion gehörenden Gerade zurückgibt.
- Definiere die Methode `table(self, start, end, step=1)`

Aufgabe 12

```
1  class Linear_function:
2
3      def __init__(self, a, b):
4          self.a = a
5          self.b = b
6
7      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'''
```

Aufgabe 13

Die Klasse `Vector` implementiert einen Datentyp für die Komponentendarstellung von Vektoren im dreidimensionalen Raum.

(a) Welche Ausgaben macht der Python-Code in den den Zeilen 31–33?

(b) Vervollständige die Spezialmethode `__sub__(self, other)`, welche die Differenz der beiden Vektor-Objekte zurückgibt.

(c) Vervollständige die Spezialmethode `__abs__(self)`, welche den Betrag, d. h. die Länge eines Vektor-Objekts zurückgibt.

(d) Vervollständige die Methode `dot(self, other)`, die das Skalarprodukt der beiden Vektoren berechnet und zurückgibt. *Zur Erinnerung:* $\vec{a} \cdot \vec{b} = x_a \cdot x_b + y_a \cdot y_b + z_a \cdot z_b$.

(e) Teste die implementierten Methoden aus (b)–(d) mit geeigneten Beispielen.

```
1  class Vector:
2
3      def __init__(self, x, y, z):
4          self.x = x
5          self.y = y
6          self.z = z
7
8      def __str__(self):
9          return '({0},{1},{2})^T'.format(self.x, self.y,
               self.z)
10
11     def __add__(self, other):
12         x = self.x + other.x
13         y = self.y + other.y
14         z = self.z + other.z
15         return Vector(x, y, z)
```

```
17      def __rmul__(self, a):
18          return Vector(a*self.x, a*self.y, a*self.z)
19
20      def __sub__(self, other):
21          ...
22
23      def __abs__(self):
24          ...
25
26      def dot(self, other):
27          ...
28
29  a = Vector(2,3,1)
30  b = Vector(3,6,-5)
31  print(a)
32  print(a+b)
33  print(10*b)
```

Aufgabe 13

(a) Zeile 31:

Aufgabe 13

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

Aufgabe 13

(a) Zeile 31: `(2,3,1)^T`
Zeile 32:

Aufgabe 13

(a) Zeile 31: `(2,3,1)^T`
Zeile 32: `(5,9,-4)^T`

Aufgabe 13

(a) Zeile 31: `(2,3,1)^T`
Zeile 32: `(5,9,-4)^T`
Zeile 33:

Aufgabe 13

(a) Zeile 31: `(2,3,1)^T`
Zeile 32: `(5,9,-4)^T`
Zeile 33: `(30,60,-50)^T`

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

```
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))
```