Klasse

Eine *Klasse* ist ein Bauplan, um Objekte zu erzeugen. Sie definiert die Daten (*Attribute*) und Funkionalität (*Methoden*) der Objekte. Sowohl auf die Attribute als auch auf die Methoden können mit der Punkt-Notation zugegriffen werden.

Objekt

Ein *Objekt* (Synonym *Instanz*) ist ein Bündel aus Attributen und Methoden, das gemäss einer Klassendefinition aufgebaut ist. Oft entspricht ein Objekt einer Sache in der realen Welt. So könnte `v` das Objekt sein, das gemäss der Klassendefinition `Vektor` erstellt wird. Ein Objekt besteht somit aus einer Menge von Attributen und Methoden, die zu einer Einheit zusammegefasst sind.

Instanziierung

Die *Instanziierung* ist der Prozess, bei dem ein Objekt einer Klasse erzeugt wird. In der Klassendefinition definiert die spezielle Methode `__init__(self,...)`, wie dieses Objekt erzeugt wird.

Methode

Eine *Methode* ist ein Stück Funktionalität eines Objekts. Methoden werden in der Klassendefinition, ähnlich wie Funktionen, mit dem Schlüsselwort `def` definiert.

Eine *Objektmethode* (oder *Instanzmethode*) ist eine Methode die an eine Instanz gebunden ist und die Zugriff auf die Attribute der Instanzbjekts hat. Dieser Zugriff wird in der Methodendefinition über das erste Argument realisiert, das üblicherweise den Namen `self` hat.

Eine *Klassenmethode* ist eine Methode die an die Klasse gebunden ist und nur Zugriff auf die Klassenattribte hat.

`self`

Das erste Argument bei der Definition einer Objektmethode ist immer `self`. Es steht für die Instanz, welche die Methode aufruft. Damit erhält der Python-Interpreter die Information, um welches konkrete Objekt es sich handelt. Beim Aufruf einer Methode mit `obj.method(...)` wird `self` nicht mehr angegeben, da `self` implizit im Objektnamen enthalten ist.

Kapselung

Kapselsung bezeichnet den kontrollierten Zugriff auf die Objekte einer Klasse über eine Schnittstelle.

Attribute

Ein *Attribut* ist eine Variable, die zu einer Klasse (*Klassenattribut*) oder zu einem Objekt (*Objektattribut*) gehört.

Ein *Klassenattribut* (oder *statisches Attribut*) ist eine Variable, deren Wert von allen Objekten (und dem Klassenobjekt) geteilt wird.

Eine *Instanzattribut* ist eine Variable, welche Werte enthält, die zu einer einzelnen Instanz gehören und nicht mit anderen Instanzen geteilt werden.

Methoden

Eine *Methode* ist eine Funktion, die zu einer Klasse (*Klassenmethode*) oder zu einem Objekt (*Objektmethode*) gehört.

Eine *Klassenmethode* (oder *statische Methode*) ist eine Funktion, die ohne eine Objekt (bzw. vom Klassenobjekt) ausgeführt und nur Zugriff auf die Klassenvariablen hat.

Eine *Instanzmethode* ist eine Methode, die in Verbindung mit einem konkreten Objekt ausgeführt wird und daher Zugriff auf die Eigenschaften dieses Objekts hat.

Überladen von Methoden

Das *Überladen* bezeichnet die Möglichkeit, Objekte verschiedener Klassen mit (unterschiedlichen) Methoden auszustatten, die aber den gleichem Namen haben.

Ein bekanntes Beispiel ist der Operator „`+`“ (eigentlich die Methode `__add__`) mit dem in Python sowohl ganze Zahlen, Gleitkommazahlen und Zeichenketten „addiert“ werden können.

Python besitzt eine Menge sogenannter *Spezialmethoden*, die es erlauben, die Namen bestehender Operatoren und Funktionen für die eigenen Klassen zu verwenden. Hier eine kleine Auswahl:

Name	Aufruf mit
`__add__(self, other)`	`obj1 + obj2`
`__lt__(self, other)`	`obj1 < obj2`
`__ge__(self, other)`	`obj1 >= obj2`
`__eq__(self, other)`	`obj1 == obj2`
`__str__(self)`	`str(obj)`
`__len__(self)`	`len(obj)`
`__getitem__(self, key)`	`obj[key]`
`__iter__(self)`	`for x in obj:`

Vererbung

Eine Klasse `Child` kann mit `class Child(Parent):` in der Klassendefinition die Eigenschaften und Methoden der Klasse `Parent` übernehmen.

Dabei können die von der Mutterklasse geerbten Attribute und Methoden durch zusätzliche Attribute und Methoden erweitert werden. Ferner ist es möglich, Methoden der Mutterklasse zu *überschreiben*, d. h. neu zu definieren.

Aufgabe 1

Welche Ausgaben macht das folgende Modul in den Zeilen 43–49.

```
1  class Figur():
2
3      PI = 3.14
4      counter = 0
5
6      def was_bin_ich(self):
7          return 'Eine Figur'
8      def flaeche(self):
9          pass
10
11 class Rechteck(Figur):
12
13     def was_bin_ich(self):
14         return 'Ein Rechteck'
15
16     def __init__(self, a=1, b=1):
17         self.a = a
18         self.b = b
19         Figur.counter += 1
20
21     def __str__(self):
22         return 'a={0.a}, b={0.b}'.format(self)
23
24     def flaeche(self):
25         return self.a * self.b
26
27 class Kreis(Figur):
28
29     def __init__(self, r=1):
30         self.r = r
31         Figur.counter += 1
32
33     def __str__(self):
34         return 'r={0.r}'.format(self)
35
36     def flaeche(self):
37         return Figur.PI * self.r**2
38
39 r1 = Rechteck(3, 4)
40 r2 = Rechteck()
41 k = Kreis(10)
42 print(Figur.PI)
43 print(r1, r1.flaeche())
44 print(r2, r2.flaeche())
45 print(k, k.flaeche())
46 print(r1.was_bin_ich())
47 print(k.was_bin_ich())
48 print(Figur.counter)
```

Aufgabe 2

Welche Ausgaben macht das folgende Modul in den Zeilen 27–34?

```
1  class A(object):
2      def __init__(self, v):
3          self.v = v
4      def x(self):
5          return self.v + 1
6      def y(self):
7          return self.v + 2
8      def z(self):
9          return self.v + 3
10
11 class B(A):
12     def __init__(self, v):
13         self.v = v
14     def x(self):
15         return self.v + 3
16
17 class C(B):
18     def __init__(self, v):
19         self.v = v
20     def z(self):
21         return self.v + 4
22
23 a = A(10)
24 b = B(20)
25 c = C(30)
26
27 print(a.x())
28
29 print(b.x())
30 print(b.y())
31
32 print(c.x())
33 print(c.y())
34 print(c.z())
```

Aufgabe 3

Erkläre die folgenden Begriffe im Zusammenhang mit der objektorientierten Programmierung

(a) Klasse

(b) Instanz (oder Objekt)

(c) Objekteigenschaft

(d) Objektmethode

(e) Klasseneigenschaft

(f) Klassenmethode

(g) Konstruktor

(h) Vererbung

Aufgabe 4

Vervollständige die Methoden der folgenden Klasse zum Rechnen mit Vektoren in $\mathbb{R}^3$.

```
class Vektor:

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

    def __str__(self):
        '''Gib eine Stringdarstellung des Vektors zurück.'''
        ...









    def __add__(self, other):
        '''Gibt die Summe der Vektoren zurück.'''
        ...









    def __abs__(self):
        '''Gibt die Länge des Vektors zurück.'''
        ...
```

Prüfungsstoff

1. Du kannst folgende Begriffe prägnant erklären:

   - Klasse
   - Instanz (Objekt)
   - Instanzvariable, Klassenvariable
   - Instanzmethode, Klassenmethode
   - Konstruktor
   - Vererbung
   - Kapselung
   - Überschreiben von Methoden

2. Du kannst einfache OOP-Programme in der Programmiersprache Python interpretieren. Um dies zu tun, kennst die Bedeutung der folgenden Schlüsselwörter:

   - `class <Name>`
   - `class <Name>(<Elternklasse>)`
   - `__init__(self,...)`

   Die Namen der Spezialmethoden wie `__str__()` oder `__add__()` werden, falls nötig, zur Verfügung gestellt.

3. Du weisst, dass bei der Definition von Instanzmethoden als erstes Argument jeweils ein formaler Objektname (üblich ist `self`) angegeben werden muss.

4. Du kannst eine einfache Klasse aufgrund einer vorgegebenen API (Programmierschnittstelle) implementieren.

5. Du weisst wie Variablennamen innerhalb der Vererbungshierarchie aufgelöst werden.

6. Du weisst, dass jede Kindklasse automatisch die Variablen und Methoden ihrer Elternklasse erbt und dass diese Variablen und Methoden auch überschrieben oder durch neue Variablen und Methoden ergänzt werden können.

7. Du kannst die wesentlichen (im Unterricht behandelten) Vorzüge von OOP aufzählen.