

Objektorientierte Programmierung (OOP)

Grundidee



Bereits mit der Definition von Funktionen haben Sie ein Werkzeug kennengelernt, wie Sie Code übersichtlich und wiederverwendbar gestalten können. Während es für kleinere Softwareprojekte meist vollkommen ausreichend ist, den Code in ein paar Funktionen zu gliedern, ist es bei größeren Projekten sinnvoll einen Schritt weiterzugehen und den Code **objektorientiert** zu konzipieren. Die Objektorientierte Programmierung (OOP) ist ein Konzept in der Softwareentwicklung, bei dem Objekte aus der realen Welt im Code abgebildet werden. Die Grundidee ist hierbei, die Daten (*Variablen*) und den zugehörigen Code (*Funktionen*) eines Objekts im Code zu einer **Einheit** zu **bündeln**. Dieser Ansatz löst viele Probleme, da die Daten und ihre Operationen dadurch nicht mehr lose über ein Programm verstreut sind. (adaptiert aus [1–3])

Die Klasse als Bauplan für ein Objekt

Daten (*Variablen*) und ihre Operationen (*Funktionen*) werden zu einer Einheit gebündelt, die man **Klasse** nennt. Eine Klasse ist ein **Datentyp**, der als Vorlage bzw. Bauplan für ein oder mehrere **Objekte** dient. Die Klasse enthält *Variablen* zur Speicherung der Informationen und *Funktionen*, um das Verhalten des Objekts umzusetzen. Im Umfeld der OOP spricht man bei Variablen nun von **Attributen** und bei Funktionen von **Methoden**.



In der OOP ändern sich die **Bezeichnungen** für Daten und ihre Operationen:

Variable → Attribut

Funktion → Methode

Durch eine Klasse können **beliebig viele Objekte** dieses Typs erstellt werden. Jedes Objekt einer Klasse kann dabei eigene, von anderen Objekten **unabhängige** Werte der Attribute (Eigenschaften) haben.



Eine Klasse ist ein Bauplan für ein Objekt. Jedes Objekt hat seine eigenen Eigenschaften.

Beschreibung einer Klasse mit dem Klassendiagramm


Klassen werden mithilfe eines **Klassendiagramms** dargestellt, in welches alle Attribute und Methoden der Klasse eingetragen werden. Dieses besteht aus einem **Rahmen** ①, einem **Klassennamen** ② und jeweils einem Bereich für **Attribute** ③ und **Methoden** ④. Das Klassendiagramm gibt somit Auskunft über die Eigenschaften und das Verhalten der Klasse bzw. der daraus erzeugten Objekte.



Im Klassendiagramm werden **keine Details** über die Implementierung definiert, sondern nur die **wichtigsten Elemente** der Klasse notiert. Die Angabe des Datentyps der Attribute (**blau** dargestellt) und der Übergabeparameter der Methoden (im Beispiel keine vorhanden) ist oftmals sinnvoll und hilfreich.

Definition einer Klasse in Python

Auf Grundlage eines Klassendiagramms kann die Klasse in Python definiert werden. Eine **Klasse** wird in Python mit einem frei wählbaren **Namen** (z.B. Auto) und dem Schlüsselwort `class` definiert:



Belegung der Attribute mit **Standardwerten**

```

class Auto:

    def __init__(self):
        1 self.bezeichnung = ""
          self.baujahr = 2000
          self.verkauft = False

    2 def daten_anzeigen(self):
      # Daten auf der Konsole ausgeben...

    def verkaufen(self):
      # self.verkauft = True
    
```

Die Klasse Auto definiert **Attribute** (**1**) für die Bezeichnung, das Baujahr und den Verkauft-Status. Die Attribute werden innerhalb einer Initialisierungsfunktion definiert, die zunächst nicht relevant ist. Außerdem sind die **Methoden** `daten_anzeigen()` und `verkaufen()` (**2**) definiert, welche die Daten auf der Konsole ausgeben bzw. den Status des Autos auf „verkauft“ setzen. Sowohl die Attribute als auch die Methoden sind innerhalb der Klasse angesiedelt. Den Attributen wird das Schlüsselwort `self` vorangestellt, dessen Bedeutung zunächst keine Rolle spielt.

Erstellen von Objekten

Um ein Objekt, also eine **Ausprägung** (sog. **Instanz**) dieser Klasse zu erstellen, kann man ein Objekt der Klasse mit einem frei wählbaren Objekt-Namen (z.B. `erstesAuto`) wie folgt erzeugen:

```

erstesAuto = Auto() # Objekt der Klasse "Auto" erstellen.

erstesAuto.bezeichnung = "VW Golf" # Werte an Attribute des Objekts zuweisen.
erstesAuto.baujahr = 2018
erstesAuto.verkauft = False

erstesAuto.daten_anzeigen() # Methode des Objekts aufrufen.
    
```

Im Beispiel werden nach dem Erstellen des Objekts den **Attributen** neue Werte zugewiesen. Außerdem wird die **Methode** `daten_anzeigen()` des Objekts aufgerufen. Auf die Attribute und Methoden eines erstellten Objekts wird mit dem **Punkt-Operator** (`.`) nach dem Objekt-Namen zugegriffen.



Auf die Attribute und Methoden eines erstellten Objekts wird mit dem **Punkt-Operator** (`.`) nach dem Objekt-Namen zugegriffen, z.B. `erstesAuto.bezeichnung = "VW Golf"`

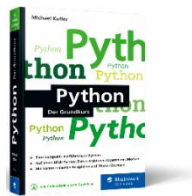


Fragen zur **Selbstkontrolle**:

- Überlegen Sie sich, welcher Zusammenhang zwischen einer Klasse und einem Objekt besteht.
- Überlegen Sie sich, wie viele Objekte von einer Klasse erstellt werden können.
- Beide Methoden innerhalb der Klasse Auto nehmen keine Übergabeparameter entgegen. Überlegen Sie sich, wie die Methoden ohne Übergabeparameter auf die nötigen Informationen bzw. Attribute zugreifen können.



FÜR SCHNELLE



[4]



Für Schnelldenkende: Informieren zur Konstruktor-Methode `__init__()`

- Informieren Sie sich mithilfe des Python-Fachbuchs zur Konstruktor-Methode im Kapitel Objektorientierte Programmierung (Kapitel 11.2 „Hello, Class!“).

Quellenverzeichnis

- [1] pngimg.com. "blue Volkswagen Golf PNG car image image with transparent background." Zugriff am: 11. Januar 2024.
- [2] M. Kofler, *Python: Der Grundkurs*, 2. Aufl. (Rheinwerk Computing). Bonn: Rheinwerk, 2022. [Online]. Verfügbar unter: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6807936>
- [3] P. Loos, *Objektorientiertes Programmieren in Visual C#: Eine methodische Einführung für Einsteiger und Fortgeschrittene* ; [mit Visual C# 2005 Express Edition auf CD (Fachbibliothek). Unterschleißheim: Microsoft Press, 2006.
- [4] Rheinwerk Verlag GmbH. "Python - Der Grundkurs." Zugriff am: 11. Januar 2024. [Online.] Verfügbar: <https://www.rheinwerk-verlag.de/python-der-grundkurs/>