# Lektion 16 – Vererbung

## Nachschlagen bei…

W3Schools (Kapitel Inheritance) oder <https://entwickler.de/online/development/einfuehrung-programmierung-objektorientierte-programmentwicklung-197372.html> oder <https://www.python-kurs.eu/python3_vererbung.php>.

## Anweisungen

Studiere die Erläuterungen in diesem Dokument bzw in den Nachschlagewerken. Beantworte die Fragen und kopier den Code der Aufgaben hierher.

## Erläuterungen

**Was ist Vererbung bei OOP?** Um die Wiederverwertbarkeit von Code zu steigern besteht in den meisten objektorientierten Sprachen die Möglichkeit, von einer Klasse eine Subklasse (Kindklasse) zu erzeugen. Die ursprüngliche Klasse nennt man dann Superklasse (Elternklasse).



Wenn nicht anders angegeben, erben Subklassen alle Eigenschaften und Methoden der Superklasse! Man kann aber Methoden überschreiben, in dem man Methoden mit gleichem Namen neu definiert. Man kann auch weitere Methoden und Eigenschaften definieren, die die Superklasse nicht hat.

## Beispiel in Python

# Superklasse: es wird allgemein die Klasse Fahrzeug definiert

class fahrzeug(object):

 def \_\_init\_\_(self,farbe, baujahr):

 self.farbe=farbe

 self.baujahr=baujahr

 self.geschwindigkeit=0

 def beschreibung(self):

 print("Ich wurde im Jahr",self.baujahr," gebaut und habe die Farbe",self.farbe)

 def beschleunigen(self,x):

 self.geschwindigkeit+=x

 def geschwindigkeitsanzeige(self):

 print("Ich fahre mit",self.geschwindigkeit,"km/h")

# Subklasse: der Name der Superklasse wird in Klammer angegegeben

class pkw(fahrzeug):

 def \_\_init\_\_(self,hersteller, typ, farbe, baujahr):

 # der Konstruktor der Superklasse wir mit super()... aufgerufen

 super().\_\_init\_\_(farbe, baujahr)

 self.hersteller=hersteller

 self.typ=typ

 # eine Methode mit demselben Namen wie bei der Superklasse überschreibt die Methode der Superklasse

 def beschreibung(self):

 print("Ich bin ein",self.hersteller,self.typ+".")

 super().beschreibung()

 # die Methoden beschleunigen() und geschwindigkeitsanzeige() werden nicht überschrieben sondern einfach geerbt.

class motorrad(fahrzeug):

 def \_\_init\_\_(self,hersteller, typ, farbe, baujahr):

 super().\_\_init\_\_(farbe, baujahr)

 self.hersteller=hersteller

 self.typ=typ

 def beschreibung(self):

 print("Ich bin eine",self.hersteller,self.typ+".")

 super().beschreibung()

 # neue Methoden können definiert werden

 def bremsen(self,x):

 self.geschwindigkeit-=2\*x

# zuerst wird ein Objekt instanziert

auto1 = pkw("Skoda","Oktavia","blau",2017)

auto1.beschreibung()

auto1.beschleunigen(30)

auto1.geschwindigkeitsanzeige()

# es können beliebig viele weitere Objekte instanziert werden

bike1 = motorrad("Honda","Goldwing","rot",2013)

bike1.beschreibung()

bike1.beschleunigen(100)

bike1.geschwindigkeitsanzeige()

bike1.bremsen(30)

bike1.geschwindigkeitsanzeige()

## Fragen

Was gibt der Code des Beispielprogramms aus? (kopier die Ausgabe hierher)

Wie wird in der Kindklasse eine Methode der Elternklasse aufgerufen?

Was kann man machen, wenn eine Methode der Subklasse etwas anderes machen soll, als die gleichnamige Methode der Superklasse?

## Aufgaben

1. Kopier das Beispielprogramm in die Datei vererbung.py. Erstelle eine weitere Subklasse lkw. Erzeuge ein Objekt davon und rufe die Methode beschreibung() auf.
2. Erzeuge eine Datei staubsaugerroboter.py. Kopiere die Roboterklasse aus roboter.py hinein. Definiere eine Subklasse der Klasse roboter namens „staubsaugerroboter“. Sie soll zusätzlich die Eigenschaft „bereich“ besitzen (Küche, Wohnzimmer,…). Überschreibe die Methode sayhello() so, dass er dazusagt, für welchen Bereich er zuständig ist. Erzeuge eine Instanz von staubsaugerroboter und rufe die Methoden sayhello() und lackieren() auf.