Kapitel 17.2: Vererbung – Glossar

Vererbung (Inheritance)

Wenn Objekte viele Gemeinsamkeiten haben, kann man sie meistens unter einem Überbegriff zusammenfassen. In der Objekt-orientierten Programmierung bildet man das mit Basisklassen und abgeleiteten Klassen ab.

Basisklasse

Die Basisklasse stellt die Grundfunktionalität bereit, die alle davon abgeleitete Klassen gemeinsam haben. Dazu gehören sowohl Attribute als auch Methoden. Die Basisklasse kann nicht auf neue Attribute oder Methoden in einer abgeleiteten Klasse zugreifen. Die Basisklasse wird manchmal auch *Elternklasse*, *Oberklasse* oder *Superklasse* genannt.

Abgeleitet Klasse

Eine abgeleitete Klasse erbt von ihrer Basisklasse alle Attribute und Methoden. Die Attribute können sowohl gelesen als auch geschrieben werden. Die Methoden können aufgerufen werden. Außerdem kann eine abgeleitete Klasse die Methoden der Basisklasse überschreiben. Die abgeleitete Klasse wird auch *Kindklasse*, *Unterklasse* oder *Subklasse* genannt.

Die Klassendefinition wird um die Basisklasse erweitert: `class AbgeleiteteKlasse(Basisklasse):`

super()

Wird in der abgeleiteten Klasse eine Methode der Basisklasse überschrieben, dann handelt es sich häufig eher um eine Erweiterung als eine komplett andere Funktionalität. Gibt es eine Methode sowohl in der Basisklasse als auch in der abgeleiteten Klasse, wird normalerweise immer die Methode der abgeleiteten Klasse genommen. Um im Fall der Erweiterung aber auf der Funktion der Basisklasse aufbauen zu können, kann man mit `super().` explizit auf die Basisklasse zugreifen.

Beispiel:

```
def a_method(self):
    super().a_method()
    # eigene Anweisungen
```

pass Anweisung

`pass` ist eine Anweisung, die nichts macht. Definiert man eine Funktion oder Methode in python, muss diese aber mindestens eine Anweisung (Statement) enthalten. Die pass Anweisung benutzt man also, wenn eine Funktion oder Methode nichts macht. Das ist sinnvoll ...

- wenn man dabei ist, eine Klasse zu entwickeln und schon einmal alle Funktionen definieren möchte. Man kann diese Funktionen dann auch schon aufrufen – auch wenn sie nichts machen;

- in Basisklassen, um eine Methode, die jede abgeleitete Klasse überschreiben sollte zu kennzeichnen;
- wenn man explizit angeben möchte, dass hier nichts passieren soll. Das ist oft nützlich, um z. B. in if-Anweisungen den Überblick zu behalten. Die `pass`-Anweisung kann man also auch in if-Anweisungen oder Schleifen benutzen.