# Methoden implementieren

Wir haben bereits die ersten Grundbegriffe kennen gelernt und mit Klassen und Objekten gearbeitet. Nun erwecken wir unsere Programme mit Methoden zum Leben.

Eine Methode ist eine Funktion, die auf ein Objekt angewendet werden muss. Somit können Sie Ihr Wissen über Funktionen aus dem Fach Informatik direkt nutzen. Wir unterscheiden zwischen:

* Methoden ohne/mit Rückgabewert
* Methoden ohne/mit Parameter

**Übersicht über Methoden in einem Beispiel zur Verwaltung eines Kontos (*L1\_4\_Info.py*)**

|  |  |
| --- | --- |
|  | Wir wollen die folgenden Methoden implementieren:   * anzeigen(): Zeigt die Kontoinformationen an * einzahlen(): Zahlt einen Betrag auf das Konto ein * abheben(): Hebt einen Betrag vom Konto ab * berechne\_zinsen(): Berechnet die Zinsen für das Konto |
| Fachklasse Konto |  |

## Methode ohne Rückgabewert

Zunächst erstellen wir eine Methode, mit der wir bequem die Kontoinformationen anzeigen lassen können.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Konto:  ...  def anzeigen(self):  print("Inhaber:", self.inhaber)  print("Kontonummer:",self.kontonummer)  print("Kontostand:",self.kontostand,"€")  # Hauptprogramm  ...  k.anzeigen() |

Zeile 4: Mit dem Schlüsselwort def definieren wir eine Methode mit dem Namen anzeigen(). Mit dem Parameter self können wir auf das aktuell aufgerufene Objekt zugreifen.

Zeile 11: Für das aktuelle Objekt (hier: k) wird die Methode anzeigen() aufgerufen. Dieses Objekt wird an die Methode übergeben, so dass auf seine Attribute zugegriffen werden kann.

## Methode mit Parameter und ohne Rückgabewert

Nun erstellen wir eine Methode, mit der wir einen Geldbetrag auf das aktuelle Konto einzahlen können. Häufig nutzen wir diese Art von Methode, wenn wir einen Attributwert ändern oder Text ausgeben. Hinweis: Der Parameter *self* erscheint nur bei der Definition einer Methode. Beim Aufruf wird er nicht angegeben.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Konto:  ...  def einzahlen(self, p\_betrag):  self.kontostand = self.kontostand + p\_betrag  # Hauptprogramm  ...  k.einzahlen(150) |

Zeile 3: Mit dem Schlüsselwort def definieren wir eine Methode mit dem Namen einzahlen(). Mit dem Parameter self können wir auf das aktuell aufgerufene Objekt zugreifen. Der Parameter p\_betrag kann im Methodenrumpf verarbeitet werden.

Zeile 8: Für das aktuelle Objekt (hier: k) wird die Methode einzahlen() aufgerufen. Dieses Objekt wird an die Methode übergeben, so dass auf seine Attribute zugegriffen werden kann. Der Wert 150 wird als Parameter ebenfalls an die Methode übergeben.

## Methode mit Rückgabewert

Nun erstellen wir eine Methode, mit der wir die Zinsen für das aktuelle Konto berechnen. Den Zinssatz muss der Nutzer hier selbst angeben. Häufig nutzen wir diese Art von Methode, um etwas zu berechnen. Das Ergebnis dieser Rechnung wird in einer Variablen zwischengespeichert und am Ende der Methode mit *return* zurückgegeben. Hinweis: Der Parameter *self* erscheint nur bei der Definition einer Methode. Beim Aufruf wird er nicht angegeben.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | class Konto:  ...  def berechne\_zinsen(self, p\_zinssatz):  zinsen = self.kontostand \* p\_zinssatz / 100  return zinsen  # Hauptprogramm  ...  zinsbetrag = k.berechne\_zinsen(5)  print("Die Zinsen betragen",zinsbetrag,"€.") |

Zeile 3: Mit dem Schlüsselwort def definieren wir eine Methode mit dem Namen berechne\_zinsen(). Mit dem Parameter self können wir auf das aktuell aufgerufene Objekt zugreifen. Der Parameter p\_zinssatz kann im Methodenrumpf verarbeitet werden.

Zeile 4: Eine Variable zinsen wird erstellt, in der wir das Ergebnis der Berechnung zwischenspeichern.

Zeile 5: Der Wert der Variablen zinsen wird mit *return* als Ergebnis der Methode zurückgegeben.

Zeile 9: Für das aktuelle Objekt (hier: k) wird die Methode berechne\_zinsen() aufgerufen. Dieses Objekt wird an die Methode übergeben, so dass auf seine Attribute zugegriffen werden kann. Der Wert 5 wird als Parameter ebenfalls an die Methode übergeben.