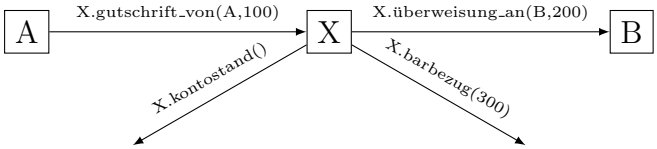
Objektorientierte Programmierung (OOP)

Prinzip: Die Programmierung bildet die zu lösende Aufgabe realitätsnah ab. *Objekte* sind *Akteure* mit *Eigenschaften* (Variablen) und *Methoden* (Funktionen). Durch *Kapselung* werden die Details der Programmierung „*verborgen*“. Auf die Objekte kann dann nur über dafür vorgesehene Funktionen (*Schnittstellen*) zugegriffen werden. Solche Programme lassen sich leichter verbessern, wenn die Schnittstellen unverändert bleiben.

Beispiel: Ein *Konto* hat die Eigenschaften *Inhaberin*, *Kontonummer*, *Kontostand*, *Zinsfuss* und die Methoden *Eröffnung*, *Einzahlung*, *Auszahlung*, *Überweisung*, *Kontoauszug*.



Vorteile:

- Programme sind besser lesbar/verständlich → weniger Fehler
- Komplexe Aufgaben können aufgeteilt werden → Teamwork
- Kapselung → Programme können leichter angepasst und verbessert werden.

OOP in Python (Zugriff auf Attribute hier über Getter- und Setter-Methoden → Kapselung)

```
class Myclass:   # Klassendefinition
    x = 5 # Klassenvariable
    def set_x(new_x): # Klassenmethode
        Myclass.x = new_x # ändert Wert der Klassenvariable
    def get_x(): # Klassenmethode
        return Myclass.x # gibt Wert der Klassenvariable zurück
    def __init__(self, a): # Konstruktor
        self.a = a # Weist Objektvariable den Wert von a zu
    def set_a(self, new_a): # Objektmethode (Setter-Methode)
        self.a = new_a # ändert Wert der Objektvariable
    def get_a(self): # Objektmethode (Getter-Methode)
        return self.a # gibt Wert der Objektvariable zurück
    def exchange(self, other): # Objektmethode
        self.a, other.a = other.a, self.a # tauscht self.a und other.a
# Client-Code:
objekt1 = Myclass(7)   # Konstruktor wird aufgerufen und objekt1 erzeugt
objekt2 = Myclass(4)   # Konstruktor wird aufgerufen und objekt2 erzeugt
objekt1.set_a(3)   # Attribut a von objekt1 (von 7) auf 3 ändern
print(objekt1.get_a())   # => 3 (Attribut a von objekt1 holen und ausgeben)
print(objekt2.get_a())   # => 4 (Attribut a von objekt2 holen- und ausgeben)
Myclass.set_x(8)   # Attribut x der Klasse (von 5) auf 8 ändern
print(Myclass.get_x())   # => 8 (Attribut x der Klasse holen und ausgeben)
```

Synonyme: Variable = Attribut = Eigenschaft; Methode = Funktion; Objekt = Instanz