Objektorientierte Programmierung

Python 3 · kompakte Grundlagen mit direkt nutzbaren Beispielen

1 | Klasse & Objekt

- Klasse ist Bauplan, Objekt eine Instanz.
- Attribute speichern Zustand.

```
class Drucker:
    pass

drucker_1 = Drucker()
drucker_1.name = "Neptune"
```

2 | __init__

- Initialisiert neue Objekte.
- self verweist auf das aktuelle Objekt.

```
class Drucker:
    def __init__(self, name):
        self.name = name
        self.online = False
```

3 | Methoden

```
class Drucker:
    def __init__(self, name):
        self.name = name

    def anzeigen(self):
        print(f"Drucker: {self.name}")
```

4 | Objekt verwenden

```
n4 = Drucker("Neptune 4")
n4.anzeigen()
print(n4.name)
```

5 | Instanz vs. Klasse

- Instanzattribute gehören einem Objekt.
- Klassenattribute werden geteilt.

```
class Sensor:
    einheit = "°C"
    def __init__(self, wert):
        self.wert = wert
```

6 | Kapselung als Konvention

- _name bedeutet: intern verwenden.
- Property kann Zugriff kontrollieren.
- Python setzt Privatsphäre nicht strikt durch.

```
class Motor:
    def __init__(self):
        self._drehzahl = 0
```

Objektorientierte Programmierung

Python 3 · kompakte Grundlagen mit direkt nutzbaren Beispielen

7 | Property

```
class Motor:
    def __init__(self):
        self._rpm = 0

    @property
    def rpm(self):
        return self._rpm
```

8 | Vererbung

```
class Geraet:
    def status(self):
        return "OK"

class Drucker(Geraet):
    pass
```

9 | Komposition

- Objekt enthält andere Objekte.
- Oft flexibler als Vererbung.

```
class Drucker:
    def __init__(self, sensor):
        self.sensor = sensor

drucker = Drucker(Sensor(25))
```

10 | __str__

```
class Material:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return self.name
```

11 | Wann OOP?

- Mehrere gleichartige Objekte mit Zustand.
- Daten und Verhalten gehören zusammen.
- Nicht jede kleine Funktion braucht eine Klasse.
- Klassen klein und eindeutig halten.
- Abhängigkeiten über Konstruktor übergeben.

12 | Mini-Übung: Filament

Aufgabe

Erstelle eine Klasse Filament mit Name und Düsentemperatur sowie einer anzeigen()-Methode.

Musterlösung

```
class Filament:
    def __init__(self, name, duese):
        self.name = name
        self.duese = duese
    def anzeigen(self):
        print(f"{self.name}: {self.duese} °C")
Filament("PLA", 205).anzeigen()
```