OOP in Python

Recap

- Erläutere die folgenden Begriffe:
  - Klasse
  - Objekt
  - Attribut
  - Methode
- Extrahiere aus der folgenden Beschreibung Klassen, Objekte, Attribute & Methoden:

  „Wasser, Wein & Bier sind Getränke. Sie lassen sich durch einen Alkoholgehalt näher beschreiben. Niklas ist eine Person und kann ein Getränk trinken.“

Klassen definieren

```
# Definieren der Klasse:
class Person:
    pass

# Instanziieren von Objekten
klaus = Person()
laura = Person()

print(klaus) # <__main__.Person object at 0x000001804DC86B60>
print(laura) # <__main__.Person object at 0x000001804DC86740>
```

- Für Klassennamen gelten die gleichen Regeln wie für Funktionsnamen
  - Konvention: CapWords (bzw. upper camelCase)
- Problem: Klaus & Laura können nichts!

Klassen mit Attributen

```python
# Definieren der Klasse:
class Person:
    name = "Klaus"
    alter = 39
    beruf = "Anwalt"

# Instanziieren von Objekten
klaus = Person()
laura = Person()

print(klaus.name)   # "Klaus"
print(klaus.alter)  # 39
print(klaus.beruf)  # "Anwalt"

print(laura.name)   # "Klaus"
print(laura.alter)  # 39
print(laura.beruf)  # "Anwalt"
```

- Gut:
  Klaus hat jetzt einen Namen, ein Alter & einen Beruf!
- Schlecht:
  Laura hat genau die gleichen Attributswerte wie Klaus!
- Hässlich:
  Jede Person heißt Klaus, ist 39 und arbeitet als Anwalt
- Fazit:
  Attribute nicht fix setzen sondern flexibel

Klassen mit Attributen

```python
# Definieren der Klasse:
class Person:

    def __init__(self, name, alter, beruf):
        self.name = name
        self.alter = alter
        self.beruf = beruf

# Instanziieren von Objekten
klaus = Person(name="Klaus", alter=39, beruf="Anwalt")
laura = Person("Laura", 37, "Programmierer")

print(klaus.name)   # "Klaus"
print(klaus.alter)  # 39
print(klaus.beruf)  # "Anwalt"

print(laura.name)   # "Laura"
print(laura.alter)  # 37
print(laura.beruf)  # "Programmierer"
```

- __init__() Methode:
  Legt fest, was passieren soll, wenn Objekt instanziiert wird (->Konstruktor)
- Erster Parameter: self
  - Referenz zum Objekt selber
  - Für Instanz klaus gilt:
    self = klaus
  - Für Instanz laura gitl:
    self = laura
- Kann grundsätzlich auch anders genannt werden

Attribute adressieren & verändern

```python
print(klaus.name)   # "Klaus"

print(klaus.alter)  # 39
klaus.alter += 1
print(klaus.alter)  # 40

print(klaus.beruf)  # "Anwalt"
klaus.beruf = "Koch"
print(klaus.beruf)  # "Koch"

print(klaus.__dict__)
# {'name': 'Klaus', 'alter': 40, 'beruf': 'Koch'}
```

- Fazit: In Python werden keine getter- und setter-Methoden benötigt!
  - Aber oft macht es Sinn damit zu arbeiten (dazu später mehr)

Methoden

- Regeln:
- Mind. 1 Parameter: self
- In der Klassendefinition immer self.<attribut> nutzen!
- self.name != name

```python
# Definieren der Klasse:
class Person:

    def __init__(self, name, alter, beruf):
        self.name = name
        self.alter = alter
        self.beruf = beruf

    def laufen(self):
        print(self.name, "läuft")

    def essen(self, gericht:str):
        print(self.name, "isst", gericht)

    def grüßen(self, other):
        print(self.name, "grüßt", other.name)

    def func():
        print("Das geht nicht!")

# Instanziieren von Objekten
klaus = Person(name="Klaus", alter=39, beruf="Anwalt")
laura = Person("Laura", 37, "Programmierer")

klaus.laufen()        # Klaus läuft
klaus.essen("Pizza") # Klaus isst Pizza
klaus.grüßen(laura)  # Klaus grüßt Laura
klaus.func()         # TypeError:
                     # Person.func() takes 0 positional arguments
                     # but 1 was given
```

Übung

1. Überführe den folgenden Sachverhalt in Python mit OOP:
   „Ein Onlineshop soll entwickelt werden. Dabei besteht der Shop aus einer Vielzahl von Artikeln. Jeder Artikel hat einen Namen und einen Preis. Diese werden beim Anlegen eines neuen Artikels festgelegt. Zusätzlich gibt es einen Warenkorb, welcher Artikel beinhalten kann. Es soll möglich sein, dem Warenkorb Artikel hinzuzufügen. Zusätzlich soll es möglich sein den Gesamtpreis aller Artikel des Warenkorbs zu berechnen.“

2. Erweitere das Programm laut folgender Anforderung:
   „Es sollen sich Elemente aus dem Warenkorb entfernen lassen. Dabei soll der Artikelname der zu entfernenden Elemente angegeben werden.“

Hausaufgabe

1. Überführe den folgenden Sachverhalt in Python mit OOP:
   „Ein Autohaus soll modelliert werden. Dieses besteht aus 3 Stellplätzen auf denen Autos stehen können. Zu jedem Auto ist die Marke, der Modellname und die aktuelle Tankfüllung (in Liter) gespeichert. Die Tankfüllung ist zu Beginn immer 100%. Die Autos können fahren, jedoch nur, wenn der Tank nicht leer ist. Dabei wird solange gefahren, bis der Tank leer ist.“
2. Erweitere das Programm um folgende Anforderung:
   „Der Chef des Autohauses kann die Autos auf den Stellplätzen tanken. Dazu wählt er einen Stellplatz und die zu tankende Menge in Liter aus. Sollte sich auf dem gewählten Stellplatz kein Auto befinden oder der Tank des Autos bereits voll sein, passiert nichts.“