



Alexander Jenke, Theodor Straube

## Übung6: Klassen

Programmierkurs Python // Mittwoch, 04. Dezember 2019

#### ToC

- OOP
- Klassen
  - Definition
  - Instanziierung
  - Vererbung
- Spezielle Funktionen
- Variablen-Sichtbarkeit





#### **OOP - Objektorientierte Programmierung**

- Die Programmarchitektur wird in Anlehnung an die tatsächliche Umgebung der Anwendung entworfen.
  - > Aufteilen des Problems in beteiligte Akteure und deren Aktionen
  - > Einzelne Aufgaben werden mit den benötigten Informationen gekapselt
- Mehrere Akteure können in unterschiedlichen Zuständen sein.
  - > Einzelne Instanzen sind nach dem Akteur-Bauplan aufgebaut aber selbstständig
- Objekte können untereinander Interagieren und Nachrichten austauschen
- Akteure werden Klassen
- Instanzen werden Objekte
- Aktionen werden Funktionen
- Informationen werden Variablen
- Nachrichten werden Funktionsaufrufe und Zugriffe auf Variablen anderer Objekte





# **Klassen Definition**

• Syntax: *class* gefolgt von *Klassenname* 

class Computer:
pass

- Alles was zur Klasse gehört wird darunter um eine Stufe eingerückt
- Klassen können enthalten:
  - Klassen
  - Funktionen
  - Variablen
- Funktionen in einer Klasse haben als erstes Argument `self`, welches das Objekt der Klasse enthält auf welchem die Funktion ausgeführt wird
- pass ist ein Platzhalter um Compiler-Fehler zu umgehen



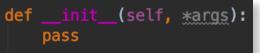


#### Klassen Instanziierung

Instanziierung mit Klassenname()



- In den Klammern werden die zur Initialisierung benötigten Parameter übergeben, so wie in der \_\_init\_\_ Funktion gefordert.
- die \_\_init\_\_-Funktion initialisiert neue Instanzen
  - self ist das neue Objekt welches initialisiert werden soll
  - \*args sind die hierfür benötigten Argumente, so wie von normalen Funktionen gewohnt







### Klassen Vererbung

 Klassen können von anderen Klassen erben und übernehmen damit alle objekt-unabhängigen Eigenschaften, wie class Laptop(Computer):
 pass

- Funktionen
- feste Variablen
- Funktionen der Oberklasse können überschrieben werden
- In der \_\_init\_\_-Funktion wird die \_\_init\_\_-Funktion der Oberklasse mit super().\_\_init\_\_() aufgerufen





#### **Spezielle Funktionen**

\_\_init\_\_\_ wird aufgerufen, wenn eine neue Instanz initialisiert wird

\_\_del\_\_\_ wird aufgerufen, wenn eine Instanz gelöscht wird

• \_\_str\_\_ wird aufgerufen, wenn print(*Instanz*) aufgerufen wird. Gibt lesbaren String zurück

• \_\_repr\_\_ gibt einen String zurück der möglichst viele Informationen über die Instanz enthält

\_\_getitem\_\_\_ gibt das Element am geforderten Index zurück, wenn die mehrere Elemente verwaltet werden

• \_\_len\_\_ gibt die Menge der verwalteten Elemente zurück

\_\_call\_\_\_ wird aufgerufen, wenn die Instanz als Funktion aufgerufen wird

• \_\_getattr\_\_ wird aufgerufen, wenn das Attribut nicht an den üblichen Orten gefunden werden kann.

\_\_add\_\_ / \_\_sub\_\_ / \_\_mul\_\_ / \_\_div\_\_ implementiert die + - \* / Operationen





#### Variablen-Sichtbarkeit

- Auf Variablen außerhalb des Scopes kann zugegriffen werden, wenn diese mit dem Keyword global zugänglich gemacht wurden
- Variablen deren Namen mit einem Unterstrich beginnen sind protected
   d.h. sie können von au

  ßerhalb der Klasse nur gelesen, aber nicht beschrieben werden
- Variablen und Funktionen deren Namen mit zwei Unterstrichen beginnen sind private d.h. sie können von außerhalb der Klasse weder gelesen noch geschrieben werden





#### nächste Woche

- Mehr Zeit für Aufgaben
- Zeit für Fragen
- Lösungen



