→ Klassen ~ ähnlich wie in C++

Bsp. https://www.w3schools.com/python/python_classes.asp

→ magic methods (~ wie operator overloading)

Aufzählung: https://www.tutorialsteacher.com/python/magic-methods-in-python

Bsp. https://www.tutorialspoint.com/python-program-to-define-class-for-complex-number-objects
https://www.analyticsvidhya.com/blog/2021/08/explore-the-magic-methods-in-python/

28.6.
müsst ihr nicht können, einfach der Vollst.heit halber

→ class vs. instance Variables:

Class Variables — Declared inside the class definition (but outside any of the instance methods). They are not tied to any particular object of the class, hence shared across all the objects of the class. Modifying a class variable affects all objects instance at the same time.

Instance Variable — Declared inside the constructor method of class (the `__init__` method). They are tied to the particular object instance of the class, hence the contents of an instance variable are completely independent from one object instance to the other.

https://medium.com/python-features/class-vs-instance-variables-8d452e9abcbd

Bsp. https://www.digitalocean.com/community/tutorials/understanding-class-and-instance-variables-in-python-3

→ für eure Zwecke könnt ihr probs immer instance variables machen, d.h. in __init__ definieren

→ self → wie this pointer in C++, ist immer das erste Argument irgendeiner Fkt in einer Klasse

The `self` parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named `self` , you can call it whatever you like, but it has to be the first parameter of any function in the class:

Example

Use the words *mysillyobject* and *abc* instead of *self*:

```
class Person:
  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age

  def myfunc(abc):
    print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

→ Inheritance

Gute Übersicht https://www.geeksforgeeks.org/inheritance-in-python/

→ wenn ihr magic methods implementiert und benutzen wollt, achtet darauf welche nun gerufen wird / werden soll. (e.g. Square oder Rectangle vom Übungs Bsp.)

→ Konzepte
- ↳ interpreted vs. compiled
- ↳ dynamic vs. static timing, type hinting
- ↳ functional programming
- ↳ generic programming

→ lambda `lambda arguments : expression`
- ↳ Erklärung + Bsp. https://www.w3schools.com/python/python_lambda.asp