

Beispiel: Klasse

```
# Klassendefinition
class Human:
    # Konstruktor (Initialisierung)
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # Methode zum Vorstellen
    def __str__(self):
        s = "Ich heiÙe " + self.name +
        ".\n "
        s += "Ich bin " + str(self.age) + "
        Jahre alt.\n "
        return s

    # setter-Methode zum Ändern des Attributes von age
    def set_age(self, new_age):
        self.age = new_age

# Deklaration und Initialisierung einer Human-Instanz
h1 = Human("Max Müller", 21)
# Ändern der Wertbelegung des Attributes age vom
Objekt h1
h1.set_age(22)
# __str__-Methode wird aufgerufen
print(h1)
```

Exkurs: Python Dataclasses

```
class Robot_traditional:
    def __init__(self, model, serial_number,
manufacterer):
        self.model = model
        self.serial_number = serial_
_number
        self.manufacterer = manufa
cturer

# versus
from dataclasses import dataclass
@dataclass
class Robot:
    model: str
    serial_number: str
    manufacterer: str
```

Magic Methods einer Klasse

Operator/Aufruf	Magic Method
obj = Klasse()	__new__ und __init__
+	object.__add__(self, other)
-	object.__sub__(self, other)
str(object) oder print(object)	object.__str__(self)
<	object.__lt__(self, other)
==	object.__eq__(self, other)
>=	object.__ge__(self, other)

Die Auflistung zeigt nur eine Auswahl der zur Verfügung stehenden Methoden einer Klasse. Die gesamte Liste (eines Scopes) erhält man beim Aufruf von `dir(object)`.

Hintergrund: Magic Methods

Der Mechanismus: Ein Ausdruck "x + y" mit x und y als Instanzen der Klasse K, wird so ausgewertet: der Interpreter prüft die Klassendefinition von K. Wenn K eine Methode `__add__` hat, so wird diese aufgerufen mit `x.__add__(y)`.

Type Hints

```
class Position:
    def __init__(self, x: int, y: int) ->
None:
        self.x = x
        self.y = y
    def __add__(self, other: Position) ->
Position:
        return Position(self.x +
other.x, self.y + other.y)
```

Kernkonzepte OOP

Datenkapselung	Attributen und Methoden können unterschiedliche Sichtbarkeitsniveaus zugewiesen werden (+ public, o protected, - private)
Vererbung	aus Oberklassen können Unterklassen abgeleitet werden, die zunächst gleiche Attribute und Methoden besitzen
Polymorphie (Vielgestaltigkeit)	zusätzlich zu vererbten Attributen und Methoden existieren weitere; vererbte Methoden können überladen werden; Objekte können bei Instanziierung unterschiedliche Strukturen annehmen

C

By **ch1ph0rst** (ch1ph0rst)
cheatography.com/ch1ph0rst/

Not published yet.
Last updated 25th March, 2026.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>