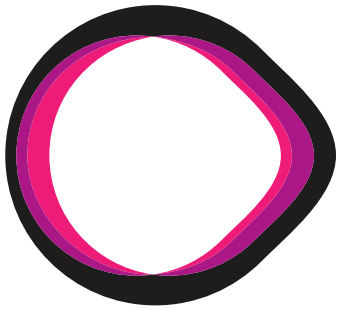
# Python – Lektion 7
# Klassen & Vererbung



OST
Ostschweizer
Fachhochschule

# Rückblick

- Klassen-Definition
  `class`
- Datenabstraktion:
  Setter/Getter, Property, public/protected/private
- Methoden und Variablen:
  `__init__()`, magische Methoden

# Rückblick

```
class Konto:
    '''Diese Klasse stellt ein Bankkonto dar.'''
    zinssatz = 0.15

    def __init__(self, inhaber, kontonummer, kontostand):
        '''Diese Methode initialisiert die Variablen.'''
        self.inhaber = inhaber
        self.kontonummer = kontonummer
        self.__kontostand = kontostand

    @property
    def kontostand(self):
        print("Der Kontostand wurde abgefragt.")
        return self.__kontostand

    @kontostand.setter
    def kontostand(self, n):
        self.__kontostand = n
        print(f"Der Kontostand wurde auf {self.__kontostand} geändert.")

if __name__ == "__main__":
    konto = Konto("Peter Müller", "9-7-8-6", 1000)
    konto.kontostand = 10000
```

# Heutige Themen

- Magische Methoden
- Vererbung
- Mehrfachvererbung

# Magische Methoden

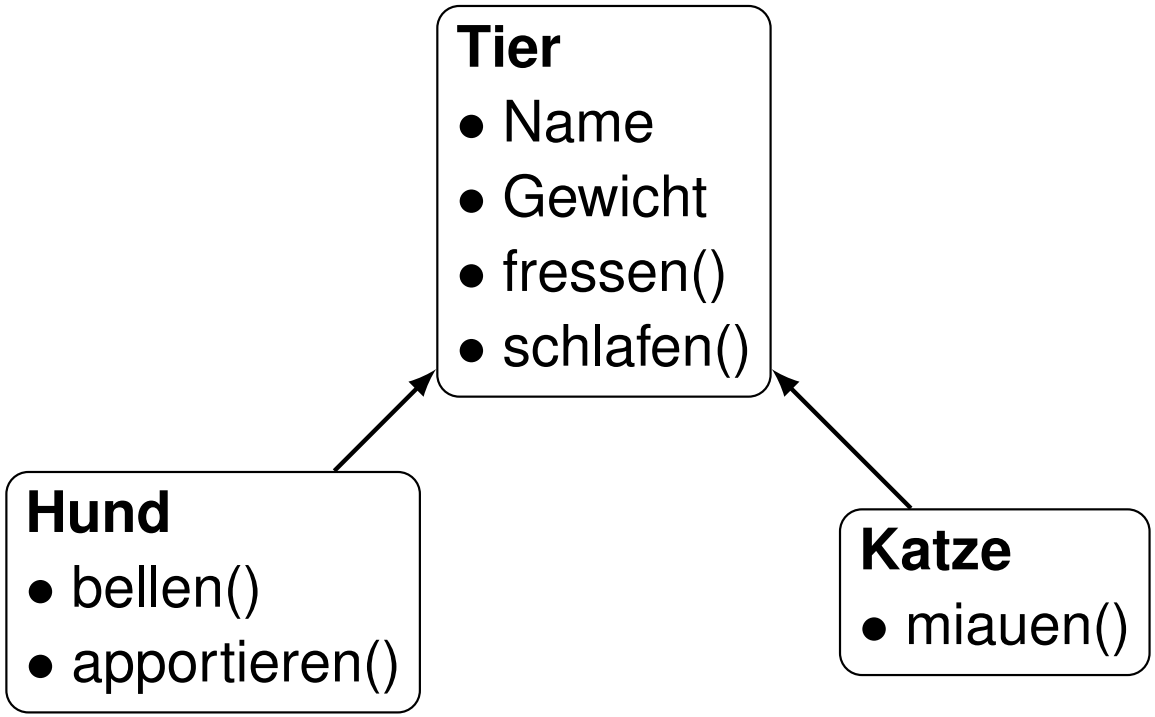
- Besondere Fähigkeiten[1] für Klassen
- Grundfunktionen
  - `__init__()`, `__del__()`, `__str__()`, ...
- Operatoren überladen
  - Arithmetische Operatoren: `+ - / * %` ...
  - Numerische Operatoren: `__int__()`, `__float__()`, `__abs__()`, ...
  - ...
- Containertypen emulieren
  - `__len__()`, `__iter__()`, `__contains__()`, ...
- ...

`http://localhost:8888/notebooks/magische_methoden.ipynb`

[1] `https://docs.python.org/3/reference/datamodel.html#special-method-names`

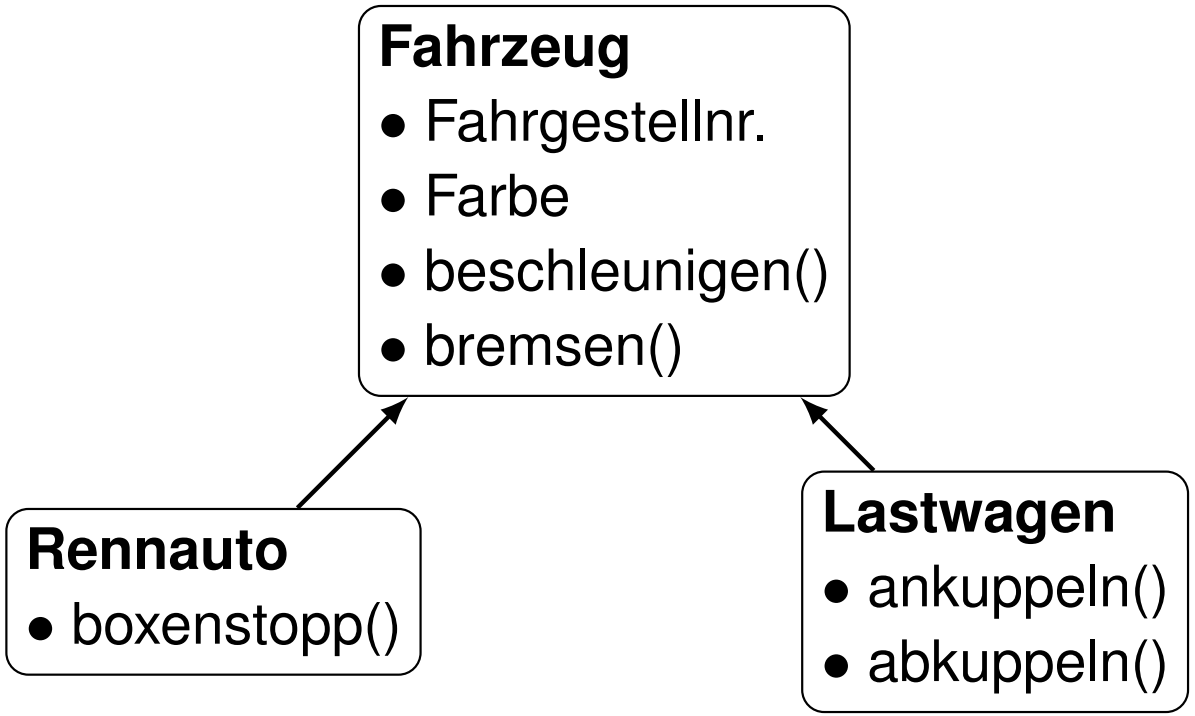
# Vererbung

- Eine neue Klasse aus einer bestehenden Klasse ableiten:



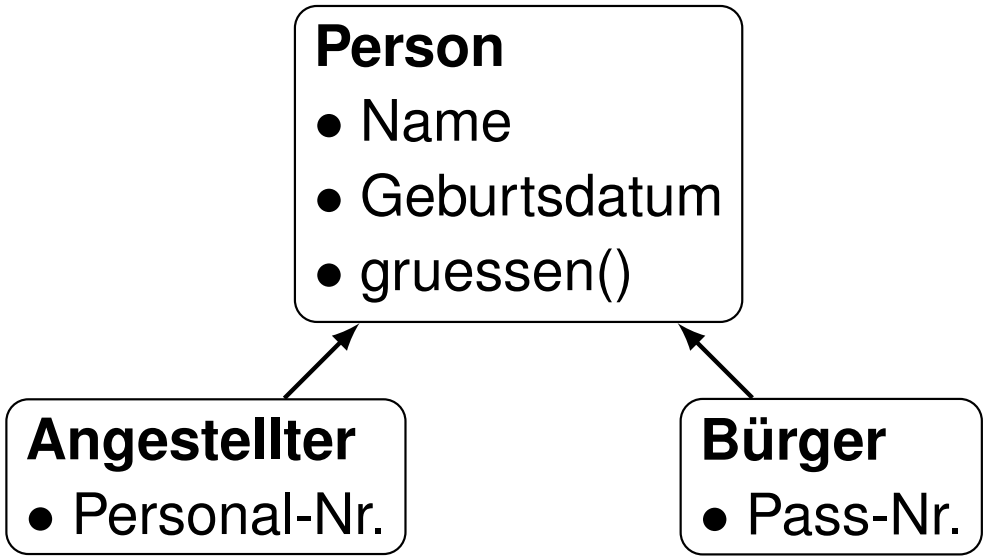
# Vererbung

- Eine neue Klasse aus einer bestehenden Klasse ableiten:



# Vererbung

- Eine neue Klasse aus einer bestehenden Klasse ableiten:

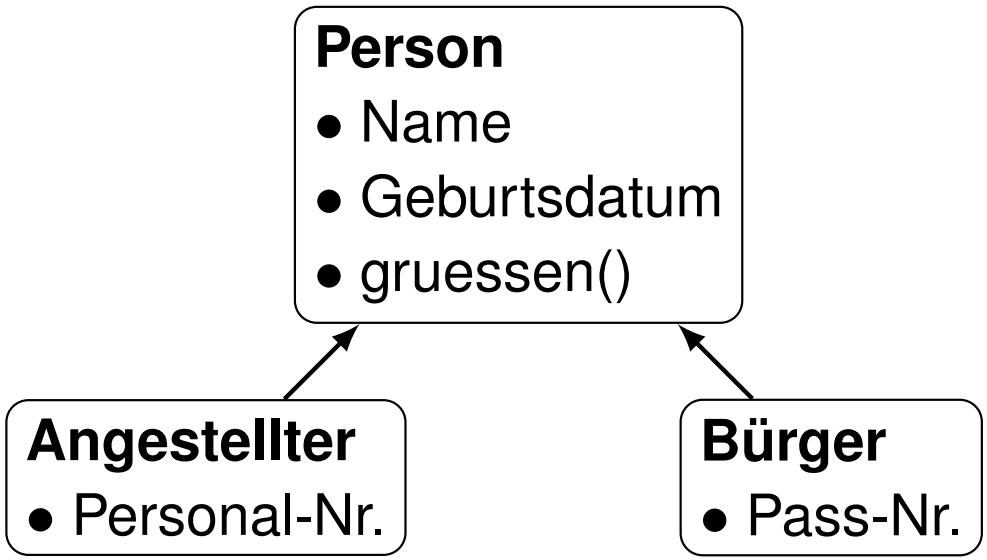



- **Person** ist eine:
  Oberklasse, Basisklasse, Elternklasse oder Superklasse

- **Angestellter** und **Bürger** sind eine:
  Unterklasse, abgeleitete Klasse, Kindklasse oder Subklasse

# Vererbung

- Eine neue Klasse aus einer bestehenden Klasse ableiten:

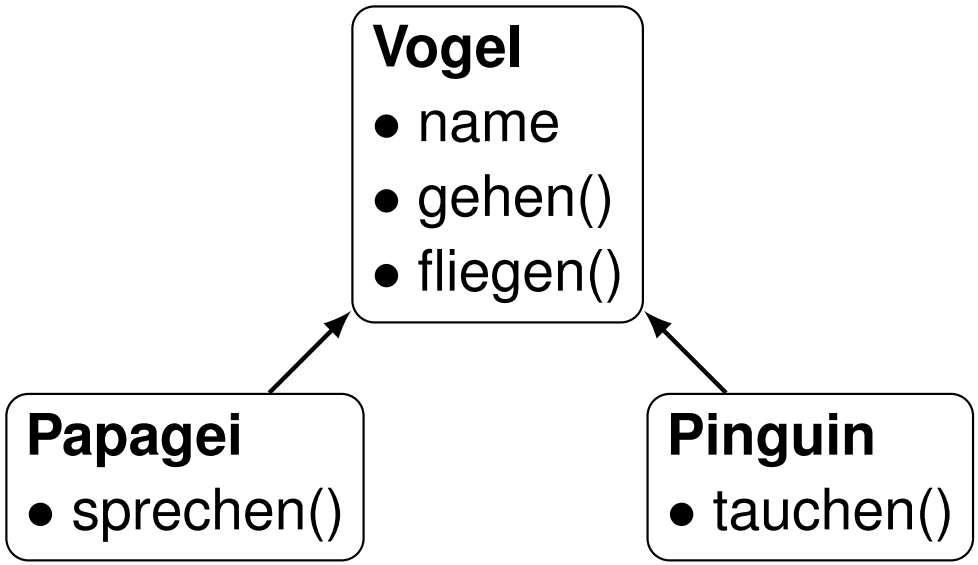



- **ist-eine** Beziehung zwischen Super- und Subklasse:
  - Angstellter **ist eine** Person
  - Bürger **ist eine** Person

- **Liskovsches Substitutionsprinzip**:
  Jedes Programm, welches mit Objekten der Superklasse funktioniert, sollte auch mit Objekten der Subklasse funktionieren.

# Vererbung

- Verletzung des Liskovschen Substitutionsprinzip:

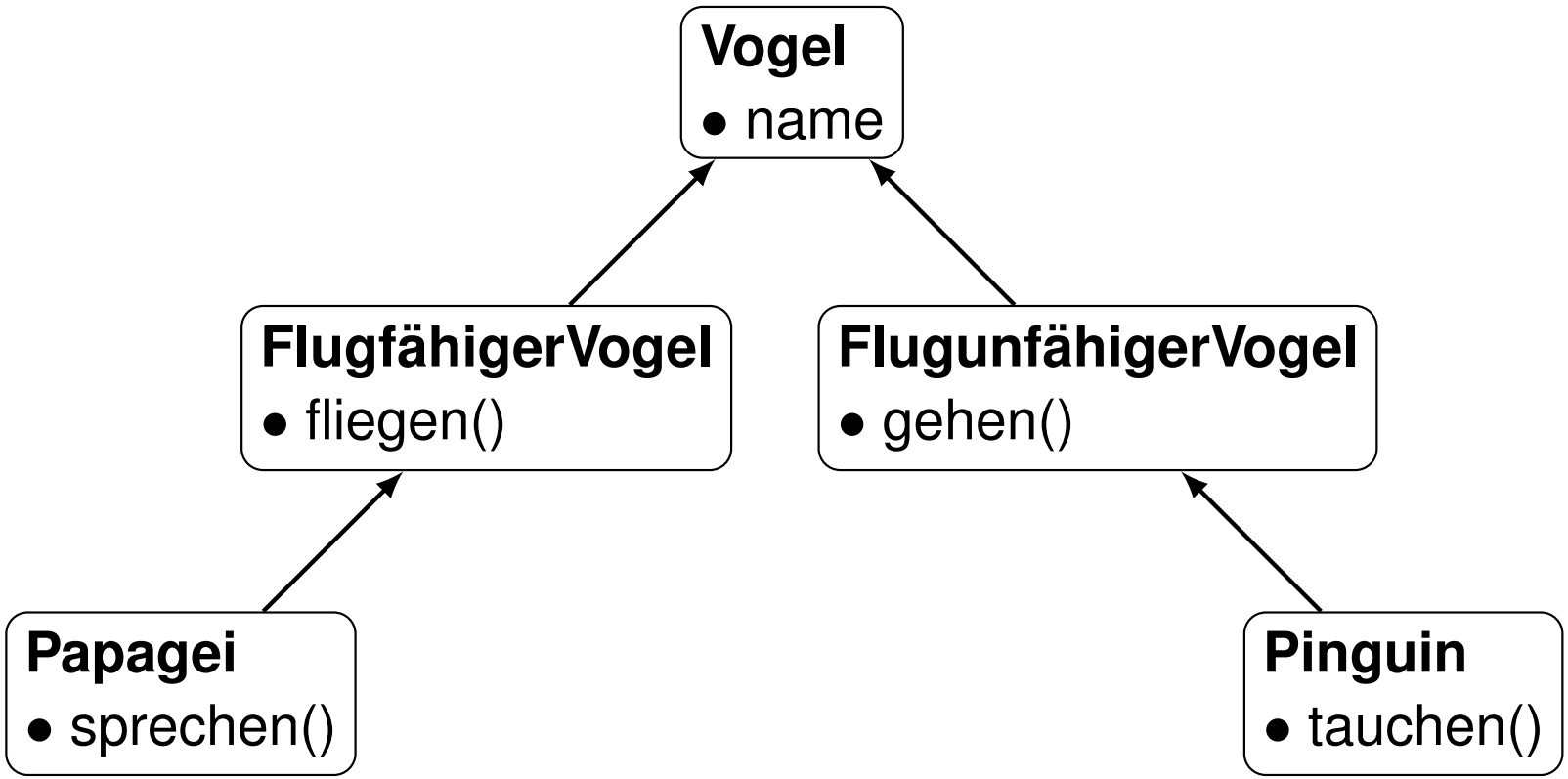



- **ist-eine** Beziehung zwischen Super- und Subklasse:
  - Papagei **ist ein** Vogel
  - Pinguin **ist ein** Vogel

- **Liskovsches Substitutionsprinzip**:
  Jedes Programm, welches mit Objekten der Superklasse funktioniert, sollte auch mit Objekten der Subklasse funktionieren.

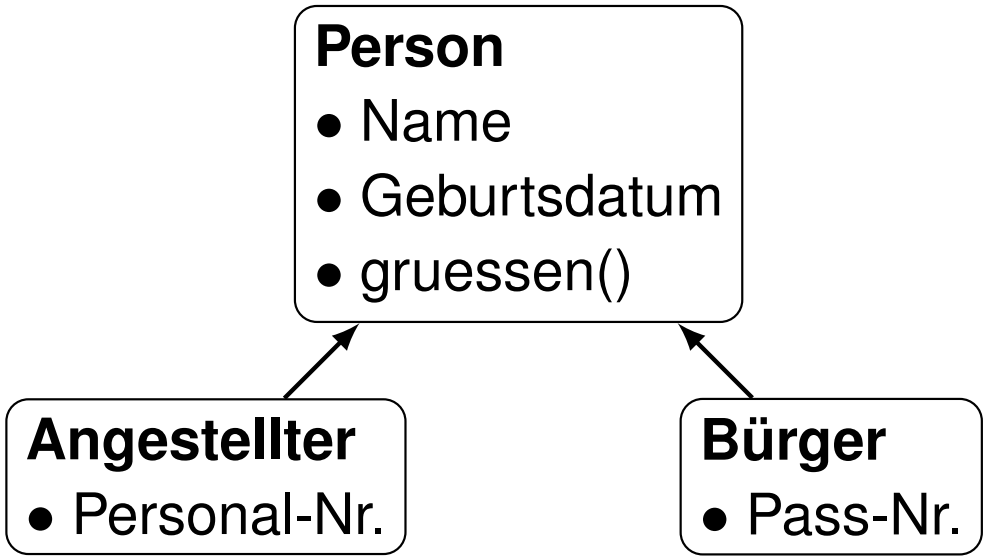
# Vererbung

- Zwischenklassen helfen das Substitutionsprinzip zu erfüllen:



# Vererbung

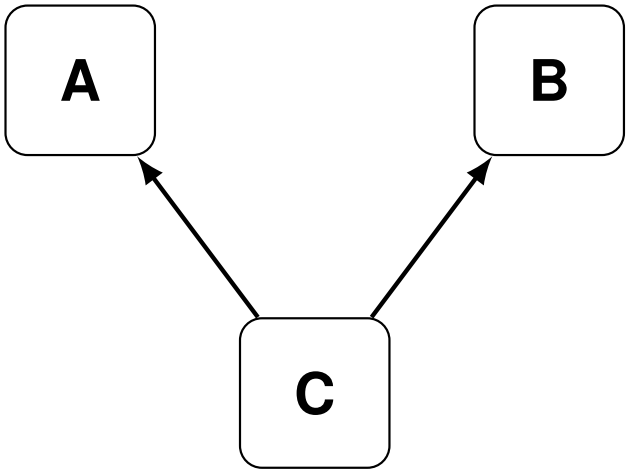
- Eine neue Klasse aus einer bestehenden Klasse ableiten:




`http://localhost:8888/notebooks/vererbung.ipynb`

# Mehrfachvererbung

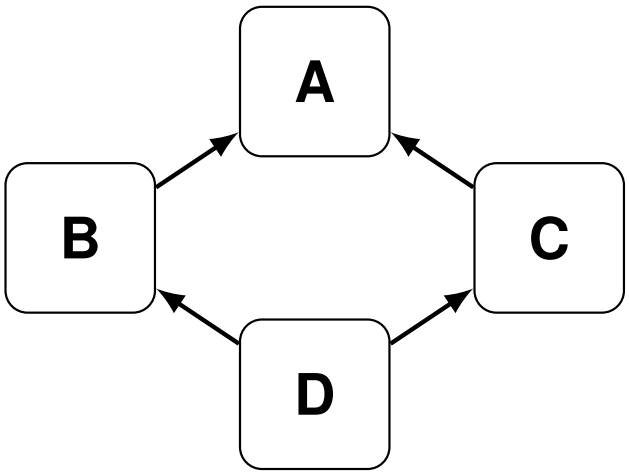
- Eine Subklasse kann von mehreren Superklassen erben:




```
class A:
    pass

class B:
    pass

class C(A, B):
    pass
```

# Mehrfachvererbung

- Eine Subklasse kann von mehreren Superklassen erben:

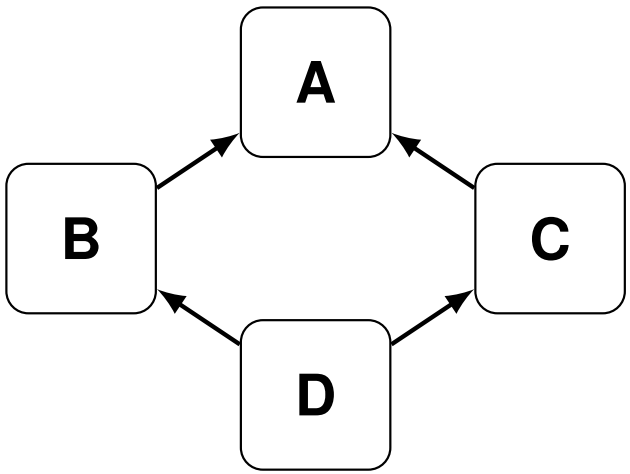



```
class A:
    pass

class B(A):
    pass

class C(A):
    pass

class D(B, C):
    pass
```

http://localhost:
8888/notebooks/method_resolution_order.ipynb

# Mehrfachvererbung

- `super()` ruft automatisch die Methode der nächsten Klasse auf
- Method Resolution Order (MRO) → C3 Superclass Linearization[2]
- Diamond-Problem ist kein Problem mit `super()`




`http://localhost:8888/notebooks/mehrfachvererbung.ipynb`

[2] `https://en.wikipedia.org/wiki/C3_linearization`