JAVACREAM

Training
Consulting
Projectmanagement

Python

Kurze Vorstellung

JAVACREAM
Training
Consulting
Projectmanagement

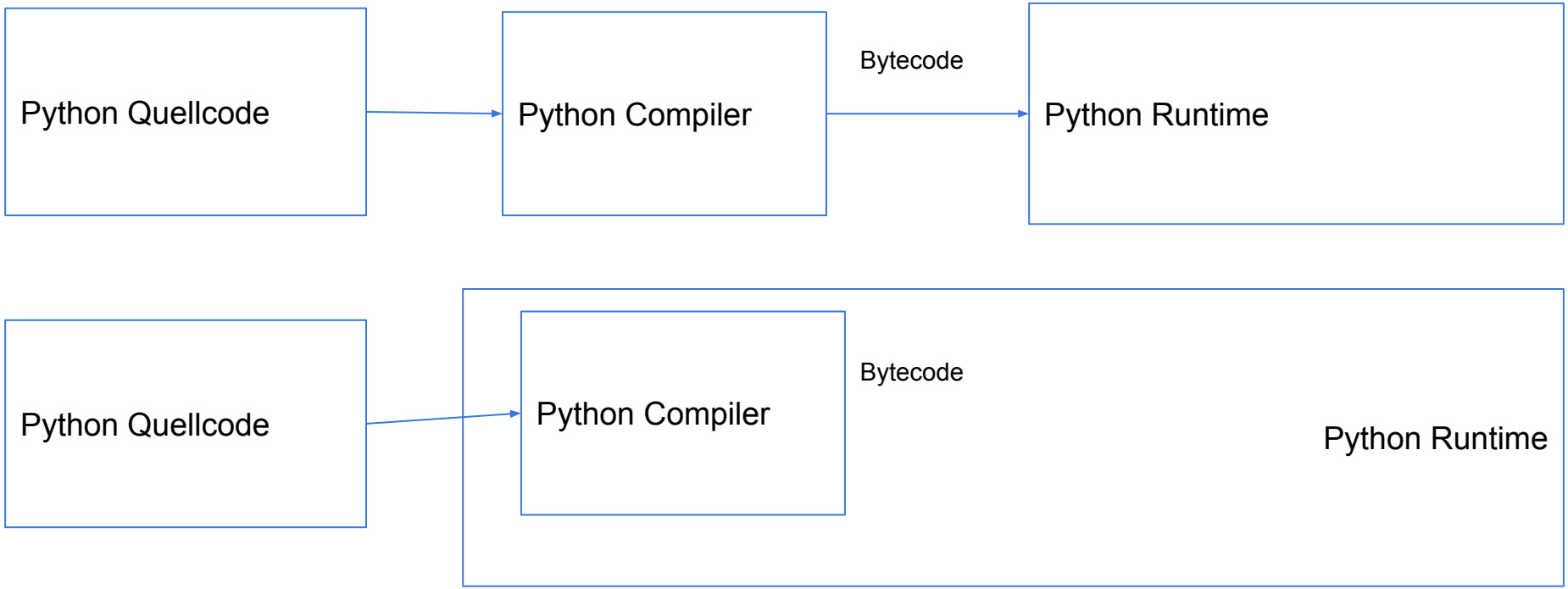
- Name
- Rolle im Unternehmen
- Themenbezogene Vorkenntnisse
- Aktuelle Problemstellung
- Konkrete individuelle Zielsetzung

Übersicht

Was ist Python?

- Python Runtime
  - Häufig genannt “der Python Interpreter”
  - Implementierungen für verschiedene Plattformen / Betriebssysteme sind vorhanden
    - Damit ist Python “Plattform-unabhängig”
    - Es gibt im Gegensatz zu z.B. Java keine Spezifikation
      - Dieser Effekt spielt heute keine große Rolle mehr
- Programmiersprache Python
  - Eigentlich eine “ganz normale” Programmiersprache
  - Einordnung
    - Sequenzielle Programm-Ausführung
    - Laufzeit-Typisiert, optional: eine statische Typisierung ist möglich
    - Objekt-orientierte

Details zum Python Interpreter

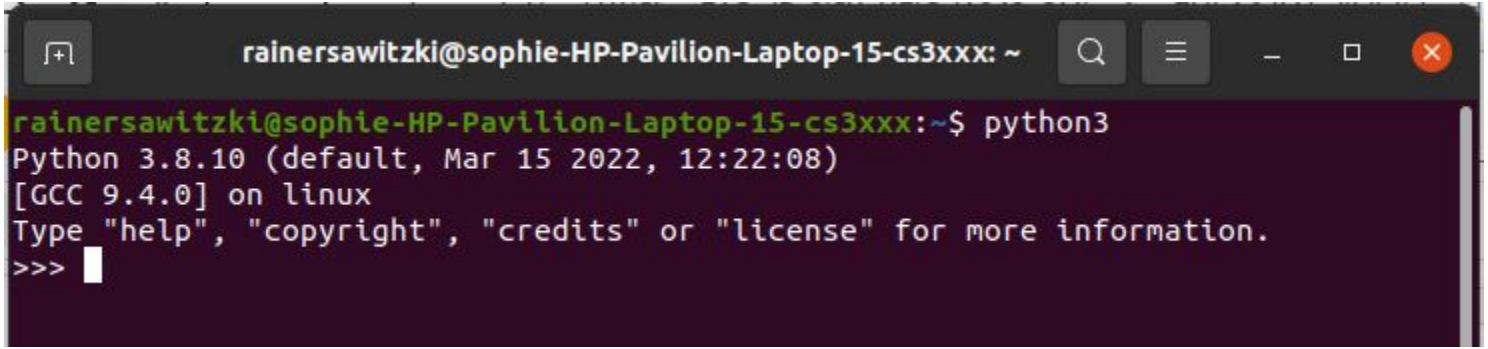


Infrastruktur

- Installation einer Python Runtime
  - Native Installation auf einem Betriebssystem

  - Verwendung eines Docker-Containers

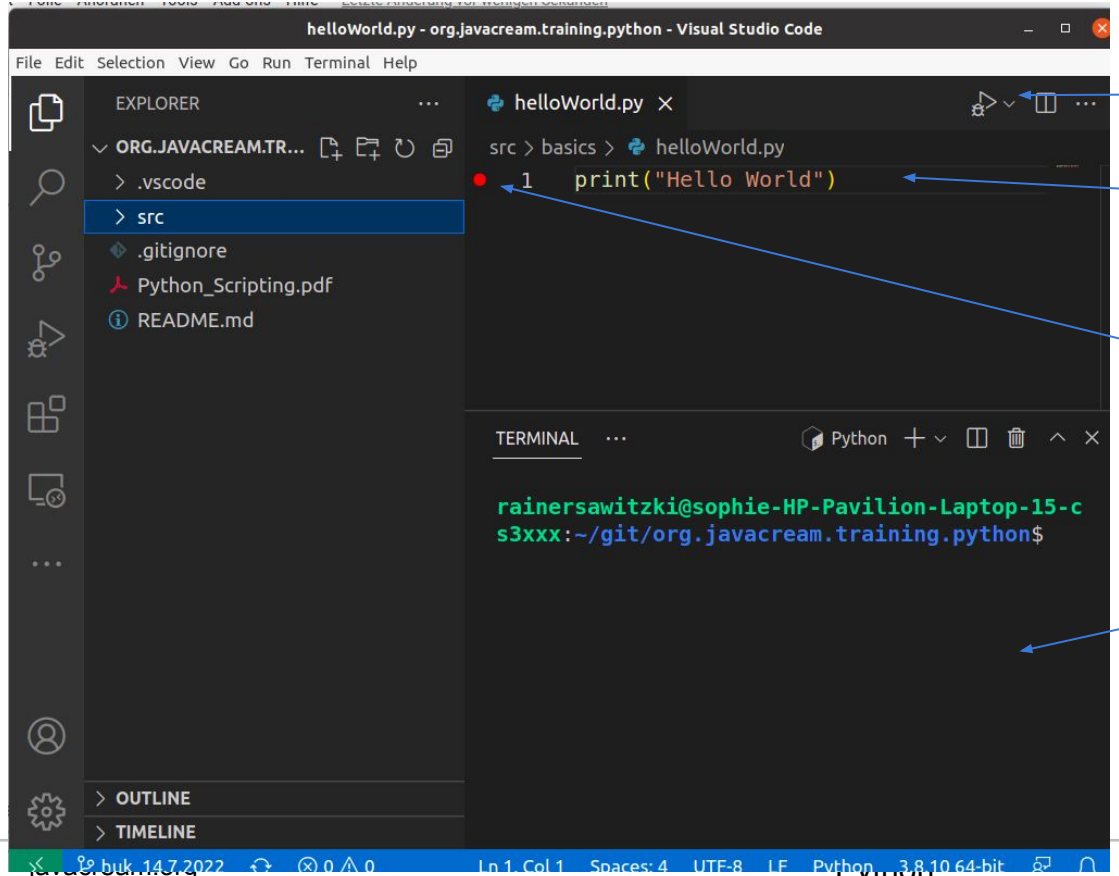
Check


rainersawitzki@sophie-HP-Pavilion-Laptop-15-cs3xxx: ~
rainersawitzki@sophie-HP-Pavilion-Laptop-15-cs3xxx:~$ python3
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

Python Development

- Entwicklungsumgebung
- **Visual Studio Code**
  - Alternativ: Eclipse-basierte Lösung, Visual Studio, IntelliJ
-

VSC



Starten / Debuggen eines Python Programms

Editor mit Syntax-Highlighting und Code Assist

Haltepunkt gesetzt “im Quellcode”

Konsole / Terminal

Python First Contact

Datentypen

- Literale
  - Zeichenketten
    - "...", '...'
  - Zahlen
    - 9, 4.2, 4.2e-9
  - Logische Werte, Booleans
    - True, False
  - + ein paar mehr…
-

Python als (fast) normale Programmiersprache

- Operatoren
  - Mathematische Operatoren
    - +, -, *, /, %
  - Zeichenketten-Konkatination
    - +
  - Vergleichs-Operatoren mit logischem Rückgabewert
    - >, <, >=, <=, ==, !=
  - Logische Operatoren
    - and, or, not, is, is not
- Kontrollstrukturen
  - If - Else
    - Arbeitet mit Einrückungen
    - if, else, elif
  - ab Python 3.10: match, das entspricht einem switch-case

Datentypen continued

- Tupel
  - Eine geordnete, unveränderbare Liste mit erlaubten Duplikaten
  - Literal
    - tuple = (element1, element2, element3)
    - value = tuple[1] # -> element2
    - #tuple[3] = element4 #Fehler, Tupels sind unveränderbar
- List
  - Eine geordnete, veränderbare Liste mit erlaubten Duplikaten
  - Literal
    - list = [element1, element2, element3]
    - value = list[1]
    - list[3] = element4
    - list[1] = element5

Datentypen continued

- Set
  - Eine ungeordnete, veränderbare Liste ohne Duplikate
  - Literal
    - set = {element1, element2, element3}
    - Zugriff auf Elemente nur über Iteration
- Dictionary
  - Eine ungeordnete, veränderbare Liste mit key=value-Paaren ohne Key-Duplikate
    - dictionary = {“spring”: “ok”, “summer”: “great”, “autumn”: “not so great”, “winter”: “depressive”}
    - Zugriff auf Elemente erfolgt mit dem key
      - value = dictionary[“summer”] #-> great

ab Python 3.x bleibt die Ordnung der keys erhalten

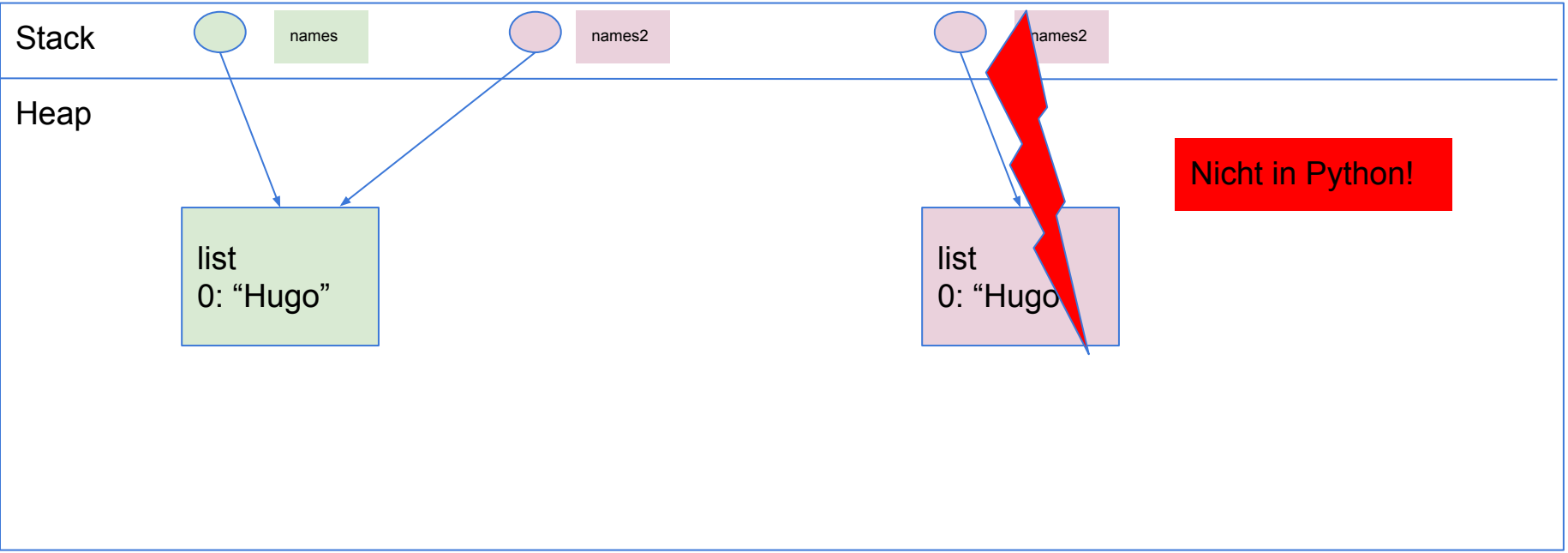
Speichermodell in Python


```
names = [“Hugo”]
```

Eine Variable in Python ist immer einer Referenz auf ein Objekt im Heap

```
names2 = names
```

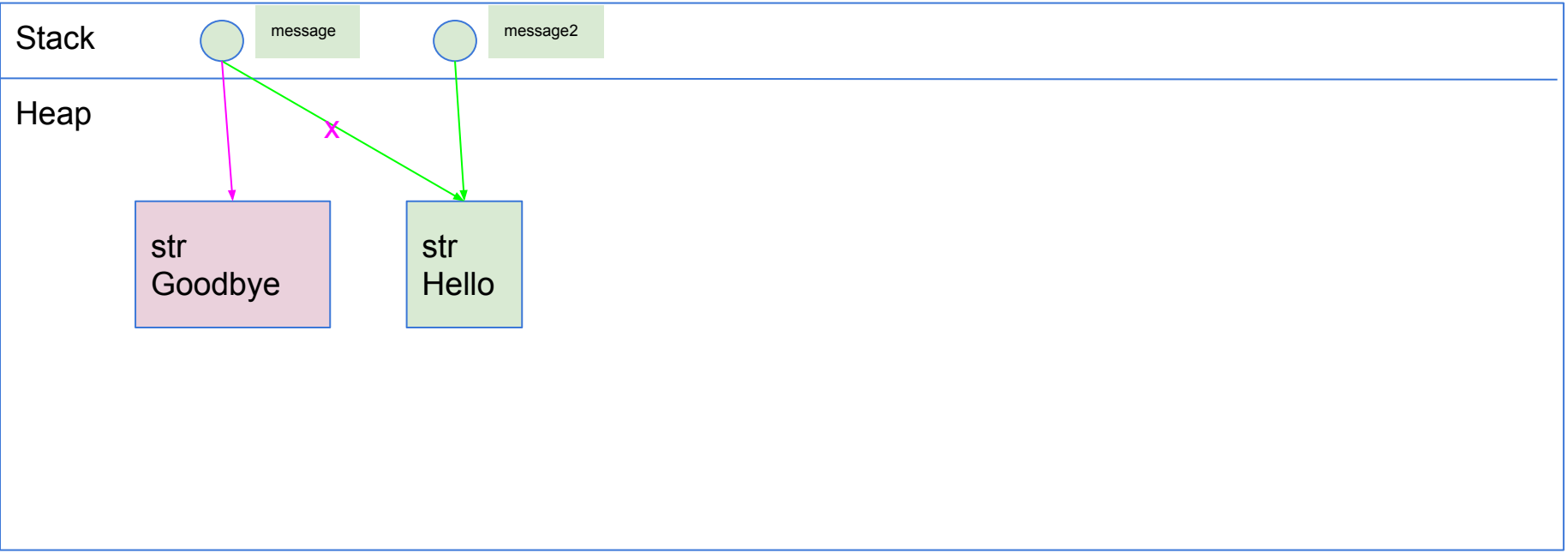
Eine Zuweisung in Python ist immer einer Kopie des Wertes der Referenz



Speichermodell


```
message = “Hello”
message2 = message
```

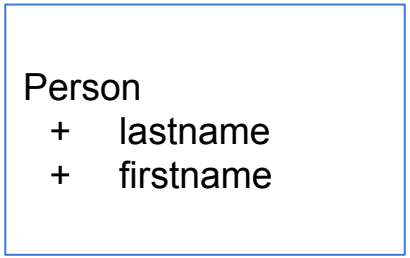
```
message = “Goodbye”
```



Namenskonventionen in Python

- Variablen und Funktionsnamen nur Kleinbuchstaben
- Zur besseren Lesbarkeit: snake-Konventionen
  - nicht personname, sondern person_name
    - Andere Sprachen: camel-Case, personName
- Benutzerdefinierte Datentypen (-> etwas später) beginnen mit einem Großbuchstaben und sind Came Case

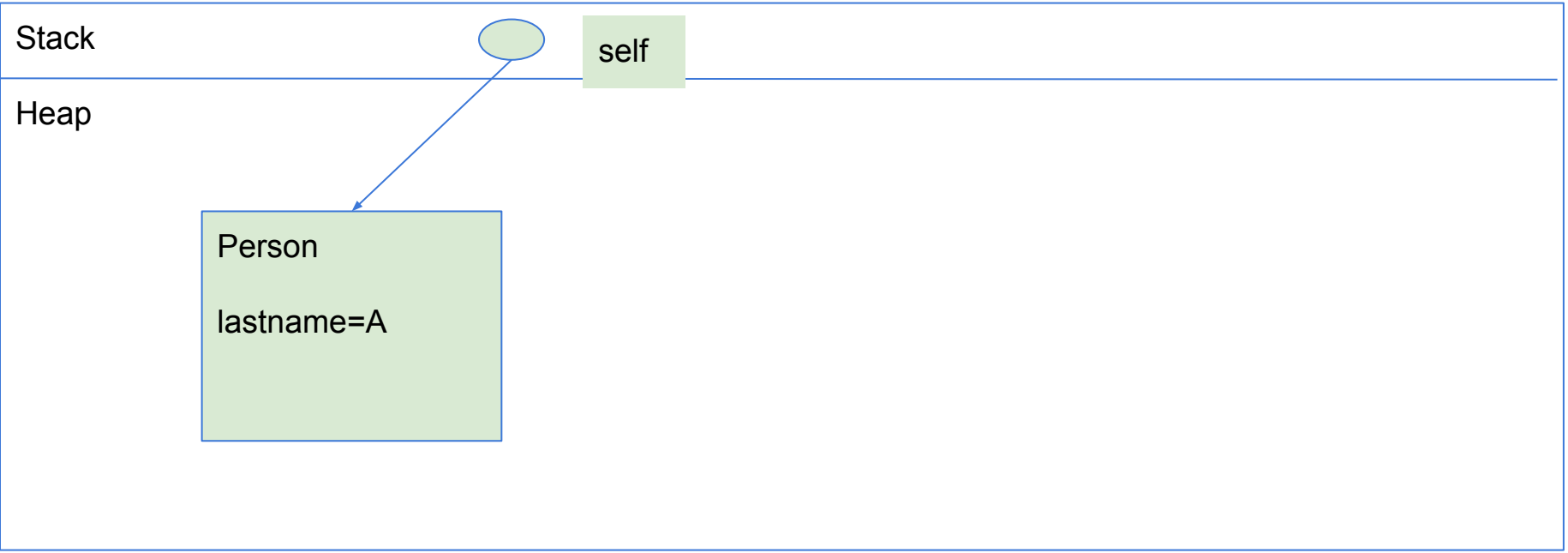
Modellierung einer Anwendung



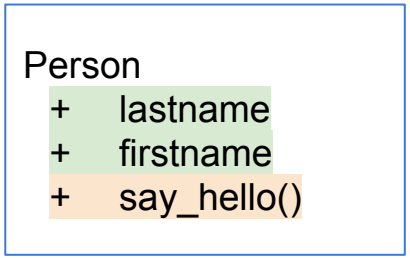
Speichermodell und self

Person(“A”, “B”)

```
__init__(lastname, firstname):
        self.lastname = lastname
```



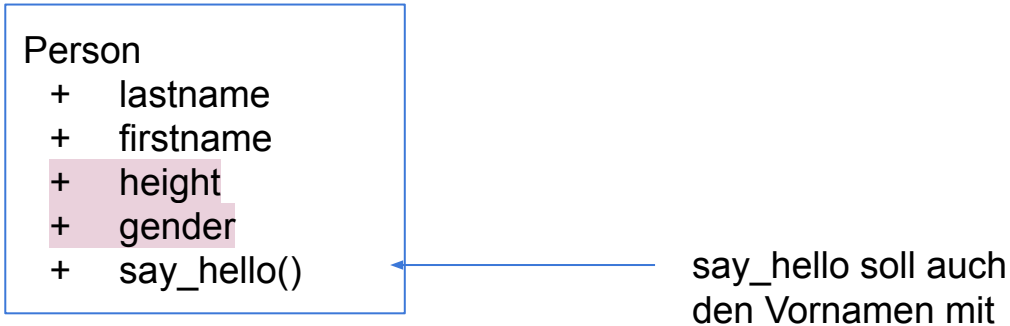
Modellierung einer Anwendung



Attribute
Fields

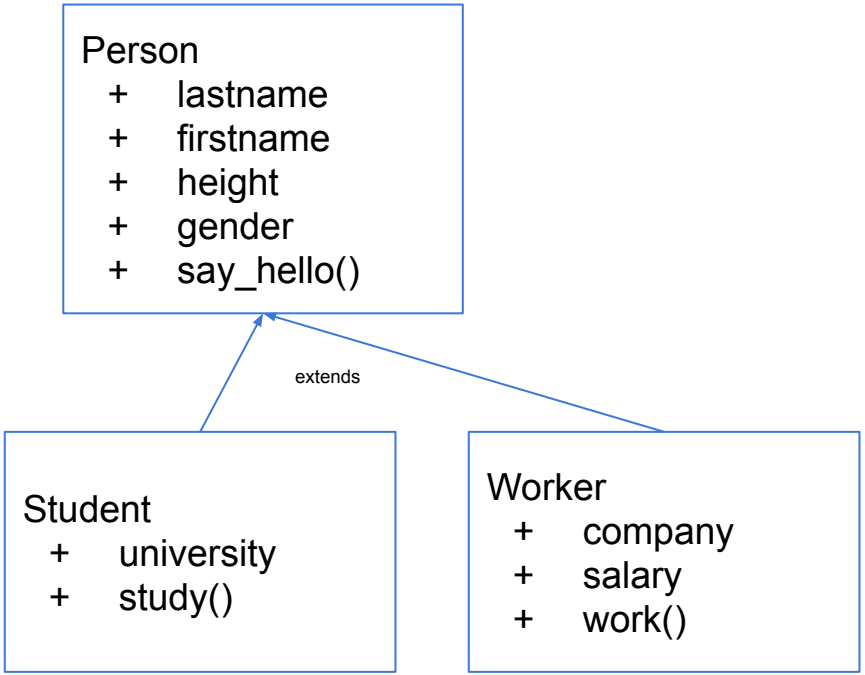
Methods

Modellierung einer Anwendung



ToDo: Umsetzung dieses Modells

Vererbung



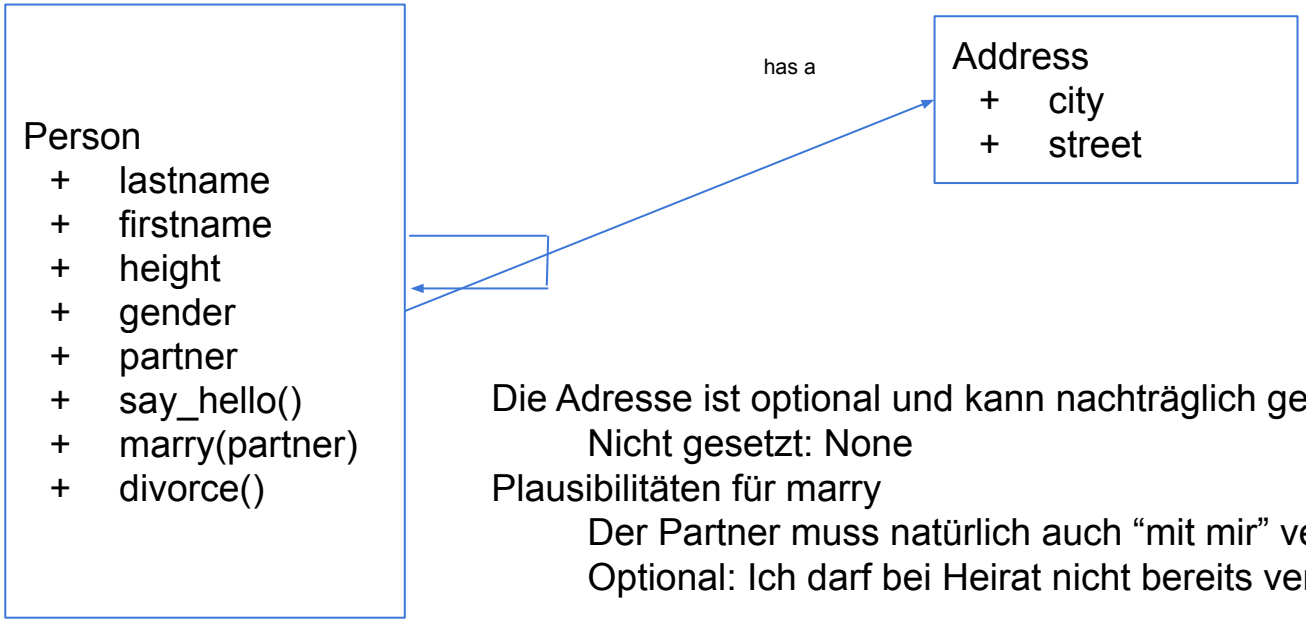
Python-Scratch

```
class Person:
        …

class Student(Person):
        …

class Worker(Person):
        …
```

ToDo Extended Modell



Die Adresse ist optional und kann nachträglich gesetzt werden
- Nicht gesetzt: None

Plausibilitäten für marry
- Der Partner muss natürlich auch “mit mir” verheiratet werden
- Optional: Ich darf bei Heirat nicht bereits verheiratet sein
- …

- Wie konsequent ist die Objekt-Orientierung umgesetzt?
  - Arbeiten mit Referenzen ist umgesetzt
    - Verfolgen der Referenzen erfolgt über den '.'-Operator
  - Python-Runtime implementiert einen Garbage Collector
  - (Polymorphes Verhalten ist umgesetzt)
  - "Alles" ist ein Objekt
    - Built-In-Functions von Python haben keinen Bezug zu Objekten
      - z.B. zur Bestimmung der Länge einer Liste gibt es die Built-In function len(list)
        - oop: list.length
      - Übersicht
        - https://www.w3schools.com/python/python_ref_functions.asp
  - Python stellt eine umfangreiche Klassenbibliothek zur Verfügung
    - Typische Funktionalitäten / Algorithmen werden als Methoden von Klassen bereitgestellt
      - z.B. String https://www.w3schools.com/python/ref_string_capitalize.asp

Todo

- Stellen Sie die marry / divorce auf einen dynamische Stil um
  - hasattr, delattr
- Verschaffen Sie sich einen Überblick
  - built-ins
  - String-Methoden
- range()-Funktion zum Erzeugen eines Range-Objektes z.B. zum Iterieren über einen Wertebereich
  - for (i= 0; i < 10; i = i + 1) -> Formulierung mit range
-