Einführung

Beispiele:

`HumanBeing.py`

Objektorientierte Programmierung

- **Python**: "*Python is a dynamic object-oriented programming language that can be used for many kinds of software development.*" (www.python.org)

- Warum Objektorientiert?
  - Mächtiges Werkzeug für den Programmierer.
    - Verständnis der Internas von Python
    - Entwurf von intuitiven Klassen (Operatoren Überladung)
  - Strukturmittel für den Designer.
    - Entwurf von Bibliotheken und Frameworks
    - Etablierung von "Lesson Learned" in Design Pattern
  - Begrifflichkeit für den Systemadministrator.
    - Verständnis der Verzeichnisdiensten LDAP
    - Besserer Umgang mit Windows Power Shell

Klasse versus Objekt

- **Klasse**: Eine Klasse beschreibt die Methoden und Variablen einer Menge gleichartiger Objekte.
- **Objekt**: Ein Objekt ist eine konkrete Ausprägung einer Klasse zur Laufzeit eines Programmes.

Klasse	Objekt
`int`	`5 2011`
`float`	`2.71 -5.3`
`string`	`"Teststring"`
`list`	`["a",123,"Teststring"]`
`dict`	`{"eins":1,"zwei":2,"drei":3}`

Während eine Klasse die Struktur von Objekten beschreibt, sind Objekte Instanzen der Klasse, die zur Laufzeit zur Verfügung stehen.

Konzepte

Charakteristisch für die Objektorientierte Programmierung sind die drei Konzepte **Kapselung**, **Vererbung** und **Polymorphie**.

- **Kapselung**: Eine Klasse bindet die Datenstrukturen und Methoden, auf den sie agiert.
- **Verbergung**: Klassen können durch Vererbung weiter spezialisiert werden.
- **Polymorphie**: Ist die Eigenschaft eines Bezeichner, sich abhängig vom konkreten Objekt verschieden zu verhalten.

Kapselung, Vererbung und Polymorphie sind die kleinste gemeinsame Schnittmenge der Objektorientierten Programmiersprachen.

Kapselung

Die Klasse

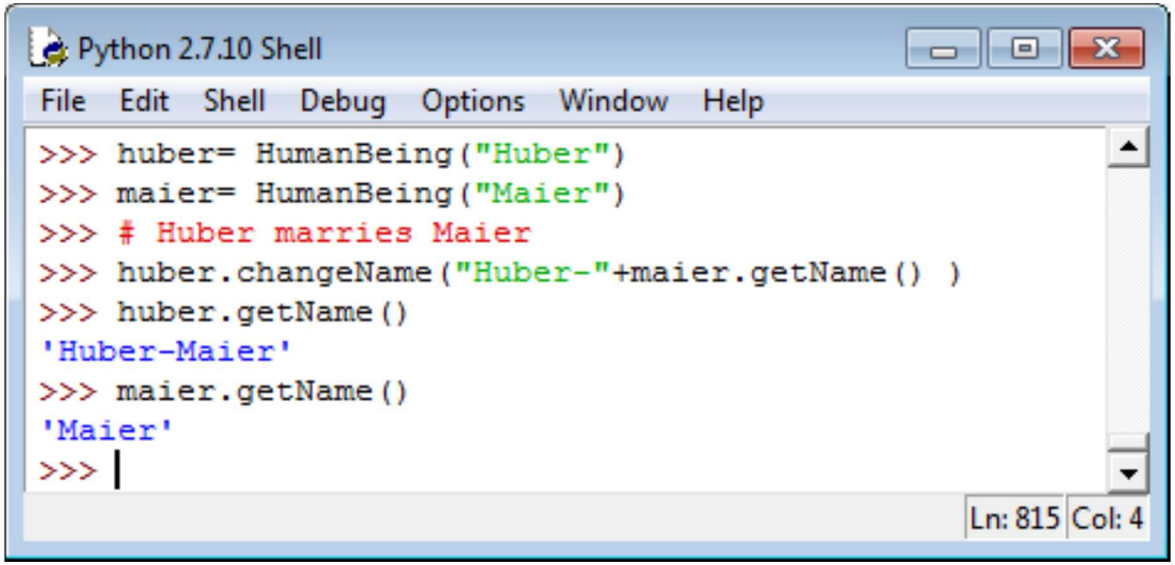
- kapselt die Datenstrukturen und Methoden in ihrem Klassenkörper.
- stellt mittels Methoden den kontrollierten Zugriff auf ihre Internas zur Verfügung.
- versteckt in ihrer Implementierung Details ihrer Implementierung vor der Außenwelt.

➡ Trennung von Interface und Implementierung

Die Eigenschaft der Klasse, ihre Attribute von der Aussenwelt zu kapseln, wird *Information Hiding* genannt.

Kapselung

```
class HumanBeing(object):
    def __init__(self,na):          # Implementation
        self.__name= na             # Implementation
    def getName(self):              # Interface
        return self.__name
    def changeName(self,newName):   # Interface
        self.__name= newName
```



Vererbung

Die Klasse

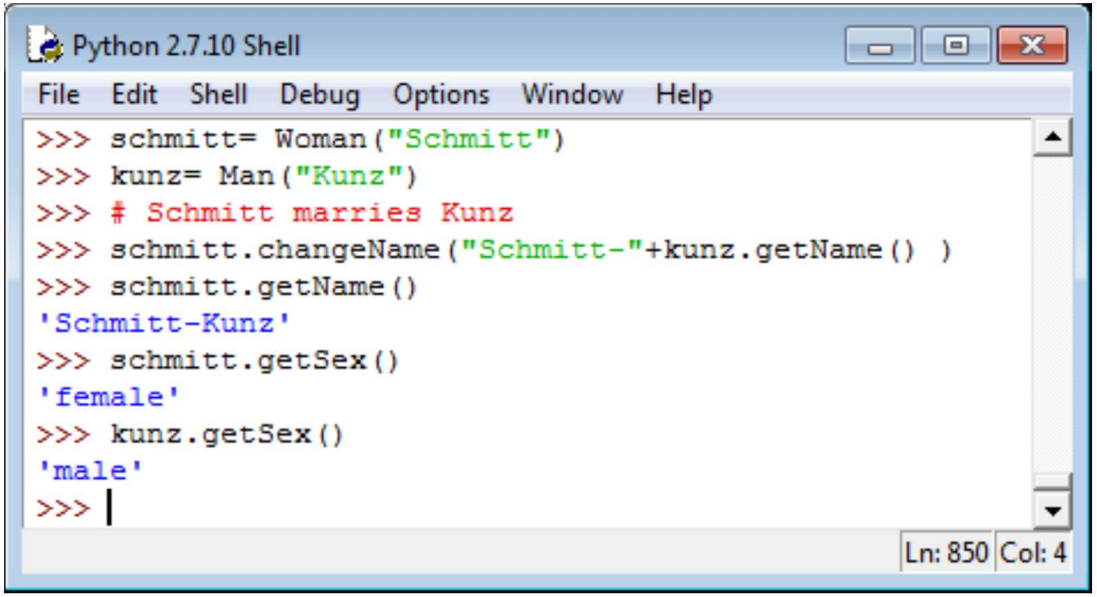
- wird in einer abgeleiteten Klasse spezialisiert.
- die abgeleitete Klasse erbt alle Eigenschaften der Basisklasse.

⇒ Die abgeleitete Klasse ist auch ein Basisklasse. (is-a Relationship)

Die Eigenschaft der abgeleiteten Klasse, die Eigenschaften der Basisklasse automatisch zu erhalten, wird als *code reuse* bezeichnet.

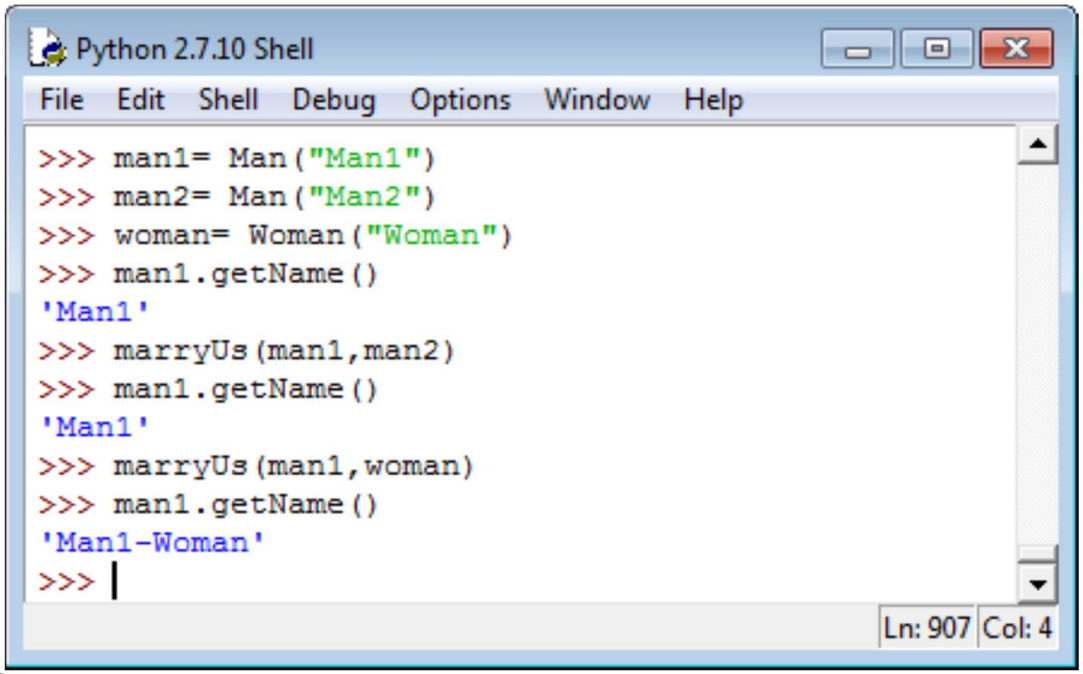
Vererbung

```
class Man(HumanBeing):
    def getSex(self):
        return "male"

class Woman(HumanBeing):
    def getSex(self):
        return "female"
```



Polymorphie

```
def hasDifferentSex(fir,sec):
  return fir.getSex() != sec.getSex()

def marryUs(fir,sec):
  if (hasDifferentSex(fir,sec)):
     fir.changeName(fir.getName()+"-"+sec.getName())
```



Objektorientierte Programmierung

- Seien sie kreativ. Erzeugen sie ein paar Männer und Frauen und interagieren sie mit ihnen in der REPL.

Details

Beispiele:

`HumanBeing.py`, `Student.py`, `MyNumber.py`, `Fraction.py`

Objektmodell

Eine Klasse

- ist ein abstrakter Oberbegriff für Objekte mit einer gemeinsamen Struktur und einem gemeinsamen Verhalten.
- eine Klasse definiert das Aussehen einer Instanz.
- besitzt Attribute in der Form von Variablen und Methoden.
- wird durch das Schlüsselwort `class` definiert.

- Die Klasse `HumanBeing` besitzt die Attribute

```
HumanBeing.__init__
HumanBeing.getName
HumanBeing.changeName
```

Warum ist `self.__name` keine Attribut der Klasse `HumanBeing`?

Objektmodell

`self`

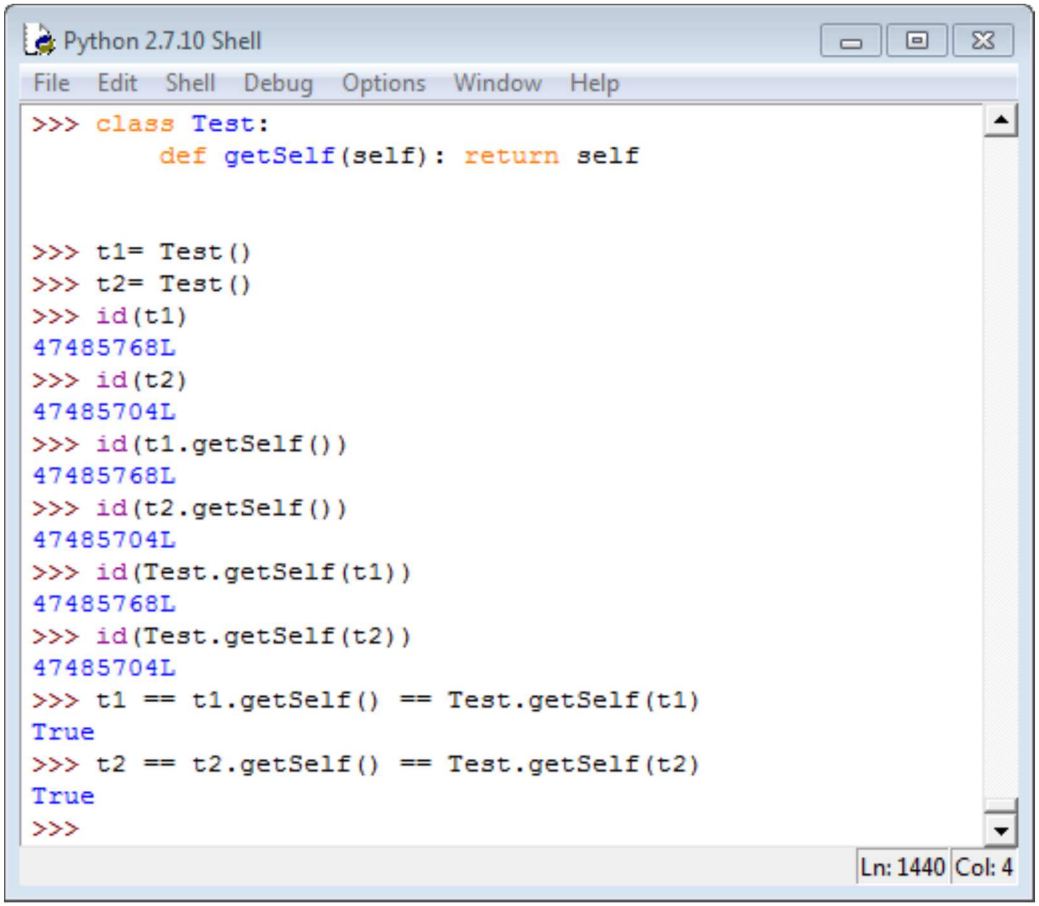
- hilft die Instanzen (Objekte) einer Klasse zu unterscheiden.
- bindet alle Attribute eines Instanz.
- muss für jede Instanz einer Klasse explizit angegeben werden.
- stellt die Identität einer Instanz dar.

Der Aufruf aus der Klasse von

- Variablen: `self.__name`
- Methoden: `self.changeName("newName")` oder

```
HumanBeing(self,"newName)
```

Objektmodell



Die Instanz einer Klasse und `self` sind das gleiche Objekt.

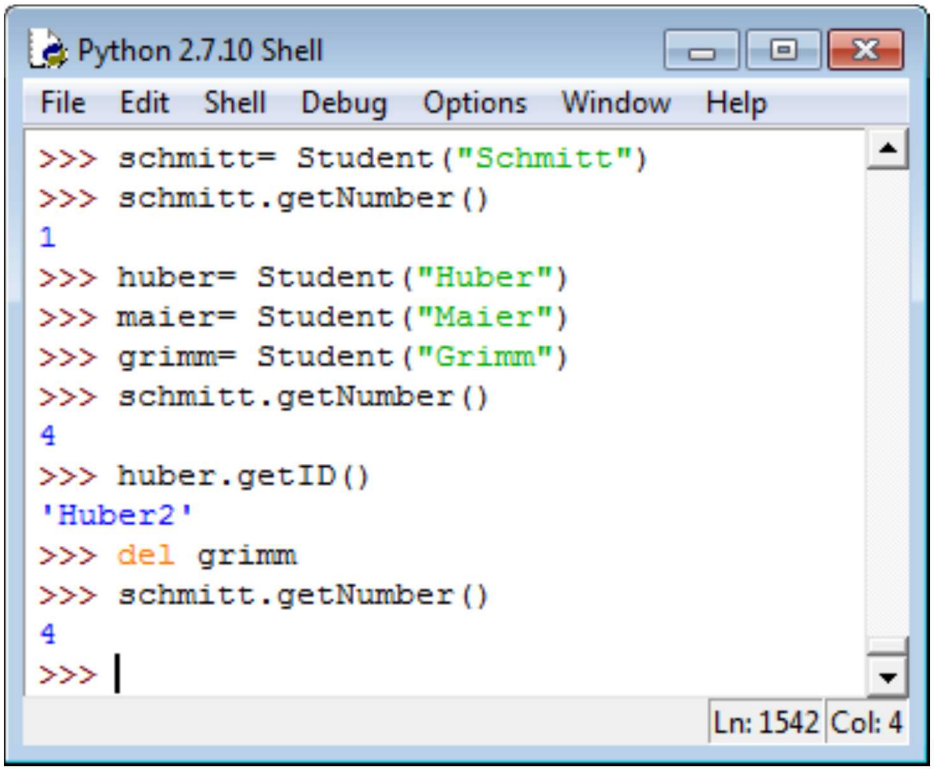
Objektmodell

- **`__init__`**
  - wird bei jeder Instanziierung eines Objekts, sofern sie vorhanden ist, automatisch prozessiert. ➡ *Konstruktor*

- **Instanzvariablen**
  - werden in einer Methode der Klasse definiert.
  - müssen mit `self` angesprochen werden (`self.__name`).
  - existieren für jede Instanz der Klasse.

- **Klassenvariablen**
  - werden im Klassenkörper definiert.
  - müssen mit den Klassennamen angesprochen werden.
  - teilen sich alle Instanzen einer Klasse.

Objektmodell




```
class Student:
  number=0

  def __init__(self,n):
    Student.number += 1
    self.__name= n
    self.setID(n+str(Student.number))

  def getNumber(self):
    return Student.number

  def  setID(self,id):
    self.__ID= id

  def getID(self):
    return self.__ID
```

Objektmodell

Leider habe ich vergessen, die Anzahl der Studenten beim Ableben eines Studenten (`del grimm`) zu dekrementieren.

Implementieren sie die Funktion `__del__` , die der Python-Interpreter automatisch ausführt, wenn ein Objekt seine Gültigkeit verliert.

Konzepte vertieft

Kapselung

- Python kennt keine durch den Python-Interpreter zugesicherte Zugriffskontrolle (public, protected oder private) auf Attribute der Klasse
- Attribute, die mit zwei Unterstrichen beginnen und
  - ohne Unterstriche enden, sind per Konvention privat (*private Attribute*)
  - mit zwei Unterstrichen enden, sind der Python Implementierung vorbehalten
- Private Attribute werden vom Compiler vermangelt

Der Python-Interpreter vermangelt ein privates Attribut, indem es ihm einen Unterstrich und den Klassennamen voranstellt.

Schreiben sie eine Klasse Test, die ein Methode `__test__`, `__test` und `_test` besitzt.

Legen sie eine Instanz t der Klasse Test an und betrachten sie anschließend mit der `dir`-Funktion diese Instanz.

Entsprechen die Namen ihren Erwartungen?