

Zwei Arten von Attributen

- Die Daten, die von einem Objekt gespeichert werden und keine Methoden sind, heißen Attribute. Es gibt zwei Arten:
 - Instanzattribute (= Instanzvariablen):**
Variable, die einer bestimmten Instanz einer Klasse gehört. Jede Instanz kann ihren eigenen Wert für diese Variable haben. Dies ist die gebräuchlichste Art von Attributen.
 - Class attributes (= Klassenvariablen):**
Gehört einer Klasse. Für alle Instanzen dieser Klasse hat dieses Attribut den gleichen Wert. In manchen Programmiersprachen als "static" bezeichnet. Nützlich für Konstanten oder als Counter für die Anzahl der Instanzen, die bereits erstellt wurden.

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 65

Klassenvariablen

- Bisher: Variablen waren Instanzvariablen, d.h., jede Instanz hat eigene "Kopie"
- Klassenvariable** := Variablen, die innerhalb der Klasse genau 1x existieren
 - Alle Instanzen einer Klasse haben eine Referenz auf das gemeinsame Klassenattribut,
 - wenn eine Instanz es verändert, so wird der Wert für alle Instanzen verändert.
- In Python: definiere Klassenvariable außerhalb einer Methode
- Notation zum Zugriff: `self.__class__.name`

```
class sample:
    x = 23
    def increment(self):
        self.__class__.x += 1
```

```
>>> a = sample()
>>> a.increment()
>>> a.__class__.x
24
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 66

Introspektion

- Was tun, wenn Sie den Namen des Attributs oder der Methode einer Klasse nicht kennen, aber trotzdem auf das Element zur Laufzeit zugreifen wollen...
- Lösung: **Introspektion**
 - Methoden, um alle Attribute (Inst.vars und Methoden) einer Klasse aufzulisten
 - Zu einer Zeichenkette, die den Namen des Attributs oder der Methode beinhaltet, eine Referenz zu bekommen (die man verwenden kann)
 - Theoretisch: man könnte sogar zur Laufzeit Methoden hinzufügen
- Im folgenden nur 2 (von vielen!) Möglichkeiten der Introspektion

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 69

getattr(object_instance, string)

```
>>> f = student("Bob Smith", 23)
>>> getattr(f, "full_name")
"Bob Smith"
>>> getattr(f, "get_age")
<method get_age of class studentClass at 010B3C2>
>>> getattr(f, "get_age")() #Das können wir aufrufen.
23
>>> getattr(f, "get_birthday")
# Verursacht AttributeError - No method exists.
```

```
class student(object):
    def __init__(self, name = "", age = 0):
        self.name = name
        self.age = age
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 70

hasattr(object_instance, string)

```
>>> f = student("Bob Smith", 23)
>>> hasattr(f, "full_name")
True
>>> hasattr(f, "get_age")
True
>>> hasattr(f, "get_birthday")
False
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 71

Identitätsfindung

- Ein Objekt hat mehrere "Identitäten"
- Gleichheit mit anderen Objekten: `x == y`
 - Liefert True oder False (kann in der Klasse undefiniert werden!)
 - Überprüft **Inhalt** (= Instanzvariable) auf Gleichheit
- Eindeutige ID: `id(x)`
 - Liefert eine eindeutige Zahl für jedes Objekt zu einem best. Zeitpunkt
- "is a"-Beziehung ("Instanz von")


```
isinstance(x, (ClassName1, ClassName2, ...))
```

 - Liefert True oder False (liefert True auch, falls x Instanz einer Unterklasse)

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 72

Anwendungsbeispiel

- z.B. um verschiedene Aktionen innerhalb einer Methode durchzuführen, abhängig vom Typ des tatsächlichen Parameters

```
class MyClass( object ):
    def __init__( self, other ):
        if isinstance( other, MyClass ):
            ... # make a copy
        elif isinstance( other, (int, float) ):
            inst_var = other # create inst.var
        else:
            inst_var = 0 # default
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 73

Kein Bedarf für Freigaben

- Objekte braucht man nicht zu löschen oder freizugeben
- Python hat eine automatische *Garbage Collection* (Speicherbereinigung)
- Funktioniert mit *Reference Counting* (im 2. Semester mehr)
- Python ermittelt automatisch, wann alle Referenzen auf ein Objekt verschwunden sind und gibt dann diesen Speicherbereich frei
- Funktioniert im Allgemeinen gut, *wenige memory leaks*

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 74

Dokumentation

- Häufige Haltung: Source-Code sei die beste Dokumentation
 - "UTSL" ("use the source, luke")
- Aber:
 - ist viel zu *umfangreich* für einen schnellen Überblick
 - unterstützt das *Navigieren* zu gesuchten Informationen nur wenig
 - gibt keine Auskunft, welche *Annahmen / Voraussetzungen* einzelne Systemteile über das Verhalten anderer Teile machen (Schnittstellen)
 - gibt keine Auskunft, warum gerade diese *Lösung* gewählt wurde (warnt also nicht vor subtilen Fallen)
 - ...
- Deswegen: Korrekte (d.h. auch aktuelle) und hilfreiche *Dokumentation* ist extrem wichtig für die Entwicklung und Wartung eines Programms!

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 75

In Python integrierte Dokumentation

- In Python ist Doku fester Bestandteil der Sprache!
 - Geht noch weiter als in Java

```
def f():
    """
    blub
    bla
    """
    ...
help(f)
```

Ausgabe

Help on function f in module __main__:

f() blub bla

- Diese Kommentare heißen *Docstrings* in Python

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 76

Vollständige Boilerplate für Docstrings

```
"""
Documentation for this module.
More details.
"""

def func():
    """ Documentation for a function.
    More details.
    . . .
    """

class MyClass:
    """ Documentation for a class.
    More details.
    """
    def __init__(self):
        """ Docu for the constructor. """
        . . .
    def method(self):
        """ Documentation for a method. """
        . . .
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 77

Automatische Generierung aus Source

- In allen Sprachen kann die Dokumentation zum großen Teil als Kommentar in den Source eingebettet werden
- Tools (wie z.B. Doxygen www.doxygen.org) erzeugen daraus automatisch sehr ansprechende und effiziente HTML-Seiten (oder LaTeX, oder ...)

 - Einzige Bedingung: Markup der Doku durch sog. *Tags*

- Vorteile:
 - Hoher Automatisierungsgrad (also Arbeitersparnis)
 - durch enge Kopplung an Implementierung leichter aktuell zu halten als separate Dokumente
 - Sowohl interne (Developer-) als auch externe (API-) Doku kann aus demselben Source generiert werden

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 78

Boilerplate für Dokumentation einer Funktion

```
def func( a ):
    """! @brief Einzeilige Beschreibung
    """
    @param param1 Beschreibung von param1
    @param param2 Beschreibung von param2

    @return
    Beschreibung des Return-Wertes.

    Detaillierte Beschreibung ...

    @pre
    Annahmen, die die Funktion macht...
    Dinge, Aufrufer unbedingt beachten muss...

    @todo
    Was noch getan werden muss

    @bug
    Bekannte Bugs dieser Funktion

    @see
    andere Methoden / Klassen
    """
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 79

Dokumentation von Klassen, Moduln, ...

```
"""! @brief Documentation for a module.
    """
    More details.
    """

def func():
    """! @brief Blub """
    . . .

class MyClass:
    """! @brief Class Blub ...
    """

    def __init__(self):
        """! @Brief the constructor """
        . . .

    def method(self, p1):
        """! @Brief Documentation for a method
        @param p1 blubber
        """
        . . .

    ## A class variable.
    classVar = 0;

    ## @var _memVar
    # a member variable
```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 80

Doxygen-Dokumentation als HTML

The screenshot shows a web browser displaying the HTML output of Doxygen documentation for a Python class named 'pyexample.PyClass'. The page includes sections for 'Public Member Functions', 'Static Public Attributes', 'Detailed Description', and 'Member Function Documentation'. The 'Member Function Documentation' section shows a method signature: 'def pyexample.PyClass.pyMethod(self)'. The page is generated by Doxygen 1.8.13.

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 81

Interaktives Beispiel: Rationale Zahlen

- Ziel: Neue Klasse (= Typ) **Rational** mit allen Operatoren

The screenshot shows a code editor with the implementation of a Rational class in Python. The code includes a class definition with methods for addition, subtraction, multiplication, and division, as well as a __str__ method for string representation. The code is annotated with comments and docstrings.

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 82

Unbehandelte Features

- Vererbung, Unterklassen,
- Exceptions
- Lambda-Funktionen
- Compilieren von Code zur Laufzeit durch das Programm selbst
- Embedding
- ...

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Einführung in Python, Teil 2 83