Script language: Python

Classes and objects

Cédric Saule

Universität Bielefeld

12. Februar 2016

Python

–

Classes and objects

Overview

- Again, what is OOP?
- Classes
  - Methods
  - Constructor / Destructor / Attributes
  - Visibility
  - Static attributes and Methods
- Heredity
- Magic Members

Again, what is OOP?

- Data encapsulation
- Methods / Functions
- Visibility
- Heredity (Reusability)

Classes

- Objects are defined through a class.

```
class bar(object):
    pass
```

- `pass` is a `noop` built-in function.
  - Free variable parameter for *Prototyping*.
  - *Ready* Program does not contain `pass` instructions.

Methods

- Methods are similar to functions, so they are defined on the same way.
  - `self` is **always** the first argument.
  - Definition within the blocks `class`.

```
class bar(object):
      def noop(self):
          pass
      def jump(self,from, after):
          pass
```

- Create an object of a given class: `obj = bar()`.
- `obj` is a reference on an object instance.
- Explicit destruction of references: `del(obj)`.

Constructors and Destructors, Attributes

- Constructors are called through objects instanciation.
- Constructors can be defined as methods: `__init__(self)`.
- Any number of arguments(`self` is the first argument).
- The destructor has no argument (except `self`): `__del__(self)` and is called by the GC **before** the object is deleted.
- Attributes are initialized in the constructor.

```
class foo(object):
      def __init__(self):
          self.a = 1
          self.text = "Hello␣World"
```

Heredity

- So far all the methods / attributes are public.
- Do not make always a sense.
- Restriction of visibility by naming.

`name`	public	
`_name`	protected	**W**arning!
`__name`	private	

- Access to private variable $\rightarrow$ Error attributes.

Static methods and attributes

- Static attributes are defined in the classes.
- The are accessed via `CLASSESNAME.ATTRIBUTESNAME`.
- Static methods have **no** `self` argument.
- Definition via `static method`.

```
class Konto(object):
    IntegerNumber = 0

    def showNumber():
        print Konto.IntegerNumber

    showNumber = staticmethod(IntegerNumber)

    ...
```

Exercise

Create a **stack** regardless on the data types / objects available in Python. The **stack** must have the functions **push** and **pop** as member. (Methods use what has been learned in the chapters "modulesör "classes and objects")

Heredity

- Previously: Only encapsulation of data and methods.

Heredity

- Previously: Only encapsulation of data and methods.
- Now: Inheritance.
- Each class must extends the **object**.

```
class bar (object):
        ...
```

- foo inherits from bar and is extended by ...

```
class foo (bar):
        ...
```

Heredity (2)

- The parent class will be called by `__init__`.

```
class foo (bar):
      def __init__(self):
          bar.__init__(self)
          ...
```

  Beware of undefined states!

- *Multiple inheritance* is possible in Python.
- Order from left to right.

Exercise

Write a class **ExtStack** which extends the class to the methods **empty** and **size**.

Magic Member

- Specify properties of a class
- Attributes and Methods
- Always start and end with two `__`.
- A selection:

`__init__(self[,...])`	Constructor.
`__del__(self)`	Destructor.
`__str__`	String Representation.
`__[eq|ne]__(self,other)`	(Non-)equality.
`__[lt|le|gt|ge]__(self,other)`	Comparability.
`__cmp__(self,other)`	Comparability.
`...`	

- $\rightarrow$ Python Documentation