

Script language: Python

Introduction



Cédric Saule

Technische Fakultät
Universität Bielefeld

10. Februar 2016

Python

—

Introduction

Properties, extensions, first steps.

Python – Versions overview.

- Dynamic programming language, unfortunately, it is most of the time known as a scripting language.
- First version: 1991.

Current version :

3.4.2 Current *stable* Release

- Many performances and improvements.
- Syntax partially modified.
- No Backward compatibility before 2.X.

- ### 2.7.5
- 2.X are the Python versions most widely used.
 - From 2.6 late implementations to 3.X functionalities.
 - Many modules, which are not compatible with the 3.X functionalities.
 - Backward compatibility due to the autoconverter → 2to3.

Python – Properties.

- Optimized for readability.
- Few keywords.
- Multi OS-Support.
- Multi-paradigm: imperative, object / aspect oriented and functional programming.
- Excellent help systems are integrated.

Python – Internals.

- Strongly object-oriented → Everything is an object!
 - No literals (no magic numbers).
 - Strongly typed.
 - Functions are objects: Can be overloaded during execution → *Dependency Injection*.
- Blocks are characterized by *TAB* indents.
- Duck typing : Variables are typed, not the underlying objects!

Python – Features and enhancements.

Python-Shell Interactive Python mode.

IDLE Built-in development environment.

iPython Powerful alternative interactive shell with lots of magic™.

Cython We can extract parts of the code in Python and to run them optimally on the machine.

The first script.

Text editor vi(m), emacs, gedit, kate → NO word processor.

File extension *.py

Preamble *#!/usr/bin/env python*

Rights *chmod u+x <File name>*

```
#!/usr/bin/env python  
print("My first output.")
```

```
$ chmod u+x test.py
```

```
$ ./test.py
```

iPython – An alternative Python Shell.

- Tab-Completion.
- Integrated help on ? and ??.
- OS Shell: !<COMMAND>
- Integrated debugger.
- Command history.
- Tutorial: %magic

THE tool of choice for interactive experiments!

Start \$ ipython

End > quit or CTRL^d

Exercise – First steps

Check the following expressions in the interactive shell (iPython) and try to explain what is happening here and why:

```
> 1/7
> 1.0/7.0
> print 1./7

> 0.9999999999999999
> print 0.9999999999999999

> 0.9999999999999999*9/9
> 0.9999999999999999/9*9
```

Python

—

Introduction

Object references, namespaces, Callables, GC

Objects, references and instances.

Python is strongly object-oriented.

- „primitive“ data structures like `int` are objects.
- Functions are objects.
- In fact, everything in Python is an object.

Objects, references and instances.

```
> 23      # Instance of type int
```

```
> a = 23   # a is a reference on an instance  
           # of type int with the value 23.
```

```
> b = a    # b references the same instance than a.
```

```
> b = 46
```

Which value contains a and b ?

Objects, references and instances.

Basic functions you need to know:

- `dir()` List all objects properties (namespaces, Callables).
- `type()` Return the type of the given object.
- `help()` Call the help.

They are also called `builtin_function_or_method`

Exercises – Who I am and what can I do?

Consider the following expression:

```
> text = "Hello_world!"
```

Apply the commands `type()`, `dir()`, and `help()` on the variables `text`. What happens here?

Look at using `help()` the help for functions/type.

Namespaces and Callable

Create a complex number $c=1+2j$. Print the real and imaginary parts of this number by using `real` and `imag` and finally compute the conjugate of this complex number.

```
> imag(c)
```

```
> c.imag
```

```
> c.real
```

```
> c.conjugate
```

```
> c.conjugate()
```

Namespaces and Callable

Create a complex number $c=1+2j$. Print the real and imaginary parts of this number by using `real` and `imag` and finally compute the conjugate of this complex number.

```
> imag(c)           # ->NameError
> c.imag            # Namespace

> c.real            # Namespace

> c.conjugate       # The object's function
-> Callable
> c.conjugate()     # Call
```

Namespaces can be compared with attributes (Java) and Callable with function ($\rightarrow$ We have to `__call__()` implements the functions.)

Identity comparison vs. content comparison

```
> a = "Hello World!"
> b = a
> c = "Hello World!"

> a == b
True
> a == c
True
> a is b      # id(a) == id(b)
True
> a is c      # id(a) == id(c)
False
```

- Identity of an object: `id()`
- Operator of identity comparison : `is`
- Operator of content comparison: `==`

Garbage Collector

The same as *Java*, Python has a *Garbage Collector*, which cleans unneeded instances. An instance can be deleted from the PC when there is no anymore reference to it.

```
> a = 23
> a = 46    # no reference to instance
             # (int :: 23)
             # -> can be removed by the GC
             and free memory.

> del(a)    # del() removed references explicitly
```