Python
Language introduction

Gerald Senarclens de Grancy
santa@sbox.tugraz.at
http://grancy.at.tf

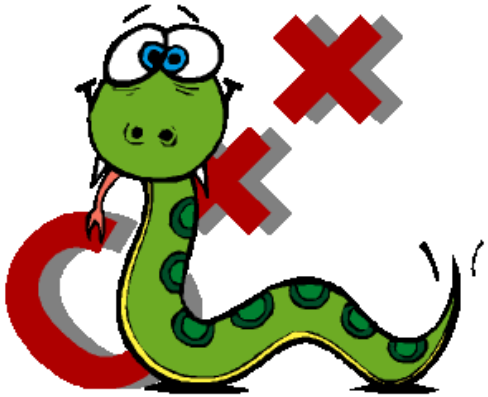

Table of contents

- Open Source
- object oriented scripting/ programming language
- completely dynamic language (type declaration not necessary)
- platform independent (Windows, Unix, Linux, Mac, Amiga, BeOS, Win CE, Dos, QNX, Psion Series 5, OpenVMS, VxWorks and other environments providing a Java virtual machine)
- provides important high level data types
  - strings, tuples, listes, dictionaires and files
- simple and elegant syntax (very easy to learn)

- developed by Guido van Rossum in 1990
  - beginnings at "Centrum voor Wiskunde en Informatica"
  - maintained at the moment at Corporation for National Research Initiatives (CNRI)
- Python Software Foundation (PSF)
  - responsible for further language improvements
- Python Business Forum (PBF)
- the community, SIGs (special interest groupes)

- Standard Python Software
  - Python Debugger

```
>>> import pdb
>>> pdb.run('function(arguments)')
```

- GNU Emacs Python mode
- Python Shell (to "play" around, testing statements and gain expirience)
- IDLE
- Boa Constructor, wxGlade
- hundrets of libraries in the standard distribution (and even more freely available on the internet)

- `$ python program.py`
- as shell script (not possible when using Windows)
  - `#!/usr/bin/env python`
  - `$ chmod u+x program.py`
- `import program`
- using an integrated development environment

- truth tests
  - all empty values and everything equal 0 is considered being "false"

    `[], {}, (), 0, 0.0, None`
- if

```
if test:
   suite
[elif test:
   suite]
[else:
   suite]
```

- while

```
while test:
   suite
[else:
   suite]
```

- for

```
for target in sequence:
   suite
[else:
   suite]
```

- **`pass`** (empty statement, like ";" in Java)
- **`break`** (continues execution after the innermost loop)
- **`continue`** (jumps to the next iteration of the innermost loop)
- **`return`** (leaves the current function returning a value or **`Null`**)
- no **`switch case`**
  - can be done by indexing dictionaries

- Strings (immutable)
  “`Python's`”, `r'Python`”`s raw'`, `u`“““`multiline unicode`”””
  - strings are concatenated automatically
  - special letters: `%` `\n`, `\t`, `\\`,...
  - basic methods on string s

```
s.strip(), s.lstrip()
s.find(sub [, start [, end]])
s.rfind(sub [, start [, end]])
s.replace(old, new [, maxtimes]
s.capitalize(), s.swapcase(), s.islower()
...
```

- Tuples (immutable)
  `(), (3), (1, 6, 2), ('a', 4, 'string', 34.876, (3, 1))`
- Lists (mutable)
  `[], [3], [1, 6, 2], ['a', 4, 'string', [[1], (3, 1)], (1, [2])]`
  - basic methonds on list l

    ```
    l.append(element), l.insert(i, element)
    l.remove(element)
    l.index(element), l.sort([function])
    l.count()
    l.pop([i])
    ...
    ```

- sequence operations
  - concatenation `+`
  - repetition `*`
  - indexing `[i]` (`l[0]`, `s[4]`)
  - slicing `séq[start:end]` (`t[1:3]`, `l[:]`, `s[:-3]`)
  - some more

```
elem in seq, elem not in S
for elem in seq
len(seq)
min(seq), max(seq)
```

- Dictionaries (mutable)
  `{}, {'spam': 1, 'egg': 5}, {'d': 'Ed':1, 'l':[1, 2] }`
  - basic methods on dictionary d

```
d.keys(), d.values(), d.has_key(key)
d.copy(), d.clear()
d.update(dictionary)
d.get(key, [default]), d.setdefault(key, [default])
d.popitem()
...
```

- Files
- read

```
file = open('filename', 'r')
file.read(), file.read(n)
file.readline(), file.readlines(), file.xreadlines()
```

- write

```
file = open('filename', 'w')
file.write(string), file.writelines(stringlist)
```

- general file methods

```
file.closed, file.mode, file.name
file.close(), file.tell()...
```

- Functions

```
def myFunction(arg1, arg2):
    return arg1, arg2
def myFunction2(arg='default value'):
    print arg
def myFunction3(*args):
    for arg in args:
        print arg
def myfunction4(**args):
        keys = args.keys()
        for key in keys:
            print args[key]
```

- object oriented features
  - classes are "object factories"

```
class MyClass[(superclass)]:
   static = 'salut'
   def __init__(self, arg):
      self.property = arg
   def __repr__(self):
      reprStr = 'property = ' + str(self.property) + '\n'
      reprStr += 'static = ' + str(self.static)
      return reprString
   def setProperty(self, newValue):
      self.property = newValue
   def getProperty(self):
      return self.property
myObject = MyClass(arg)
```

- Exceptions

```
myError = 'a very simple exception'
def problemFunction():
   raise myError, 'additional description of the exception'
try:
   problemFunction()
except myError, data:
   print 'Exception caught:', data
else:
   print 'no exception occured'

try:
   raise 'new Exception'
finally:
   print 'doing cleanup'
```

- for operating system concerns use the **`os`** module
  - concerning platform independence use eg. **`os.sep`**, **`os.linesep`**
  - the following command gives access to a system shell **`os.system('shell command')`**
- **`sys`**
  - everything about the Python system itself (eg. **`sys.argv`**)
- **`pickle`** (allows writing objects to files very easily)
  `pickle.dump(myObject, file)`, **`object = pickle.load(file)`**
- some others: **`string`**, **`re`** (regular expressions)

- small selection of the available GUI libraries
  - Tkinter
  - wxPython
  - Java Swing (using Jython one has access to the complete Java API and all Java libraries)
- real life examples
  - BitTorrent
  - Boa-Constructor
  - Nicotine

- http://www.python.org
  - official Python documentation
- the Python Tutorial
- Python newsgroups
  news://your.news.server/comp.lang.python
  news://your.news.server/comp.lang.python.announce
- recommandable books
  - Learning Python, 2nd Edition
  - Python Pocket Reference 2nd Edition

You can obtain the document's LaTeX sources at http://www.sbox.tugraz.at/home/s/santa/ (section Lectures), where you can also find a French version of this document. All examples I used during my presentations are freely available at the former mentioned webside.