Le langage Python

http://www.python.org

Python est un langage portable, dynamique, extensible, gratuit.
Approche modulaire et orientée objet de la programmation.
Python est développé depuis 1989 par Guido van Rossum.

Caractéristiques du langage

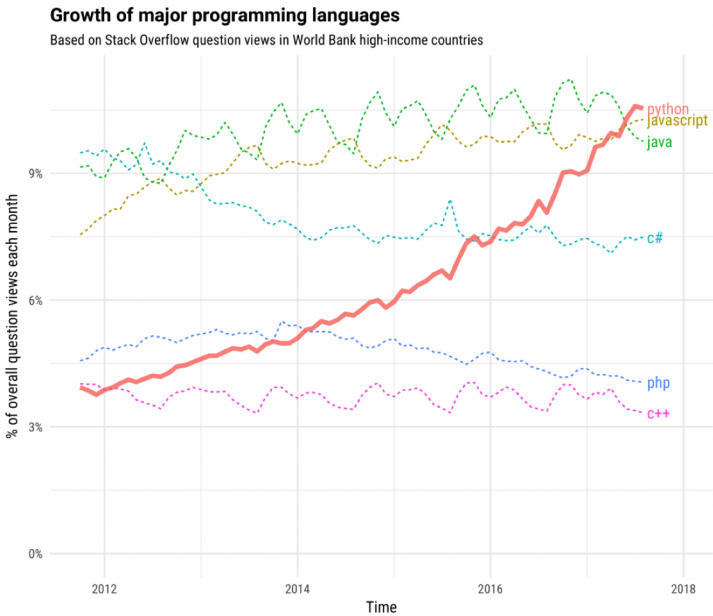
- ✓ **Libre** (GPL)
- ✓ **Portable** (L!nux, Window$, M@c, …)
- ✓ **Syntaxe** simple (basée sur les tabulations) avec des types de données évolués
- ✓ Mécanisme d'**extension**
  - o C, C++, Fortran
- ✓ **Interprété** avec typage dynamique
- ✓ **Orienté objet** : tout est objet
- ✓ Gestion automatique de la **mémoire** (comptage de référence)
- ✓ **Bibliothèque** standard
  - o Calcul matriciel,
  - o Interface graphique
  - o visualisation 2D et 3D,…

Python – Popularité

- Popularité grandissante (#1 des questions sur stackoverflow)
- Devient une référence en web, data science, éducation …
- Utilisé par les grands groupes

"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language" (Peter Norvig, director of search quality at Google, Inc.).

- En infographie et traitement de l'image
  - Langage de script de nombreux modeler/rendering engine (Blender, Ogre3D, Maya, 3ds Max)
  - API de référence pour deep learning (Keras, TensorFlow, Theano, etc.)



Python – Basic Types

Integer & Long

```
>>> 2 + 3
5
>>> type(5)
<type int>
>>> type(3333333333333333)
<type long>
```

Float

```
3.14 10. .001 1e100 3.14e-10
```

Arithmetic Operators

```
+,-,*,/
>>> 5 + 3
8
# # integer vs float division
>>> 20 / 3    >>> 20 / 3.
6             6.6666
>>>
```

Boolean

```
True, False
```

Boolean Operators

```
x and y ; x or y ; not x
is, is not
<, <=, >, >=, ==, !=, <>
x < y <= z ⇔ x < y and y <= z
```

String

```
string= "..." or '...'
'spam eggs`,"spam eggs"
```

Standard operation

```
>>> name[0:2] # slicing
# Concatenation
>>> s=`little'+' '+`fish`
>>> len(s) # length
```

Python – Advanced Types

Tuple (immutable)

```
tuple= (x,y,z)
>>> a = ('spam', 100, 1234)
```

Standard functions

```
>>> len(a)
>>> a[-2] # a[len(a)-2]
>>> a[0:1] # slicing
>>> a[:2] # a[0:2]
>>> a[2:] # a[2:len(a)]
```

List (mutable)

```
list= [x,y,z]
>>> a = ['spam', 100, 1234]
```

Standard functions

```
>>> a[2]= 23
>>> del a[2] # remove a[2]
```

Dict

```
dict= { key:value,...,
key:value }
>>> tree= {'a':['b','c'],
           'b':['d','e'] }
```

Standard functions

```
store value:     d[key]=value
extract value:   d[key]
delete pair:     del d[key]

dict.keys() -> [key]
dict.values() -> [value]
dict.items() -> [(key,value)]

dict.has_key(key) -> bool
key in dict
```

Python – Control Flow

If

```
>>>if x == 5:
...    print "value < 5"
...elif 10 < x < 20:
...    print "value in ]10,20[«
...else:
...    print "more than 20"
```

LOOP: while, for

```
# Fibonacci
>>> a, b = 0, 1
>>> while b < 10:
...    a, b = b, a+b

>>> a = ['cat', 'window']
>>> for x in a:
...     print x, len(x),
cat 3 window 6
```

Range

```
range(i,j,k) ⇔ Interval(0,i)
>>> range(0,8,3)
[0,3,6]
```

List Comprehension

```
[f(x) for x in seq] ⇔ Foreach
[f(x) for x in seq if cond(x)]

>>> vec = [2, 4, 6]
>>>[3*x for x in vec if x > 3]
[12,18]
```

Python – Function, Classes, etc

Function

```
# function definition
def function (arg1,..., argn):
    """docstring"""
    <block>
```

Return

```
# function: return
>>> def sum(a,b):
...     return a+b
# procedure: no return (None)
>>> def sum(a):print a+b
```

Default Arguments

```
>>> def sum(a= 1,b= 2):
...     return a+b
>>> sum(b=3)
4
```

Class definition

```
class ClassName:
   <statement-1>
   def __init__(self, arg):
   . . .
   <statement-N>
```

Example

```
# Ctor with arguments
class Complex:
  def __init__(self, r, i):
    self.r= r
    self.i= i
  def __str__(self):
    s="(%d,%d)"%(self.r,self.i)
    return s
```

Python – Modules

Définition

File containing Python code

Chargement

import

```
>>> import os
>>> os.name
'nt'
```

from module import *

```
>>> from os import *
>>> name
'nt'
```

Accés au nom

module.__name__: nom du module

```
>>> if __name__ = "__main__":
...    run_test()
```

Permet d'ajouter des tests

Numeric python

```
>>> import numpy as np
>>> a = np.array([[1,2,3],
[4,5,6]])
>>> a.shape
(2,3)
>>> a[:,1] # all y coord
[2,5]
```

Linear algebra

```
>>> from numpy.linalg import *
>>> norm(a[1])
8.774964387392123
```

2D plot

```
>>> from matplotlib.pyplot
import *
>>> plot(x,y)
>>> show()
```

Development environment

- Console (ipython, python)

fboudon — IPython: Users/fboudon — ipython — 80×24

```
Last login: Thu Sep 13 11:21:24 on ttys001
papaya:~ fboudon$ ipython
Python 2.7.15 (default, May  2 2018, 00:53:27)
Type "copyright", "credits" or "license" for more information.

IPython 5.4.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: %run myscript.py
```

- Editeur (Sublime text, PyCharm)

compare_zbufferrendering.py — UNREGISTERED

zbufferengine.cpp | compare_zbufferrendering.py

```
48
49  class PglViewer (QGLViewer):
50      def __init__(self, parent = None, label = None):
51          QGLViewer.__init__(self,  parent)
52          self.scene = None
53          self.discretizer = Tesselator()
54          self.glrenderer = GLRenderer(self.discretizer)
55          self.bboxcomputer = BBoxComputer(self.discretizer)
56          self.animationMode = eStatic
57          self.forceclear = True
58          self.camera().setViewDirection(Vec(-1,0,0))
59          self.camera().setUpVector(Vec(0,0,1))
60          self.timer = QTimer()
61          self.timer.setSingleShot(True)
62          self.timer.timeout.connect(self.drawCPU)
63          self.label = label
64          self.timeout  = 1000
```