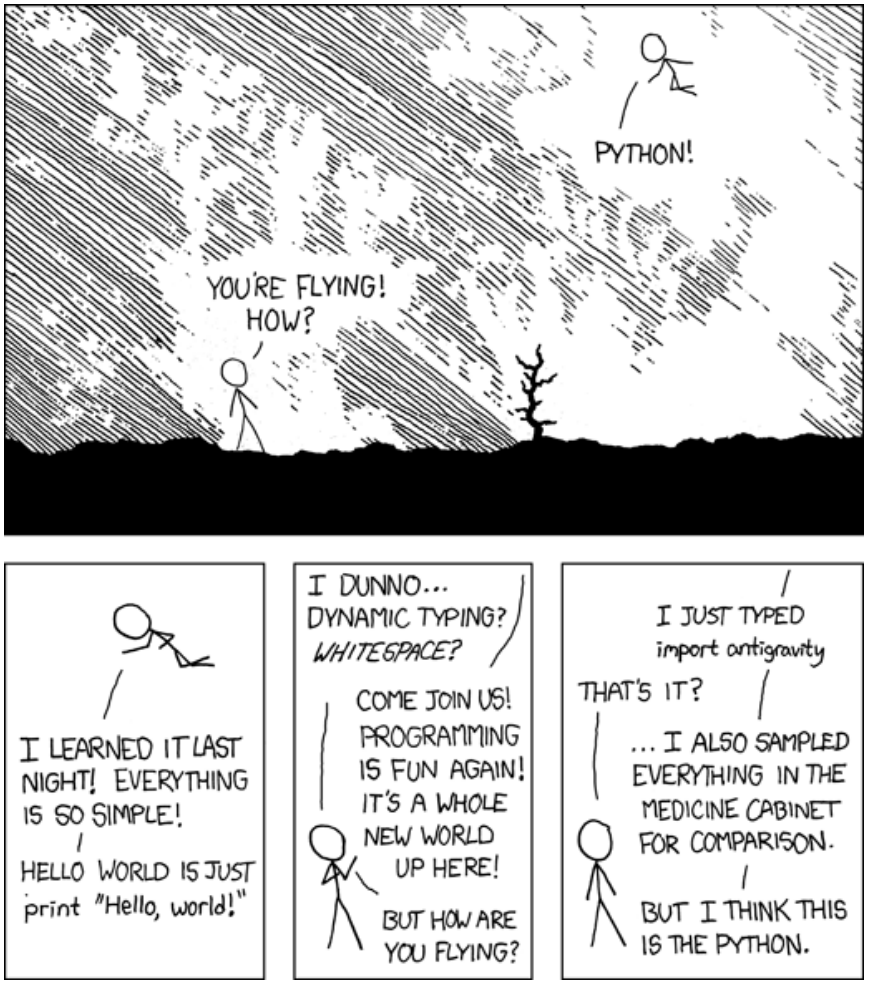
Introduction au langage Python

Pierre Bourdon Nicolas Hureau

LSE

29 septembre 2011

Pourquoi ?


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Plus sérieusement

Python est :

- Simple
- Suffisamment rapide
- Mature
- Utilisé
- Pratique

Plan

Bonjour, monde !

```
# Ce programme affiche "Bonjour, monde !"
print "Bonjour, monde !"
```

Opérations de base

```
x = 21              # Variable
print x             # Affiche 21
print x + x         # Affiche 42
print x * 2         # Affiche 42
print x / 4         # Affiche 5
print x % 4         # Affiche 1
```

Tables de multiplication

```
n = 7
for i in range(10):
    print n, "x", i, "=", n * i

# 0 x 7 = 0
# 1 x 7 = 1
# ...
# 9 x 7 = 63
```

Chaines de caractères

```
chaine = "Bonjour, monde !"
print chaine.upper() # BONJOUR, MONDE !
print chaine.lower() # bonjour, monde !
print chaine.endswith("monde !") # True
print chaine.startswith("Bonjour") # True
```

Listes

```
liste = [3, 7, 14, 15, 22, 41]
print liste[0] # 3
print liste[1] # 7
print liste[-1] # 41
print liste[-2] # 22

for i in liste:
    print i + 1
```

Dictionnaires

```
vrais_noms = { "delroth": "Pierre Bourdon",
               "kalenz": "Nicolas Hureau" }

print vrais_noms["delroth"]
print vrais_noms["kalenz"]

print "kalenz" in vrais_noms # True
print "chiche" in vrais_noms # False
```

Python est :

- Interprété
- Dynamique
- Multi-paradigmes
- Pragmatique

Zen of Python

- Explicit is better than implicit
- Readability counts
- Special cases aren't special enough to break the rules.
- There should be one– and preferably only one –obvious way to do it.
- Namespaces are one honking great idea – let's do more of those !

Lancer un shell interactif

```
$ python
Python 2.7.2 (default, Jun 29 2011, 11:10:00)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license"
for more information.
>>> 1 + 1
2
>>>
```

Lancer un script

```
$ cat script.py
print "Hello, world!"
$ python2 script.py
Hello, world!
```

Pas de compilation !

Python 101 - Conditions, boucles

```
if True:
    # Ceci est un commentaire
elif False:
    # Les blocs sont marqués
else:
    # par leur indentation

mon_nom = "Nicolas" if True else "Pierre"

while 4 == 2:
    print "ZoBuBuGa"

for shadock in planete:
    print shadock, "pompait"
```

Python 101 - Fonctions, opérateurs

```
def is_leap_year(year):
    return (year % 4 == 0 and
            year % 100 != 0) or
            year % 400 == 0
is_leap_year(2000) # True

not True           # False
3 > 2 > 1          # True
"a" in "Test.a"    # True
15 ** 2            # 225
"a" * 5            # 'aaaaa'
```

Exceptions

```
raise Exception

try:
    file = open("fichierinexistant")
    content = file.read()
except IOError, err_msg:
    print err_msg
```

Structures de données

On a déjà vu les listes et les dictionnaires

```
# Tuple
tup = 1, 2          # (1, 2)
a, b = tup          # a = 1, b = 2
a, b = b, a         # swap !

# Set
set(["gl", "pm", "pb", "nh"])
# set('pb', 'nh', 'gl', 'pm')
foobar_set = set("foobar")
# set('f', 'o', 'o', 'b', 'a', 'r')
"a" in foobar_set # True
```

Classes

```
class bde:
    def __init__(self, nom):
        self.nom = nom
        self._nbr_vote = 0
    def __str__(self):
        return "Votai %s" % self.nom
    def ajouter_vote(self):
        self._nbr_vote += 1
    def nbr_vote(self):
        return self._nbr_vote
test = bde("Test.")
print test                  # Votai Test.
test.ajouter_vote()
print test.nbr_vote()   # 1
```

Principle of Least Astonishment

There should be one – and preferably only one – obvious way to do it.
Prenons exemple sur les autres langages : s.length(), s.getLength(), s.size() ? Ou même s.length, s.size, s.Count ?
Python :

```
len(s)
```

Principle of Least Astonishment

Obtenir la taille d'une donnée devrait toujours se faire de la même façon

```
class ma_bite:
    ...
    def __len__(self):
        return sys.maxint
    ...

len(ma_bite) # sys.maxint
```

Modularité

Il est évidemment possible de coder dans plusieurs fichiers grâce au mecanisme de modules :

```
super_programme.py
mon_module.py
mon_package
|-> __init__.py
|-> sous_module.py
```

Modularité

```
import mon_module
from mon_package import sous_module
```

with

Vous souvenez-vous de comment lire le contenu d'un fichier ?

with

```
file = open("fichierinexistant")
content = file.read()
file.close()
```

with

```
with open("test") as file:
    content = file.read()
```

Le fichier sera fermé automatiquement lors de la sortie du bloc

Listes en compréhension

- En maths, $\{x \times 2 | x \in [|1;5|]\}$
- En python,

```
[x * 2 for x in range(1, 6)]
```

-
```
[x * 2 for x in range(1, 6) if x % 3 == 0]
```

Slices

```
l = [1, 2, 3, 5, 7, 11]
print l[:4] # [1, 2, 3, 5]
print l[4:] # [7, 11]
print l[2:4] # [3, 5]
print l[:-2] # [1, 2, 3, 5]
```

Ensembles

- `a = set([1, 2, 3, 4])`
- `b = set([2, 3, 5, 7])`
- `print a & b # set([2, 3])`
- `print a | b # set([1, 2, 3, 4, 5, 7}`
- `print a ^ b # set([1, 4, 5, 7])`

Générateurs

```
def fibonacci_up_to(n):
    a = b = 1
    while a <= n:
        yield a
        a, b = b, a + b

for f in fibonacci_up_to(10):
    print f # 1, 1, 2, 3, 5, 8
```

Décorateurs

```
@cache
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)
```

Flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/hello")
def hello():
    return "<h1>Hello, world!</h1>"

app.run(host="127.0.0.1", port=8080)
```

Pygments

```
from pygments import highlight
from pygments.lexers import CLexer
from pygments.formatters import HtmlFormatter

code = "int main(void) { return 0; }"
print highlight(code, CLexer(), HtmlFormatter())
```

Twisted

- Bibliothèque réseau géniale
- HTTP, POP3, SMTP, SSH, FTP, . . .
- Asynchrone

NumPy

Exemple

$$\begin{aligned} 4x + 3y + 2z &= 2 \\ 3x + 6y - 8z &= 5 \\ 7x + 2y - 4z &= -1 \end{aligned}$$

```
from numpy import matrix
from numpy.linalg import solve

print solve(matrix("4 3 2; 3 6 -8; 7 2 -4"),
            matrix("2; 5; -1"))
# [-0.65517241, 1.4137931, 0.18965517]
```

PIL

```
from PIL import Image
im = Image.open("test.jpg")
im = im.rotate(45)
im = im.thumbnail((800, 600), Image.ANTIALIAS)
im.save("out.png")
```

ctypes

```
import ctypes

my_lib = ctypes.cdll.LoadLibrary("libmachin.so")
my_lib.fonction(1, "a", 2)
```

ctypes + unittest

```
def test_fgetc_basic(self):
    """fgetc correctly read one character"""
    open("test-file", "w").write("abc")
    fp = my.fopen("test-file", "r")
    self.assertEqual(my.fgetc(fp), ord("a"))
    self.assertEqual(my.fgetc(fp), ord("b"))
    self.assertEqual(my.fgetc(fp), ord("c"))
    self.assertEqual(my.fgetc(fp), my.LBS_EOF)
    self.assertEqual(my.fclose(fp), 0)
    os.remove("test-file")
```

Et plein d'autres !

- Plusieurs milliers de bibliothèques sur PyPI
- `easy_install <libname>`
- Moteur 3D, bibliothèque réseau, framework web, ...

Pour aller plus loin

- `http://learnpythonthehardway.org/`
- `http://docs.python.org/`
- `http://diveintopython.org/`

Questions ?

- Slides disponibles sur `http://lse.epita.fr/` prochainement
- `#epita @ irc.rezosup.org`
- `delroth@lse.epita.fr - kalenz@lse.epita.fr`