Python

Présentation du langage de programmation Python

SED Rocquencourt
David Froger

5 juillet 2013

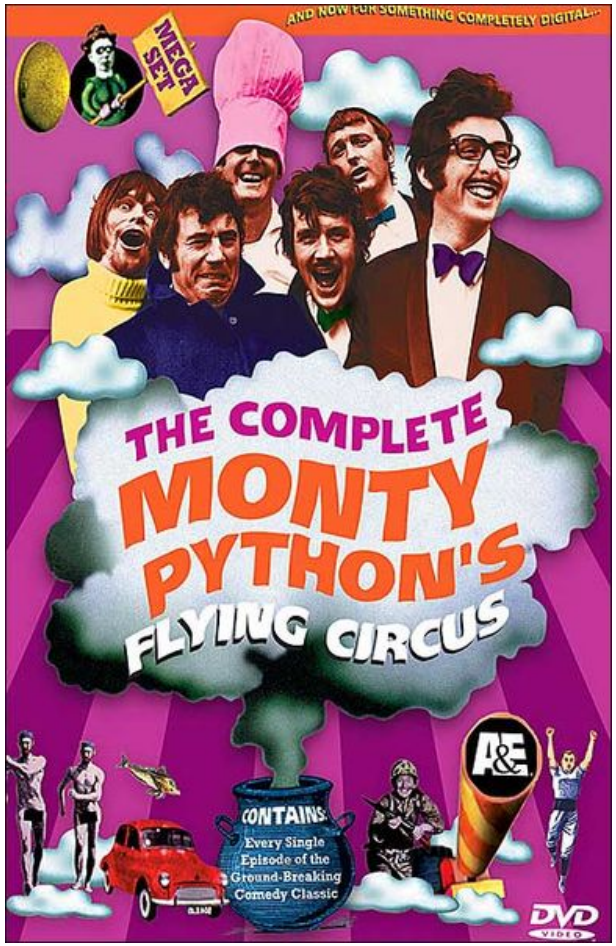
INTRODUCTION

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et d'une approche de la programmation orientée objet simple mais efficace. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur de nombreuses plateformes.

SOMMAIRE

1
Qu'est-ce que Python ?

D'où vient le nom ?



- Vient de la série humoristique télévisée britannique *Monty Python's Flying Circus*
- Références fréquentes aux *Monty Python*

Qu'est-ce que Python ?

- Langage de prototypage rapide orienté objet
- Pas seulement un langage de script
- Pas seulement « un autre *Perl* »
- Facile à apprendre, lire, utiliser
- Extensible
  - *C*, *C++*, *Fortran*
  - *Java*, (grâce à *Jython*)
- Embarquable dans d'autres applications

Licence

- Python Software Foundation License (PSFL) : licence libre semblable à la licence BSD
- Compatible GPL (Python 2.0.1, 2.1.1, et nouvelles versions)

Maturité

- Langage mature (23 ans)
- Supporté par une large communauté
- Nombreuses ressources (livres, vidéos, sites internets, ...)
- Concepts simples, faciles à apprendre
  - se lit comme du *pseudo-code*
  - approprié comme premier langage
  - approprié comme dernier langage

Propriétés de haut niveau

- Extrêmement portable (GNU/Linux, Mac OS X, Windows, et bien d'autres)
- Interprété (compilation implicite et automatique en *bytecode*)
- Gestion automatique de la mémoire
  - Comptage de références
  - *Garbage collector* pour la détection de cycles
- Fiable : pas d'erreur de mémoire causées par vos bogues

Où est-il utilisé ?

- Prototypage rapide
- Programmation Web (coté client et serveur)
- Scripts
- Applications scientifiques
- Comme langage d'extension
- Traitement XML
- Base de données
- Application GUI
- Enseignement

Propriété du langage

- Tout est objet
- Fonctions, classes, modules, packages
- Gestion des exceptions
- Typage dynamique, polymorphisme
- Surcharge des opérateurs
- Indentation pour la structure des blocs

Types de données de haut niveau

- Nombres : entiers, flottants, complexes
- Chaînes de caractères (immuables)
- Conteneurs : listes et dictionnaires
- Autres types (données binaires, expressions régulières, ...)
- Modules d'extensions peuvent définir de nouveau types 'built-in'

Bibliothèque standard riche

- Chaînes de caractères, dates/heures, mathématiques
- Persistances de données, bases de données, compression, cryptographie
- Parseur de ligne de commande, loggeur
- Thread, processus
- Internet : email, json, cgi, url, smtp, HTML, XML
- Interface graphique : TK
- Énormément de bibliothèques externes

Comparé à Perl

- Plus facile à apprendre (important pour les utilisateurs occasionnels)
- Code plus lisible
- Code plus facile à maintenir
- Moins de « magie »
- Plus de sûreté
- Meilleur intégration de *Java*

Comparé à Java

- Code jusqu'à 5 fois plus cours, et plus lisible
- Typage dynamique
- Héritage multiple, surcharge d'opérateurs
- Développement plus rapide
  - Pas de phase de compilation
  - Moins de saisie de code
- Peu s'exécuter plus lentement, mais :
  - Développement beaucoup plus rapide
  - Python utilise moins de mémoire

De même pour *C/C++* (plus marqué).

Exemple de fonction

```
def gcd(a, b):
    " greatest common divisor "
    while a != 0:
      a,b = b%a, a
    return b
```

Exemple de classe

```
class Stack:
  "A well-known data structure "
  def __init__(self):
    self.items = []

  def push(self,x):
    self.items.append(x)

  def pop(self):
    x = self.items[-1]
    del self.items[-1]
    return x

  def empty(self):
    return len(self.items) == 0
```

2 Tutoriel Python

http ://docs.python.org/2.6/tutorial/

3 Bibliothèques externes

Extension Python

```c
#include <Python.h>

static PyObject *
spam_system(PyObject *self, PyObject *args) {
    const char *command;
    int sts;

    if (!PyArg_ParseTuple(args, "s", &command))
        return NULL;
    sts = system(command);

    return Py_BuildValue("i",sts);
}
```

Extension Python

```
static PyMethodDef SpamMethods[] = {
    {"system", spam_system, METH_VARARGS,
        "Execute a shell command."},
    {NULL, NULL, 0, NULL}
};

PyMODINIT_FUNC
initspam(void) {
    (void) Py_InitModule("spam", SpamMethods);
}
```

Extension Python

```
g++ -c -fPIC -I /usr/include/python2.6 spammodule.c
g++ -shared -o spam.so spammodule.o
```

```
>> import spam
>> status = spam.system('ls')
```

Package scientifiques

- NumPy : tableau de N dimensions.
- SciPy : algorithmes numériques (intégration, optimisation, . . .)
- Matplotlib : tracé 2D de haute qualité
- ipython : interpréteur Python amélioré
- h5py : interface à HDF5

Interfaces graphiques

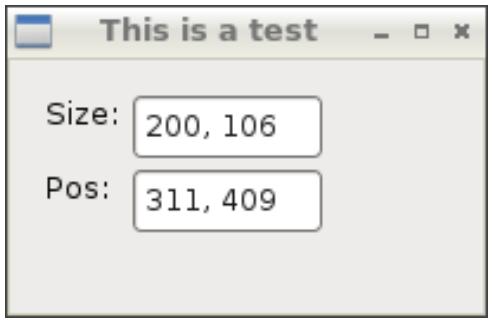
- TKinter
- PyGTK
- PyQT
- wxPython

Exemple wxPython

```
import wx

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, "This is a test")
        frame.Show(True)
        self.SetTopWindow(frame)
        return True


app = MyApp(0)
app.MainLoop()
```



Exemple wxPython

```
class MyFrame(wx.Frame):
  def __init__(self, parent, id, title):
    wx.Frame.__init__(self, parent, id, title)

    self.Bind(wx.EVT_SIZE, self.OnSize)
    self.Bind(wx.EVT_MOVE, self.OnMove)

    panel = wx.Panel(self, -1)
    label1 = wx.StaticText(panel, -1, "Size:")
    label2 = wx.StaticText(panel, -1, "Pos:")
    self.sizeCtrl = wx.TextCtrl(panel, -1, "")
    self.posCtrl = wx.TextCtrl(panel, -1, "")
    self.panel = panel
```

Exemple wxPython

```
sizer = wx.FlexGridSizer(2, 2, 5, 5)
sizer.Add(label1)
sizer.Add(self.sizeCtrl)
sizer.Add(label2)
sizer.Add(self.posCtrl)

border = wx.BoxSizer()
border.Add(sizer, 0, wx.ALL, 15)
panel.SetSizerAndFit(border)
self.Fit()
```

Exemple wxPython

```
def OnSize(self, event):
    size = event.GetSize()
    self.sizeCtrl.SetValue("%s, %s"
        % (size.width, size.height))
    event.Skip()

def OnMove(self, event):
    pos = event.GetPosition()
    self.posCtrl.SetValue("%s, %s" % (pos.x, pos.y))
```

Ressources

- http ://www.python.org/
- http ://stackoverflow.com/
- livre : Python in a Nutshell, Alex Martelli
- conférences de Guido Van Rossum