Python : Introduction

Christophe Brun <christophe@cbrun.fr>

2018

Introduction

- Python
  - 1991 : Date de sortie de la 1ère version (0.9) Pays Bas par Guido van Rossum
  - Début des langages orientés objets
  - Open source, GPL et copy left
  - Interractif, introspection, interfaçable, mais optimisé aussi bien en vitesse qu'en développement
- Un succès
  - Oui, mais pas à la hauteur
  - L'époque Java/PHP
  - sa force : il est partout mais personne n'est au courant
  - Aujourd'hui, il est enfin reconnu
  - Touche tout les domaines : Big data, embarqué, Finance, 3D, bureautique (libreoffice)

Vocabulaire

- Langage interprété : Ex : Shell, PHP
- Langage compilé : Golang, C, C++
- Langage mixte (pseudo-compilé): Python
- introspection
- Console
- *.py, *.pyc, *.pyo
- mode interractif, mode script

Les versions

- python 2.7 : version encore utilisée et installée par défaut sur les linux, c'est la version de transition vers la 3.X
- python 3.X : nouvelle version
  - non compatible avec la 2.7
  - plus de cohérence sur certaines commandes
    - print est une fonction
- la transition longue car beaucoup de libs, prog à convertir de la 2.7

La console

- `python` : lance la console standard
- `ipython` : Console plus interractif
- `repl.it` : console en ligne

Notre premier programme : le hello world

```
print('Bonjour le monde')
```

Variables

- Tt est objet
- Variable
- Règles de nommage
  - pas de contraintes sauf
  - ne pas utiliser les mots réservés
  - ne pas utiliser les formes : _xx_ __xx ou __xx__

Les types

- int : `a=1`
- bool : `vrai=True`
- float : `f=9.81`
- complexe : `c=2+5j`
- chaine de caractères : `t='Bonjour le monde'`
- les listes : `liste=[12, 'rouge', 'bleu', ]`
- les tuples: `tup=('rouge', 4, 3.4,)`
- les dict: `dico={'cle1': 12, 'cle2': 'texte'}`

Fonctions conditionnelles

- Une fonction : if - then - else
- 2 déclinaisons

```
if x < 0:
    print('x est negatif')
elif x % 2 != 0:
    print('x est positif et impair')
    print ('ce qui est bien aussi !')
else:
    print("x n'est pas negatif et est pair")
```

La forme contractée (ternaire)

```
x=4
y=3
plus_petit = x if x < y else y
```

Fonctions de boucle

- while

```
n = int(input('Entrez un entier [1 .. 10] : '))
while not(1 <= n <= 10) :
    n = int(input('Entrez un entier [1 .. 10], S.V.P. : '))
```

- for

```
for x in [2, 'a', 3.14]:
    print(x)
```

- break : stop la boucle
- continue : pour reprendre
- else : alternative (avec while et for)

Exceptions

La gestion des exceptions dans le code passe par `try .. except`

```
for x in range(-4, 5) :
    try :
        print('{ :.3f}'.format(sin(x)/x), end=' ')
    except ZeroDivisionError :
        print(1.0, end=' ')
```

ATTENTION : try .. except ... pass

```
try:
    print('du code avec une erreur')
    a:=4
except:
    pass
```

Les fonctions

- mot clé : def

```
def bonjourLeMonde():
    print('Bonjour le monde')

bonjourLeMonde()
```

- Avec paramètres

```
def bonjourAToi(prenom):
    print('Bonjour %s' % prenom)

bonjourAToi('christophe')
```

Espace de nommage / lib / import

- import
- from ... import

```
from math import pi
print(pi)
```

- découpage du code

```
# fichier utils.py
def bonjourToi(prenom):
    print('Bonjour %s' % prenom)
```

Dans notre fichier main.py

```
import utils

utils.bonjourToi('christophe')
```

Packages, modules

- philosophie de Python
  - il existe certainement un package qui fait le boulot
  - Par défaut il fait 90% du boulot
  - Me reste donc très peu de code à faire
- Si je commence à faire trop de code, c'est que j'ai loupé une lib
- Si vraiment, la lib n'existe pas, je me repose la question
- Si mais Si vraiment je n'ai rien trouvé alors je code mais je partage pour répondre au point 1

Faire un server web en 5 ligne

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

@app.route('/')
def bonjourLeMondet():
    return "Bonjour le monde"
```

```
(flask) codes FLASK_APP=server01.py flask run
 * Serving Flask app "server01"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Classes / Objects

- Une classe : attributs et méthodes

```
class Voiture:

  couleur = ''

  def roule(self):
    print('La voiture roule')

  def setCouleur(self, couleur):
    self.couleur = couleur

  def getCouleur(self):
    print('la voiture est de couleur %s' % self.couleur)
```

- Utilisation de la classe

```
maVoiture = Voiture()
maVoiture.roule()
maVoiture.getCouleur()
maVoiture.setCouleur('blanche')
maVoiture.getCouleur()
```

Et voilà c'est fini