Prise en main de Python

1 Installation

Si vous êtes sous Linux ou une version récente de MacOS X, il y a de fortes chances que Python soit déjà installé sur votre machine. Sous Windows, par contre, il ne le sera pas (à moins que vous ne l'ayez déjà installé vous-même...)

Dans tous les cas, assurez-vous de disposer d'une version **2.5.x** de python[1]. Si nécessaire, téléchargez-le depuis `http://python.org/download/` et installez-le.

Pour vérification, vous devriez pouvoir faire quelque chose dans ce genre[2] :

```
C :\> python
Python 2.5.1 [...]
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello, World!"
Hello, World!
>>> quit()
C :\>
```

Si ça fonctionne, Bravo ! Vous venez de passer la première étape de votre prise en main de Python !

Pour continuer cette prise en main, vous aurez encore besoin d'un éditeur de texte. N'importe lequel fera l'affaire en principe, mais je vous suggère d'en choisir un qui fournisse au moins la coloration syntaxique pour Python. `Scite` constitue une solution légère et multi-plateforme, mais suivant votre système d'exploitation, d'autres alternatives sont possibles (`Notepad++` sous Windows ; `TextMate` sous MacOS ; `Vim`, `Emacs`, `Geany`, etc. sous Linux ; ...)[3].

2 Tutoriel

Prenez le document "A Quick, Painless Tutorial on the Python Language" de N. Matloff et suivez-en les sections **3 à 17** (vous pourrez **ignorer** les sections **12**, **14.1.5**, **14.3**, et si vous n'êtes pas sur un système Unix aussi la section **17.1**).

Pendant la lecture de ce tutorial, profitez d'expérimenter un peu, au gré des questions qui pourraient se poser, aussi bien en mode interactif qu'en écrivant de petits programmes ou en modifiant les exemples donnés.

... Et n'hésitez pas à poser des questions !

[1] Attention ! certaines distributions Linux ont déjà passé à la 2.6, voire la 3.x ! Dans certains cas, la version 2.5 peut encore être disponible avec une commande du type `python2.5` au lieu de `python`. Sinon, il faudra la réinstaller.

[2] En **gras**, ce que vous devez taper ; le reste vous indique les réponses attendues, qui peuvent varier légèrement selon la plateforme et la version de python...

[3] Python fournit aussi un IDE, nommée IDLE, mais je n'ai jamais été très convaincu...

3 Exercices

En vous basant sur vos connaissances fraîchement acquises et si nécessaire sur d'autres ressources (cf. liste ci-dessous), répondez aux questions suivantes :

3.1 Échauffement

Pour chacune des expressions python ci-dessous, essayez de prédire quelle sera sa valeur, puis vérifiez en la tapant en mode interactif.

1. `("bla"*3)[:-3]`
2. `[1,2,3,4,5][:2]+[1,2,3,4,5][2:]`
3. `'Hello, %s!  This is question %d' % ('World',3)`
4. `10**100+1`
5. (1j)**2

3.2 Délimitation des blocs

Une des caractéristiques les plus frappantes de Python est la délimitation des blocs par l'indentation du code. Quels sont les avantages et les inconvénients de cette manière de faire ?

3.3 Chaînes de caractères

Les chaînes de caractères en python peuvent être délimitées par des guillemets (`"`) ou des apostrophes (`'`), et ce caractère peut être simple ou triple. Par exemple :

```
print 'hello'
print "hello"
print '''hello'''
print """hello"""
```

Quelles sont les différences entre ces 4 variantes ? quel est l'avantage de cette variété ?

3.4 Typage

Pour chacune des affirmations suivantes, choisir la version correcte. Illustrer votre réponse avec un exemple de code. Essayez de trouver trouver un exemple de code dans un autre langage illustrant l'autre possibilité.

1. Python est **implicitement/explicitement** typé
2. Python est **fortement/faiblement** typé[4]
3. Python est **statiquement/dynamiquement** typé

[4] Il existe différentes définition du typage fort. Nous adopterons ici la définition suivante : *un langage est fortement typé si les conversions entre types doivent être explicitement décrites*.

3.5 Encore le typage

Qu'entend-on par l'affirmation "Python supporte le *duck typing*" ? Quels sont les avantages et inconvénients de cette approche du typage ?

3.6 Commentaires

Le code suivant contient deux type de commentaires syntaxiquement distincts :

```
""" Module d'exemple d'utilisation des commentaires """

# WARNING: this code is completely useless!!!

def f(x):
    """ Une fonction vachement utile """
    x += 2 # on ajoute 2...
    # puis on retire 2...
    x -= 2
    return x

# et maintenant le main...

if __name__ == "__main__":
    # on teste notre super-fonction
    print f(1)
```

Quels sont ces deux types ? quelle est leur syntaxe respective ? Quand utiliser l'un, et quand utiliser l'autre ?

3.7 Le "main"

Le code suivant ne diffère de celui de l'exercice précédent que par l'absence du test sur `__name__`.

```
""" Module d'exemple d'utilisation des commentaires """

# WARNING: this code is completely useless!!!

def f(x):
    """ Une fonction vachement utile """
    x += 2 # on ajoute 2...
    # puis on retire 2...
    x -= 2
    return x

# et maintenant le main...

# on teste notre super-fonction
print f(1)
```

Donner un exemple de cas où ces deux codes ne se comporteraient pas de la même manière.

3.8 Un bug minimal...

Ouvrez le fichier `bug.py` et tapez simplement :

```
print "bogué?"
```

Lancez ensuite la commande `python bug.py`... qui générera une erreur.
Pourquoi ? Comment corriger ce problème ?

4 Ressources

- Quelques bons tutoriaux
  - `http://heather.cs.ucdavis.edu/~matloff/Python/PythonIntro.pdf` (que vous connaissez bien !)
  - `http://www.poromenos.org/tutorials/python`
- La documentation officielle de python (contient aussi un tutorial)
  Attention de bien utiliser la version de la doc correspondant à votre version de python... (ici, python 2.5)
  - `http://www.python.org/doc/2.5/`
    Une fois le langage pris en main, vous utiliserez constamment la "Python library reference" : `http://www.python.org/doc/2.5/lib/lib.html`
- Livres téléchargeables
  - `http://diveintopython.org/`
  - `http://inforef.be/swi/python.htm`
- Des dizaines de vidéos
  - `http://showmedo.com/videos/Python`
- Autres sources en vrac
  - `http://en.wikipedia.org/wiki/Python_programming_language`
  - `http://sebsauvage.net/python/` et surtout l'excellent `http://sebsauvage.net/python/snyppets/index.html`

... et un jour ou l'autre, vous aurez besoin d'un débogueur. Winpdb représente une excellente solution :

- `http://www.digitalpeers.com/pythondebugger/`
  Et vous avez de la chance, Matloff a aussi écrit un tutorial pour winpdb : `http://heather.cs.ucdavis.edu/%7Ematloff/winpdb.html`