(Trop) Brève introduction à Python

S. Vialette

IGM-LabInfo, Université Paris-Est

21 janvier 2008

Qu'est-ce que c'est ?

`http ://fr.wikibooks.org/wiki/Programmation_Python/Introduction`

- `Python` est un langage de script de haut niveau, structuré et open source. Il est multi-paradigme et multi-usage.
- Développé à l'origine par Guido Van Rossum en 1993, il est, comme la plupart des applications et outils open source, maintenu par une équipe de développeurs un peu partout dans le monde.
- Conçu pour être orienté objet, il n'en dispose pas moins d'outils permettant de se livrer à la programmation fonctionnelle ou impérative ; c'est d'ailleurs une des raisons qui lui vaut son appellation de "*langage agile*".
- Parmi les autres raisons, citons la rapidité de développement (qualité propre aux langages interprétés), la grande quantité de modules fournis dans la distribution de base ainsi que le nombre d'interfaces disponibles avec des bibliothèques écrites en `C`, `C++` ou `Fortran`.

Utilisation

Python se prête à un grand nombre de tâches :

- développement réseau

twisted	Framework réseau
PYRO	Python Remote Objects
PyLinda	Distributed Computing Made Easy
...	

- Framework web

Django	The Web framework for perfectionists with deadlines
Plone CMS	Open Source Content Management
Zope	Open source application server
Turbogears	Web development
...	

Utilisation

- GUI

Tk	A thin object-oriented layer on top of Tcl/Tk
wxPython	Cross-platform Python GUI toolkit
PyQt	Python bindings for Trolltech's Qt application framework
PyGTK	Python bindings for GTK+ application framework
...	

- XML

PyXML	XML package for Python
4suite	Open-source platform for XML and RDF processing
pyRXP	The fastest XML parser
...	

Utilisation

- Mutimédia

  `PyGame` Python modules designed for writing games (`SDL`)

  `pyMedia` Python module for `wav`, `mp3`, `ogg`, `avi`, `divx`, `dvd`, ...

  `Shtoom` VoIP softphone

  `PiTiVi` Open source video editor

  ...

- PDF

  ReportLab Toolkit Industry-strength PDF generating solution

  `pyPdf` Pure-Python library built as a PDF toolkit

  ...

Utilisation

- et bien d'autres

`BioPython`	Tools for computational molecular biology.
`networkx`	Creation, manipulation, and study of the structure, dynamics, and functions of complex networks
`NLTK`	The Natural Language Toolkit
`PLY`	Implementation of `lex` and `yacc` parsing
`SciPy`	Open-source software for mathematics, science, and engineering
`Simpy`	Simulation in `Python`
`jython`	Implementation of `Python` written in 100% pure `Java`
`unittest`	Unit testing framework
...	

Environnements de Développement

`emacs`	avec `python-mode` bien sûr
`SPE`	Stani's Python Editor
`PyDev`	`Python` development enviroment for `Eclipse`
`geany`	A fast and lightweight IDE
`Boa Constructor`	Cross platform Python IDE
`PyPE`	Editor written in Python with the `wxPython` GUI toolkit
...	

Interpréteur

```
$ python
Python 2.5.1 (r251:54863, Oct  5 2007, 13:36:32)
[GCC 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello world !'
hello world !
>>> import sys
>>> sys.stdout.write('hello world  !\n')
hello world  !
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__stderr__', '__stdin__',
 '__stdout__', '_current_frames', '_getframe', 'api_version', 'argv',
 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyright',
 'displayhook', 'exc_clear', 'exc_info', 'exc_type', 'excepthook', 'exec_prefix',
 'executable', 'exit', 'getcheckinterval', 'getdefaultencoding', 'getdlopenflags',
 'getfilesystemencoding', 'getrecursionlimit', 'getrefcount', 'hexversion', 'maxint',
 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks', 'path_importer_cache',
 'platform', 'prefix', 'ps1', 'ps2', 'pydebug', 'setcheckinterval', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'settrace', 'stderr', 'stdin', 'stdout', 'subversion',
 'version', 'version_info', 'warnoptions']
>>> # <Crt-D> pour quitter ...
...
$
```

Interpréteur

```
$ python --help
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit (also --help)
-i     : inspect interactively after running script, (also PYTHONINSPECT=x)
         and force prompts, even if stdin does not appear to be a terminal
-m mod : run library module as a script (terminates option list)
-O     : optimize generated bytecode (a tad; also PYTHONOPTIMIZE=x)
-OO    : remove doc-strings in addition to the -O optimizations
-Q arg : division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew
-S     : don't imply 'import site' on initialization
-t     : issue warnings about inconsistent tab usage (-tt: issue errors)
-u     : unbuffered binary stdout and stderr (also PYTHONUNBUFFERED=x)
         see man page for details on internal buffering relating to '-u'
-v     : verbose (trace import statements) (also PYTHONVERBOSE=x)
-V     : print the Python version number and exit (also --version)
-W arg : warning control (arg is action:message:category:module:lineno)
-x     : skip first line of source, allowing use of non-Unix forms of #!cmd
file   : program read from script file
-      : program read from stdin (default; interactive mode if a tty)
arg ...: arguments passed to program in sys.argv[1:]
$
```

Exécuter un programme Python

Tout cours d'introduction à un langage de programmation ne saurait commencer sans `Hello words !`.

```
#!/usr/bin/env python

helloWordString = "Hello word !"
print helloWordString
```

Il existe plusieurs façons d'exécuter un programme Python.

Depuis l'interpréteur Python

```
$ python
Python 2.5.1 (r251:54863, Oct  5 2007, 13:36:32)
[GCC 4.1.3 20070929 (prerelease)
 (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> helloWordString = "Hello word !"
>>> print helloWordString
Hello word !
>>>
```

- >>> est l'*invite* de l'interpréteur Python.
- `Ctrl-D` pour quitter.

Depuis un fichier

```
$ ls sayHello.py
sayHello.py
$ file sayHello.py
sayHello.py: ASCII text
$
$ cat sayHello.py
helloWordString = "Hello word !"
print helloWordString
$
$ python sayHello.py
Hello word !
$
```

Depuis un fichier

```
$ ls sayHello.py
sayHello.py
$ cat sayHello.py
#!/usr/bin/env python

helloWordString = "Hello word !"
print helloWordString
$
$ ls -l sayHello.py
-rw-r--r-- 1 vialette vialette 78 2007-12-18 23:47 sayHello.py
$ ./sayHello.py
bash: ./sayHello.py: Permission denied
$ chmod +x sayHello.py
$ ls -l sayHello.py
-rwxr-xr-x 1 garulfo garulfo 78 2007-12-18 23:47 sayHello.py
$ ./sayHello.py
Hello word !
$
```

Depuis l'interpréteur

- La fonction `execfile(`*filename*`)` exécute le code Python se trouvant dans le fichier *filename*.

```
$ python
Python 2.5.1 (r251:54863, Oct  5 2007, 13:36:32)
[GCC 4.1.3 20070929 (prerelease)
 (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> execfile("sayHello.py")
Hello word !
>>>
```

Depuis l'interpréteur

- À noter qu'il existe également la fonction `exec(str)` qui exécute le code Python se trouvant dans la chaîne de caractères *str*.

```
$ python
Python 2.5.1 (r251:54863, Oct  5 2007, 13:36:32)
[GCC 4.1.3 20070929 (prerelease)
 (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> exec ("helloWordString = 'Hello word !'" + \
... "\n" + "print helloWordString")
Hello word !
>>> myString = """
... helloWordString = "Hello word !"
... print helloWordString
... """
>>> exec(myString)
Hello word !
```

Depuis un autre programme

```c
#include <Python.h>

int main(int argc, char *argv[]) {
   Py_Initialize();
   PyRun_SimpleString("helloWordString = \"Hello word !\"");
   PyRun_SimpleString("print helloWordString");
   return(0);
}
```

```
$ ls /usr/include/python2.5/Python.h
/usr/include/python2.5/Python.h
$ ls /usr/lib/libpython2.5.so
/usr/lib/libpython2.5.so
$ gcc -o sayHello sayHello.c -I/usr/include/python2.5/ -lpython2.5
$ ./sayHello
Hello word !
$
```

Typage

- **langage à typage dynamique**
  Un langage dans lequel les types sont découverts à l'exécution, l'inverse du typage statique.
- **langage fortement typé**
  Un langage dans lequel les types sont toujours appliqués (un entier ne peut par exemple pas être traité comme une chaîne sans conversion explicite).

`Python` est à la fois à typage dynamique (il n'utilise pas de déclaration de type explicite) et fortement typé (une fois qu'une variable a un type, cela a une importance).