A

L'histoire de Python

Le langage Python a été créé à la fin des années 1980 à l'Institut national de recherches mathématiques et informatiques de Hollande (le CWI) par Guido van Rossum. Par commodité, nous utiliserons le raccourci GvR pour nommer ce dernier dans la suite de cette annexe.

Le langage ABC

GvR a rejoint le CWI en 1983 dans l'équipe en charge du développement du langage ABC, sur lequel il a travaillé pendant 3 ans. Cette période a fortement influencé GvR sur la conception de Python, qui hérite de certains des concepts d'ABC.

Le langage ABC est un langage de programmation interactif fortement typé, qui a été pensé pour remplacer le Basic, largement répandu à l'époque, en fournissant un environnement particulier ainsi que d'autres caractéristiques notables, comme le typage spécifique des données et la syntaxe par indentation.

Environnement de développement

La particularité de l'environnement d'ABC est qu'il n'est pas nécessaire de sauvegarder fonctions et procédures dans des fichiers sources : une fois entrées dans l'environnement interactif, leur saisie dans l'invite de commande (le prompt, symbolisé sous ABC par >>>) les conservent automatiquement d'une exécution à l'autre.

Un système de complétion de code permet en outre de faciliter la saisie des commandes. Enfin, un historique autorise à revenir en arrière sans limite.

Types de données

ABC fournit 5 types, qui permettent d'exprimer toutes formes de structures de données :

- le type `nombre`, pour les entiers et les réels, sans aucune limitation de taille, hormis la mémoire physique disponible de la machine ;
- le type `text`, pour les chaînes de caractères ;
- le type `list`, pour manipuler des collections d'éléments ordonnés ;
- le type `compound`, équivalent au type `list` mais non modifiable. C'est une sorte de *recordset* sans étiquette ;
- le type `table`, qui définit un certain nombre de clés uniques, et associe une valeur à chacune d'entre elles. Ce type est comparable à une combinaison de deux instances de type `list` : les clés et les valeurs.

Exemple de manipulation de table sous ABC

```
>>> PUT {} IN distance_paris
>>> PUT 300 IN distance_paris["Dijon"]
>>> PUT 220 IN distance_paris["Lille"]
>>> PUT 770 IN distance_paris["Marseille"]
>>> WRITE distance_paris["Dijon"]
300
>>> WRITE distance_paris
{["Dijon"]: 300; ["Lille"]: 220; ["Marseille"]: 770}
```

Il n'est pas nécessaire ici de signaler que la variable `distance_paris` est de type `table`, ABC le fait automatiquement lors de la première affectation.

Indentation du code

L'imbrication de code ABC n'est pas faite comme en C ou en Pascal par des accolades ou des délimiteurs `begin..end`. C'est l'indentation des lignes qui détermine le niveau d'imbrication du code.

Exemple de définition de la fonction message

```
HOW TO DISPLAY message:
    FOR line IN message:
        WRITE line /
```

```
>>> DISPLAY "ABC est l'ancêtre de Python"
ABC est l'ancêtre de Python
```

> **EN SAVOIR PLUS** **Le langage ABC**
>
> Pour plus d'informations sur le langage ABC, Le lecteur interessé peut se référer à l'ouvrage *The ABC Programmer's Handbook* (Geurts, Meertens, Pemberton, aux Éditions Prentice-Hall).

Le projet ABC n'a malheureusement pas eu le succès escompté en dehors du cercle du CWI et est resté relativement confidentiel.

Le projet Amoeba

GvR a rejoint en 1986 le projet Amoeba, un système d'exploitation distribué. Il a été chargé dans ce cadre de créer un langage de script pour manipuler le système plus facilement. Les contraintes du projet étaient relativement souples pour laisser GvR, fort de son expérience passée, mettre au point une première version de ce qui allait devenir le langage Python.

GvR implémenta ce langage de script en essayant de supprimer toutes les contraintes et frustrations qu'il avait vécues avec ABC.

Par exemple, ABC ne permettait pas de lire et écrire dans un fichier, et cette fonctionnalité ne pouvait pas être ajoutée facilement au langage, dénué de tout concept de bibliothèque ou de tout système de programmation d'entrée/sortie souple.

L'extensibilité fut le premier chantier de GvR car il voulait que Python, même si son objectif premier était de fonctionner pour Amoeba, puisse être étendu facilement par des programmeurs tiers en se basant sur un socle commun de primitives et des points d'entrée simples.

L'idée de rendre le langage portable, c'est-à-dire fonctionnel sur plusieurs plates-formes comme Amoeba bien sûr, mais aussi sur MS-Windows, Unix ou Macintosh, était aussi un objectif de GvR.

À un moment de l'histoire de l'informatique où les ordinateurs commençaient à envahir les entreprises et les foyers, le manque d'extensibilité et de portabilité condamnait ABC à un rôle mineur, et GvR, en visionnaire, a su ouvrir les portes de son langage de script.

GvR conçut les premières versions du langage qu'il appela Python, à la gloire des Monty Python dont il était fan. Lorsque la liste de diffusion fut créée plus tard, il n'était pas rare de voir régulièrement des messages de fans des Monty Python, ne pensant pas avoir affaire à un langage de programmation.

Dans les premières versions du langage, le système d'extension qui permettait d'ajouter de nouveaux types d'objets à Python à partir d'un fichier de code Python ou un fichier compilé en C, C++ ou encore en Fortran, a tout de suite été adopté et plébiscité par l'entourage de GvR.

Les versions de Python s'enchaînèrent jusqu'à la version 1.2 en 1995, date à laquelle GvR quitta le CIW pour rejoindre le CNRI (Corporation of National Research Initiatives) à Reston en Virginie (USA) pour continuer ses travaux.

Le CNRI

Cet organisme finança le développement de Python pendant cinq ans, par le biais de fonds de recherche. La Python Software Activity (PSA), le Python Consortium et des sociétés privées apportèrent également des fonds pour soutenir l'avancée du langage. Le travail au CNRI a permis de sortir plusieurs versions de Python, de la 1.3 à la 1.6.

En 2000, GvR prit la décision de quitter le CNRI, car les fonds alloués à Python étaient de plus en plus maigres et les développeurs dispatchés sur d'autres projets. De plus, l'organisme ne semblait pas très favorable au logiciel libre.

Ce départ fut relativement tendu et le CNRI insista pour modifier le texte de la licence de Python pour garder une mainmise, en provoquant à l'époque une grande inquiétude de la communauté sur la suite des événements. Accompagné de trois autres développeurs du CNRI, GvR fonda le PythonLabs, et rejoignit la startup Californienne BeOpen.com.

PythonLabs et BeOpen.com

Avec l'arrivée à BeOpen.com, l'équipe du PythonLabs passa directement de la version 1.6 à la 2.0, en intégrant des améliorations majeures, comme les *list comprehensions*, le support étendu du XML, un nouveau système de ramasse-miettes cyclique, et une nouvelle licence plus orientée Open Source.

Le projet Python 3000 était lancé en parallèle, pour accueillir la nouvelle version de Python, vouée à contenir des modifications incompatibles avec les versions 2.x, pour corriger des erreurs de conception du langage.

Un système de warning (d'avertissement) a alors été introduit pour permettre de spécifier les compatibilités ascendantes et descendantes du langage.

En d'autres termes, toute introduction de nouvelle fonctionnalité incompatible avec la version en cours, peut être aperçue et utilisée par le biais du module `__future__`, et

toute fonctionnalité qui n'existera plus dans la version suivante affiche un warning lorsqu'elle est utilisée.

Ce système est d'ores et déjà utilisé dans la série des versions 2.x.

Python Software Foundation et Digital Creations

Moins d'un an après l'arrivée à BeOpen.com, l'équipe de PythonLabs déménage une nouvelle fois pour rejoindre Digital Creation, la société qui allait devenir par la suite Zope Corp.

En Mars 2001, la Python Software Foundation voit le jour et remplace la PSA, annoncée par GvR à la neuvième conférence Python, et sponsorisée par les sociétés Digital Creation et ActiveState, contributeurs majeurs de la communauté Python de l'époque.

Le premier comité directeur réunissait des membres de PythonLabs et des responsables des deux sociétés, à savoir : Dick Hardt, David Ascher, Paul Everitt, Fredrik Lundh, Tim Peters, Greg Stein, Guido van Rossum et Thomas Wouters.

Les versions de Python se sont ensuite enchaînées jusqu'à la toute dernière en 2005 au moment de l'écriture de ce livre (2.4.2).

Python et Zope

Python a joué un rôle fondamental pour le développement du framework Zope, et inversement, Zope a beaucoup contribué au développement du langage.

Le texte ci-dessous est une interview de Paul Everitt, créateur de Digital Creations, l'entreprise qui conçoit Zope, et qui répond à la question suivante :

Quelle a été la place de Python dans l'histoire de Zope et Digital Creations ?

En 1995, la société Digital Creations a été créée pour mettre en ligne des journaux. Nous avons utilisé Python pour concevoir l'architecture de notre plate-forme de journaux en ligne et avons beaucoup participé à la communauté Python.

Jim Fulton (ndlr : directeur technique actuel) a rejoint l'entreprise l'année suivante et a lancé l'idée de publier des objets Python via le Web. Le framework « Bobo » était né et distribué sous licence Open Source.

Une application commerciale nommée Principia et entièrement écrite en Python faisait également partie de nos travaux.

En 1997, nous avons été sortis du consortium des journaux et conservé la propriété intellectuelle. En 1998, Hadar Pedhazur a investi dans l'entreprise, et nous avons concentré nos travaux Python, dans un seul et même produit Open Source : Zope. Une large communauté de développeurs pour la plupart issus de l'Open Source s'est construite autour du projet.

Python était dans notre sang dès le départ. Jim et moi sommes allés à la toute première conférence Python publique (20 personnes). Jim était alors considéré comme un, sinon le principal contributeur du noyau du langage Python.

Grâce à Python, nous étions capables de construire des systèmes web, comme des systèmes de petites annonces électroniques très dynamiques en un temps record, ce qui nous rendait très compétitifs.

Parallèlement, lorsque nous avons conçu Principia, le serveur d'applications propriétaire, nous avions décidé de cacher Python. Cette décision a eu un énorme impact aussi bien positif que négatif, sur le fonctionnement de Zope. Les idées de gérer tout un site à travers des interfaces d'administration en ligne, de stocker des portions de code restreint dans une base de données (ndlr : la ZODB), et d'étendre le serveur par des paquets d'extension, vinrent de cette décision.

Nous avons aussi apporté une nouvelle audience pour le langage Python, puisque beaucoup de gens qui choisissaient Zope, n'avaient jamais fait de Python auparavant (à la première conférence Zope à l'ICP8, la moitié de l'audience n'avait jamais utilisé Python avant Zope).

Malheureusement le choix de cacher Python a également généré une confusion sur ce qu'était Zope. La communauté Python jugeait Zope 2 comme un framework pas très Pythonique. De plus, Zope 2 lui-même vivait une crise d'identité : était-ce un produit destiné aux intégrateurs, ou un produit orienté développeur ?

Zope 3 a résolument pris un tournant en orientant le framework vers un outil pour développeurs.

Des journalistes comme Jon Udell ou Edd Dumbill considèrent que Zope est l'un des frameworks où l'Open Source a réellement vécu des innovations, pour la plupart issues des idées de Jim Fulton. Le langage Python influença beaucoup Jim dans ses idées, et offrit à Zope des fonctionnalités magnifiques : l'idée de publier des objets sur le Web est devenu un sujet informatique d'actualité, 9 ans après que Jim l'eut fait.

Une base de données transactionnelle distribuée d'objets Python, utilisée dans des sites commerciaux énormes, c'est un résultat impressionnant. L'histoire de Zope et Python est maintenant vieille de 10 ans. Place maintenant à un nouveau chapitre : Zope 3 et son souhait d'être plus Pythonique que son prédécesseur et d'intéresser davantage la communauté Python. »

— Paul Everitt, fondateur de Digital Creations.

B

Bibliothèques tierces

La philosophie *Batteries Included* de Python rencontre ses limites lorsque des fonctionnalités très spécifiques sont recherchées. Cette limitation n'est cependant pas bloquante grâce à la facilité d'extension du langage : il est possible de trouver des bibliothèques tierces pour la quasi-totalité des besoins.

D'autre part, certains modules initialement présents dans la bibliothèque standard ont été volontairement délaissés au fur et à mesure des versions du langage, pour préférer des solutions externes.

Cette annexe liste un certain nombre de bibliothèques externes, les plus fréquemment utilisées, organisées par thèmes :

- bases de données ;
- traitement de texte ;
- packaging, distribution ;
- tests fonctionnels et contrôle qualité ;
- MS-Windows ;
- interfaces graphiques ;
- reporting et conversion ;
- jeux et 3D ;
- audio et vidéo ;
- bibliothèques scientifiques ;
- Web.

Chacune des bibliothèques est présentée par un court texte et une URL suit le même schéma d'installation, présenté ci-après. L'annexe s'achève sur la présentation de *cheeseshop*, qui offre à Python ce que *CPAN* est à Perl.

Installer une bibliothèque externe

Toutes les bibliothèques externes présentées dans cette annexe sont très simples à installer car basées sur le module `disutils`, présenté au chapitre 13. Ces bibliothèques externes sont souvent livrées dans un fichier compressé sous la forme `NomDuPaquet-version.zip` ou `NomDuPaquet-version.tar.gz`.

Installer une extension se fait en trois étapes :

1. décompression du paquet, par l'outil `tar` ou équivalent ;
2. construction du paquet dans le répertoire de décompression, par l'option `build` du script `setup.py` ;
3. installation du paquet dans Python, par l'option `install` du script `setup.py`.

Lorsque la première étape est effectuée, on retrouve dans le répertoire décompressé une structure commune à toutes ces bibliothèques, à savoir :

- un fichier `setup.py`, qui contient la configuration et l'appel au framework distutils ;
- un fichier `setup.cfg`, optionnellement présent, qui contient des informations supplémentaires, lorsqu'une compilation est nécessaire ;
- des informations sur l'extension, contenues dans les fichiers `INSTALL` et `README` ;
- un certain nombre de fichiers sources.

La construction du paquet prépare un sous-répertoire `build` qui contient les éléments à fournir à Python.

Enfin, la dernière étape recopie ces fichiers dans le répertoire `site-packages` de l'installation de Python. Elle peut donc nécessiter les droits d'administrateur.

Installation de lxml

```
tziade@Tarek:/home/packages$ tar -xzf lxml-0.7.tgz
tziade@Tarek:/home/packages$ cd lxml
tziade@Tarek:/home/packages/lxml$ python setup.py build
running build
running build_py
creating build
creating build/lib.linux-i686-2.4
```

```
creating build/lib.linux-i686-2.4/lxml
creating build/lib.linux-i686-2.4/lxml/tests
[...]
tziade@Tarek:/home/packages/lxml$ sudo python setup.py install
running install
running build
running build_py
running build_ext
running install_lib
creating /usr/lib/python2.4/site-packages/lxml
creating /usr/lib/python2.4/site-packages/lxml/tests
copying build/lib.linux-i686-2.4/lxml/tests/test_etree.py -> /usr/lib/
python2.4/site-packages/lxml/tests
[...]
```

Une fois l'installation effectuée, le nouveau module doit être disponible dans le prompt.

Vérification de l'installation

```
tziade@Tarek:/home/packages/lxml$ python
Python 2.4.1 (#2, Mar 30 2005, 21:51:10)
[GCC 3.3.5 (Debian 1:3.3.5-8ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import lxml
>>>
```

> **À SAVOIR Contrôler la bonne installation d'une bibliothèque**
>
> Certaines bibliothèques fournissent des tests (souvent basés sur le framework `pyUnit`) qui peuvent être lancés pour vérifier que l'installation est correcte et que tout fonctionne comme prévu.

Bases de données

Python fournit les briques de base (DBAPI) à tout type de connecteur de base de données et propose quelques modules d'accès à des formats ultra standards comme BerkeleyDB. Cependant, aucun connecteur aux SGBD courants n'est intégré dans la bibliothèque standard.

Toutes les bases de données du marché peuvent être bien évidemment attaquées depuis Python, et cette section présente les connecteurs les plus courants. Elle inclut également un connecteur LDAP et un *mapper* SQL.

Gadfly

Codé en Python, Gadfly est un mini-système SGBD complet. L'installation de cette extension permet de créer des fichiers de stockage qui peuvent être manipulés via le langage SQL, en mode direct ou en mode client-serveur.

Gadfly supporte une charge relativement limitée et est en général utilisé pour le prototypage d'applications client-serveur : la norme DBAPI étant respectée, ce connecteur peut être facilement interchangé sans modification de code.

▸ http://gadfly.sourceforge.net/

pysqlite

pysqlite est un connecteur compatible DBAPI vers le système SQLite. Ce système léger de SGBD (non client-serveur) est de plus en plus prisé dans les applications qui ont des besoins de stockage simples et un accès unique aux données, comme les applications web. sqlite est parfois plus rapide que les SGBD client-serveur classiques.

▸ http://initd.org/tracker/pysqlite

mysql-python

mysql-python est un connecteur vers le célèbre SGBD MySQL.

▸ http://sourceforge.net/projects/mysql-python

psycopg

Connecteur pour PostgreSQL.

▸ http://initd.org/projects/psycopg1

ODBC

On peut accéder à certaines bases de données sous MS-Windows par le biais de l'ODBC (Open DataBase Connectivity). La bibliothèque Python Win32 Extensions fournit un certain nombre de modules dédiés à MS-Windows, dont le module `ODBC`.

▸ http://www.python.org/windows/win32/

python-ldap

python-ldap expose les API de OpenLDAP 2.x et quelques utilitaires annexes (lectures LDIF). Cette bibliothèque permet d'utiliser tout type de serveur compatible avec le standard LDAP (OpenLDAP, ActiveDirectory, etc.).

- http://python-ldap.sourceforge.net/

SQLObject

SQLObject est une extension qui offre une fonctionnalité intéressante de sérialisation-désérialisation d'objets vers un SGBD (mapper). À l'image de `Pickle`, le développeur manipule des objets sans se soucier des requêtes SQL nécessaires pour gérer la persistance : chaque classe est représentée par une table, chaque instance par une ligne de la table, et chaque attribut par une colonne de la table.

- http://sqlobject.org

Traitement de texte

Le besoin le plus fréquent en traitement de texte est la gestion du format XML. La bibliothèque standard propose des modules dédiés mais qui sont de plus en plus délaissés par les développeurs, en raison de problèmes de performances et surtout par un manque cruel de simplicité : manipuler un fichier XML avec ces modules nécessite un effort relativement important pour un développeur Python, habitué à plus de concision et de simplicité ou ne fournit pas de performances correctes.

Cette section présente deux extensions dédiées au traitement du XML, plus performantes et naturelles à utiliser car basées sur le principe des curseurs : ElementTree et lxml.

Un autre besoin récurrent est le traitement de fichiers HTML *non stricts* : ce type de format n'est pas lisible par des bibliothèques XML et doit être traité spécifiquement. L'extension BeautifulSoup propose un outil spécialisé.

ElementTree

Bibliothèque d'accès à XML simple d'usage, basée sur des curseurs sur le DOM. Rapide et pythonique, ElementTree est très agréable à utiliser malgré quelques

défauts (ajout d'alias sur les namespaces par exemple) et quelques manques (pas de support d'XSLT).

▸ http://effbot.org/zone/element-index.htm

lxml

lxml est un bind Python codé en Pyrex de libxml et libxslt qui fournit les mêmes API qu'ElementTree. Très rapide, pythonique et puissante, probablement la meilleure bibliothèque XML actuelle.

▸ http://codespeak.net/lxml/

Beautiful Soup

Lorsqu'il s'agit de lire du contenu HTML non strict, le développeur utilise en général les modules HTMLParser ou SGMLParser de la bibliothèque standard, ou dans certains cas, scrute le contenu avec une expression régulière.

Beautiful Soup propose une alternative intéressante en scrutant le texte à la recherche de balises, paramètres ou contenu.

▸ http://www.crummy.com/software/BeautifulSoup/

Packaging, distribution

Outre l'outil standard distutils, il existe une extension de plus en plus utilisée pour la distribution de programmes Python, à savoir setuptools.

setuptools

setuptools est une extension à distutils qui permet de distribuer un programme ou une extension Python en fournissant à l'installeur des fonctionnalités supplémentaires, comme :

- le téléchargement et l'installation automatique des dépendances ;
- la création automatique d'un PythonEggs, archive zippée contenant l'ensemble des fichiers à installer ;

- l'upload du logiciel dans le système PyPI (CheeseShop) ;
- etc.

> http://peak.telecommunity.com/DevCenter/setuptools

Tests fonctionnels et contrôle qualité

En matière de tests, les modules `unittest` et `doctest` couvrent tous les besoins basiques mais ne permettent pas de mettre en œuvre facilement des tests fonctionnels, qui restent spécifiques au type d'interface de l'applicatif.

Les extensions qui offrent un environnement de développement de tests fonctionnels, que ce soit pour des applications web ou desktop, fonctionnent toutes sur le même principe : elles mettent en œuvre un pont entre les tests et l'interface utilisateur. Twill et Funkload permettent de tester des applications web ; guitest le fait pour des applications GTK.

Enfin, PyLint et PyFlakes offrent de bon garde-fous, complémentaires aux tests, pour s'assurer de la qualité du code écrit.

Twill

Twill fournit un langage de script simple qui permet de tester une application web via des scripts Python. L'outil permet d'effectuer des requêtes vers le serveur web et d'analyser les résultats.

> http://www.idyll.org/%7Et/www-tools/twill.html

FunkLoad

FunkLoad est un outil basé sur webunit, qui permet d'écrire des tests fonctionnels en Python. Cet outil permet également de tester la montée en charge et génère des rapports complets. Les tests peuvent être conçus facilement via le navigateur grâce à TCPWatch.

> http://funkload.nuxeo.org/

guitest

Cet outil fournit des classes de base pour effectuer des tests unitaires sur des applications PyGtk.

> http://gintas.pov.lt/guitest/

PyLint

PyLint est un outil qui teste le code à la recherche d'erreurs ou de signes de mauvaise qualité. Ce programme est facilement configurable et extensible. Il est comparable à PyChecker mais propose plus de tests.

> http://www.logilab.org/projects/pylint

Pyflakes

Cet outil contrôle le code à la recherche d'erreurs, de code mort (impossible à appeler) ou de directives d'importation inutiles. Contrairement à PyChecker, cet outil n'exécute pas le code testé, ce qui le rend plus rapide et plus sécurisé.

> http://divmod.org/projects/pyflakes

MS-Windows

Il existe des bibliothèques spécialisées dans la programmation sur plate-forme MS-Windows et la technologie COM/ActiveX, à savoir les bibliothèques Win32 Extensions et win32com.

Win32 Extensions

La bibliothèque `win32` présentée dans la section base de données pour l'ODBC, contient également des modules pour :

- les API win32 (un fichier d'aide `WinHelp` avec la liste des méthodes est fourni) ;
- les services NT ;
- les Memory Mapped Files ;
- les API win32pipe et win32 ;

- Les timers win32 ;
- etc.

▸ http://www.python.org/windows/win32/

win32com

win32com permet de programmer des clients ou des serveurs COM/ActiveX.

▸ http://www.python.org/windows/win32com/

Interfaces graphiques

Il existe plusieurs toolkits graphiques qui peuvent être utilisés par le biais de bibliothèques Python, pour remplacer Tkinter. Les plus répandus sont wxPython, PyQT et PyGTK.

wxPython

wxPython est une bibliothèque d'accès au toolkit wxWidgets, qui est de loin le plus portable des systèmes d'interface. Il existe en outre des outils de conception d'interfaces qui génèrent du code Python compatible avec wxPython, comme wxGlade.

▸ http://www.wxpython.org/

PyQT

PyQT est un bind vers le toolkit graphique Qt de Trolltech et offre un accès à des widgets très avancés, comme le contrôle texte Qscintilla, utilisé par certains éditeurs comme Eric3. L'éditeur QT designer est en outre l'un des plus puissants existant pour la conception d'interfaces graphiques. Attention cependant aux licences en fonction des cas d'utilisation et des plates-formes.

▸ http://www.riverbankcomputing.co.uk/pyqt/

PyGTK

PyGTK fournit un lien entre Python et le toolkit GTK+ (The GIMP toolkit), utilisé par l'environnement Gnome. L'outil Glade peut être utilisé pour concevoir des interfaces GTK et présente la même interface que wxGlade (qui s'en inspire).

▸ http://www.pygtk.org/

Reporting et conversion

En termes de reporting, il existe une bibliothèque Open Source incontournable nommée ReportLab éditée par la société éponyme, qui permet de générer des documents PDF et possède des fonctionnalités très puissantes.

RML2PDF est un outil de conversion du format RML vers PDF et enfin rest2web une bibliothèque de création de sites web statiques générés à partir de fichiers écrits au format reStructuredText.

ReportLab

Le toolkit ReportLab permet de concevoir en Python des systèmes de génération de PDF et fournit :

- un moteur de mise en page, Platypus ,
- une bibliothèque étendue de widgets et de formes ;
- des points d'entrée pour toutes sources de données ;
- etc.

▸ http://www.reportlab.org/rl_toolkit.html

RML2PDF

Le format Report Markup Language (RML) créé par la société ReportLab permet de définir simplement un document dans un fichier de description XML. Un outil de conversion, nommé RML2PDF se charge ensuite de le convertir en PDF. Cet outil est payant mais il existe une variante Open Source éditée par Tiny ERP.

▸ http://openreport.tiny.be/index.py/static/page/trml2pdf

reStructuredText

reStructuredText est un format texte très utilisé pour la documentation de projets Python et pour l'écriture des docstrings des modules de code. Il introduit une syntaxe très simple qui permet la mise en page de texte.

Ce format est également très utilisé dans les systèmes wikiwikiweb, pour offrir aux utilisateurs un format simple à écrire et aussi riche que le HTML. Il est facilement convertible en rendu HTML par des outils comme rest2html.

- http://docutils.sourceforge.net/rst.html

rest2web

rest2web permet de générer des pages HTML statiques à partir de documents écrits au format reStructuredText.

- http://www.voidspace.org.uk/python/rest2web/

Jeux et 3D

En termes de programmation de jeux et plus généralement de scènes 3D, Python est un langage de script de choix. Les toolkits Pygame et Soya 3D permettent de bénéficier de la puissance de Python dans ce domaine.

VPython propose quant à lui un environnement de programmation 3D temps réel propice à l'étude de la modélisation.

Il est aussi possible de programmer en plus bas niveau en accédant directement aux bibliothèques 3D par le biais par exemple de PyOpenGL.

Pygame

Pygame fournit des modules d'extension pour la programmation de jeux 3D et de programmes multimédia, basés sur la bibliothèque SDL (Simple DirectMedia Layer).

- http://www.pygame.org/

Soya 3D

Soya 3D est un moteur 3D pour Python, écrit en Pyrex et doté de toutes les fonctionnalités d'un moteur professionnel.

▸ http://home.gna.org/oomadness/fr/soya/

vpython

vpython propose un environnement de programmation 3D complet, en fournissant sa propre version d'IDLE qui permet de programmer et d'animer interactivement des scènes 3D. Très pratique pour l'apprentissage de la mécanique.

▸ http://vpython.org/

PyOpenGL

Module d'extension offrant l'accès aux API d'OpenGL depuis Python.

▸ http://pyopengl.sourceforge.net/

Audio et Vidéo

Le domaine multimédia n'est pas en reste grâce à des bibliothèques très complètes comme PyMedia ou des modules spécifiques comme PyAlsa.

PyMedia

PyMedia propose un ensemble de modules pour manipuler tous types de formats audio et vidéo (`mp3`, `ogg`, `avi`, `mpeg`, etc.), modifier les échantillons par quelques filtres et piloter le matériel.

▸ http://pymedia.org/

PyAlsa

PyAlsa est un wrapper pour le système ALSA (Advanced Linux Sound Architecture)

- http://respyre.org/pyalsa.html

Bibliothèques scientifiques

Cette section regroupe différentes bibliothèques scientifiques spécialisées dans les calculs numériques comme Numerical Python et SciPy, et dans les outils dédiés à des domaines particuliers comme Biopython.

Numerical Python

Numerical Python, qui se nomme maintenant SciPy, est une bibliothèque puissante de fonctions de manipulation de matrices, de transformées de Fourier, et autres utilitaires de calcul.

- http://sourceforge.net/projects/numpy

SciPy

SciPy complète Numerical Python en fournissant des fonctions de calculs statistiques, des modules de lecture et d'écriture de matrices au format Matrix Market, etc.

- http://www.scipy.org/

Biopython

Ce projet regroupe un ensemble de modules spécialisés dans la biologie moléculaire.

- http://biopython.org

Web

Pour terminer, voici une liste de frameworks de programmation web, qui proposent des fonctionnalités plus ou moins évoluées :

- Zope : http://www.zope.org
- Quixote : http://www.mems-exchange.org/software/quixote/
- CherryPy : http://www.cherrypy.org/
- Django : http://www.djangoproject.com/
- Turbogears : http://www.turbogears.org/
- ClearSilver : http://www.clearsilver.net/ (système de template)
- Python Paste : http://pythonpaste.org/

CheeseShop

Cette annexe a proposé un tour d'horizon de quelques bibliothèques tierces et peut être complétée par l'utilisation du site CheeseShop (anciennement PyPi) hébergé sur le site de Python.org, qui est au langage Python ce que CPAN est à Perl : un annuaire centralisé de bibliothèques doté d'un moteur de recherche.

> http://www.python.org/pypi

C

Sites, flux RSS, blogs et autres friandises...

Cette annexe présente une liste de liens de la planète Python, regroupés en trois catégories :

- les sites web ;
- les flux RSS ;
- les blogs (flux RSS nominatifs).

Chaque lien, qui peut être en anglais ou en français, est commenté.

Flux RSS

EN Daily Python-URL! : le flux RSS du langage Python, géré par Fredrik Lundh et, soutenu par Secret Labs AB (Pythonware).

```
http://effbot.org
```

Ce flux, mis à jour quotidiennement, est un véritable travail éditorial, mené par un core developer du langage et qui contient une sélection des meilleures nouvelles de la planète Python.

▸ http://www.pythonware.com/daily/

EN Unofficial Planet Python : l'autre flux RSS majeur. Cette deuxième source d'informations n'est pas une sélection qualitative comme Daily Python-URL mais propose un agrégateur de flux ; il reste nécessaire de faire le tri.

▸ http://www.planetpython.org/rss20.xml

EN Cheese Shop recent updates : le flux des mises à jour des bibliothèques de CheeseShop (PyPi). Garder un œil sur ce flux peut permettre de découvrir de nouveaux outils ou de surveiller certains modules.

▸ http://www.python.org/pypi?%3Aaction=rss

EN Recipes from the Python Cookbook : le flux des recettes Python saisies dans le CookBook du site ASPN. Une lecture saine et bénéfique.

▸ http://aspn.activestate.com/ASPN/Cookbook/Python/index_rss

FR Planète Python : une initiative intéressante qui tente de regrouper des flux francophones, encore trop rares.

▸ http://python.gnu-gml.net/

EN, FR Nuxeo : blog collectif des employés de Nuxeo. Membres qui sont pour la plupart impliqués dans des projets communautaires Python et Zope.

▸ http://blogs.nuxeo.com/sections/aggregators/all_posts/exportrss

Il existe évidemment beaucoup d'autres flux, mais les liens fournis ci-dessus génèrent les informations les plus intéressantes, et produisent entre 50 et 100 nouvelles par jour, ce qui est plus que suffisant.

Blogs

EN Guido van Rossum's Weblog : GvR bloggue relativement rarement, mais il est important de l'avoir dans ses marqueurs, pour pouvoir briller en société quand il y ajoute une nouvelle entrée. Il y aborde souvent des sujets relatifs aux développements en cours sur le langage.

- http://www.artima.com/weblogs/feeds/bloggers/guido.rss

EN Bruce Eckel's Weblog : Bruce Eckel a écrit des livres sur Java, C++ et bien sûr Python. Ses entrées ne concernent pas toujours Python, mais on y retrouve de brillantes analyses et des comparatifs sur les modèles objets, la programmation générique, etc.

- http://www.mindview.net/
- http://www.artima.com/weblogs/feeds/bloggers/beckel.rss

EN Zope Dispatches : blog de Paul Everitt, le fondateur de Digital Creation, à l'origine de Zope, et actuel membre de la Zope Europe Association. L'omniprésence de Paul dans la plupart des événements Zope et Python, depuis 10 ans, fait de son blog une source d'informations incontournable, même s'il est maintenant très orienté Plone.

- http://radio.weblogs.com/0116506/rss.xml

EN Agile Testing : le blog de Grig Gheorghiu, membre de l'alliance agile, qui parle quasiment exclusivement des outils de tests pour Python.

- http://agiletesting.blogspot.com/atom.xml

Sites

FR Programmation Python : le site personnel de l'auteur, qui regroupe des éléments relatifs à ce livre et des informations Python.

> http://www.programmation-python.org

EN Site officiel de Python : sans commentaires, la référence.

> http://www.python.org

FR Site de l'Association Francophone Python (AFPY) : site communautaire avec des nouvelles, des tutoriaux Python et Zope, des forums, etc.

> http://www.afpy.org

FR Python Blanc Bleu Belge (P3B) : site communautaire belge.

> http://www.p3b.org/

FR Zopera : site francophone consacré à Zope (orienté Plone).

> http://www.zopera.org

Et enfin, pour quelque chose de complètement différent, et afin de reprendre une activité intellectuelle saine après la lecture de ce livre, l'incontournable site des Monty Python :

> http://www.pythonline.com/