Python et outils associés - Installation et utilisation

par Jean-Daniel Bonjour, EPFL-ENAC-IT, © Creative Commons BY-SA
(version préliminaire du 26.8.2015)

1. Introduction

Chapitre accessible sous ce lien.

2. Programmation Python

Chapitre accessible sous ce lien.

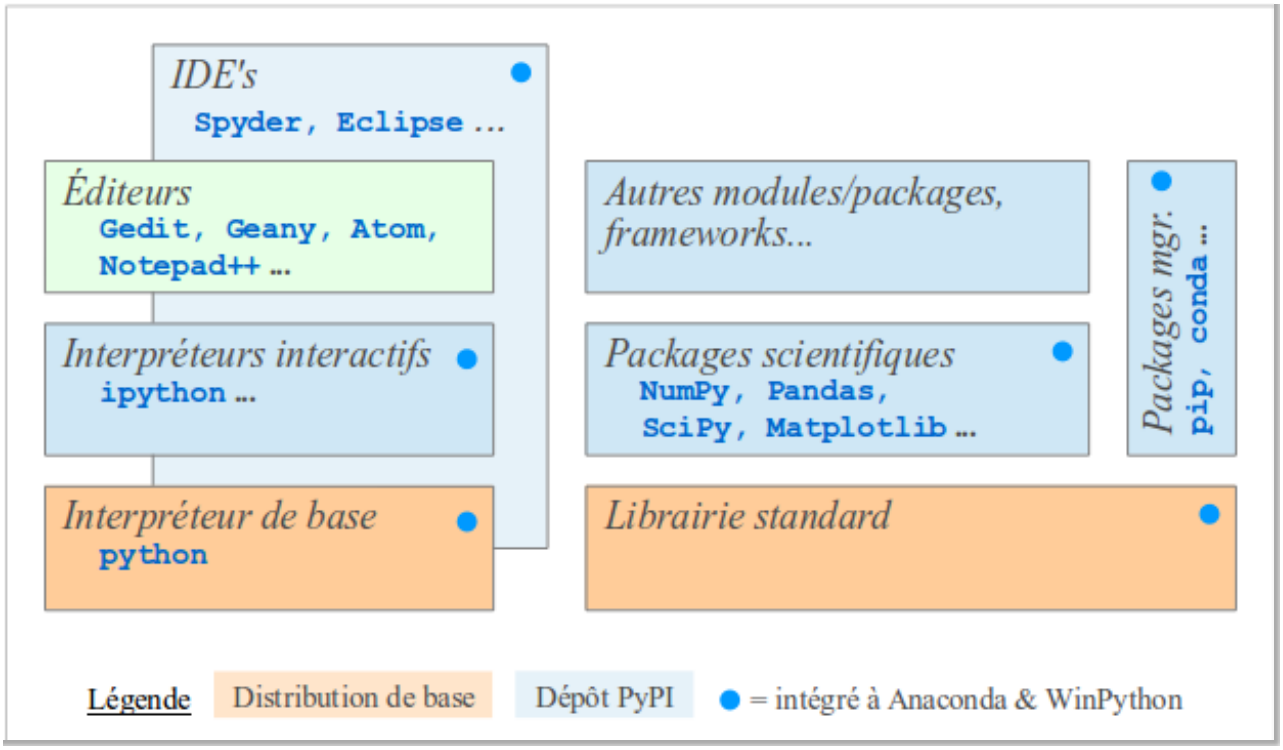
3. Scientific Python

 La rédaction de ce chapitre est en cours, merci de patienter !

4. Outils de développement Python

Un **environnement de développement** Python est constitué de différents outils et composants présentés dans la**figure** ci-dessous. Les chapitres qui suivent décrivent le rôle et l'utilisation de ceux-ci.

S'agissant de l'**installation** de Python et de ces outils sur votre machine, voyez notre chapitre 'Installation d'un environnement Python v3 complet".



4.1 Éditeurs de programmation

N'importe quel éditeur de texte permet de programmer en Python. Il est cependant utile d'en choisir un qui supporte au minimum la *coloration syntaxique* Python, ce qui est le cas de la plupart des éditeurs de programmation. Nous vous recommandons notamment (pour une liste plus complète, voir ce lien) :

- multiplateforme : **Atom** (développé par GitHub)
- sous GNU/Linux (à installer avec votre gestionnaire de packages habituel) : **Gedit**, **Geany**, Kate, etc...
- sous Windows : **Notepad++**, etc...
- sous Mac OS X : **Sublime Text** (nécessite une licence à partir de la version 4),TextWrangler, etc...

Mais cela peut être encore plus confortable de recourir à un vrai IDE (*Integrated Development Environment*), le plus utilisé actuellement pour une utilisation scientifique de Python étant Spyder (mais il y en a d'autres : voir notre chapitre sur les IDE's).

4.2 Interpréteurs Python

Il existe de nombreuses implémentations de Python... et par conséquents d'interpréteurs. Les plus connus sont :

- l'*interpréteur de base* : interpréteur par défaut intégré à toute distribution Python, écrit en C et ainsi parfois dénommé CPython
- IPython : interpréteur interactif très évolué
- bpython : version plus interactive de l'interpréteur de base (coloration syntaxique, auto-complétion...)
- IDLE : mini IDE s'appuyant sur l'interpréteur de base et le toolkit graphique tkinter
- Jython : interpréteur écrit en Java (donc tournant sur la VM Java), permettant l'utilisation d'objets Java dans du code Python
- PyPy : interpréteur écrit lui-même en Python et qui a vocation d'être très rapide (JIT compiler)
- IronPython : implémentation du langage Python dans les environnements Microsoft .NET et Mono

Il faut noter que l'interpréteur de base est l'interpréteur de référence du langage Python, et que les autres interpréteurs n'implémentent souvent pas la dernière version du langage.

Nous allons présenter ici les deux plus courants : l'interpréteur de base et l'interpréteur interactif IPython.

4.2.1 L'interpréteur de base

Lorsque Python est distribué avec le système d'exploitation (ce qui est le cas sous GNU/Linux et Mac OS X), on dispose alors de l'*interpréteur de base* et de la *librairie standard*.

L'interpréteur permet d'**exécuter des *scripts*/programmes** en frappant, depuis une fenêtre terminal : `python script.py` (ou `python3 script.py`, selon votre installation, pour forcer l'utilisation de l'interpréteur Python v3). De nombreuses options de lancement sont en outre disponibles (voir `python --help` ou `man python`). L'option `-i` est particulièrement intéressante : en frappant `python -i script.py`, juste après l'exécution du *script* on entre dans le mode mode interactif de l'interpréteur, ce qui donne la possibilité d'examiner les variables globales où tracer le stack d'erreurs.

L'interpréteur de base peut donc aussi être utilisé en **mode interactif**, et c'est ce qui se passe lorsqu'on lance la commande `python` sans passer de *script* en argument. Le mode interactif est alors signalé par le prompt `>>>`. On quittera l'interpréteur en passant la

commande `quit()` ou `exit()` ; une autre alternative consiste à frapper `<ctrl-D>` sous Linux et Mac OS X, ou `<ctrl-Z>` sous Windows (envoyant un `EOF` = fin de fichier).

Le mode interactif donne accès à une une aide en ligne :

- `help("topics")` : énumère les différents "concepts" Python au sujet desquels on peut demander de l'aide ; frapper ensuite `help("SUJET")` pour avoir de l'aide sur le *SUJET* souhaité
- `help(objet)` : affiche des informations sur l'*objet* Python spécifié (variable, fonction...) tel que son type, ses méthodes...
- `help("module")` : affiche l'aide relative au *module* spécifié ; si le module est importé on peut directement faire `help(module)` (sans guillements)
- `help("module.objet")` : affiche directement l'aide relative à l'*objet* spécifié du *module*
- `help()` : entre dans l'aide en mode interactif

Pour exécuter un *script* lorsqu'on est dans l'interpréteur interactif, on passera la commande : `exec(open('script.py').read())` (hum... pas très convivial !).

Il est possible de se définir un prologue Python, c'est à dire un fichier de code Python qui sera exécuté lors de chaque démarrage interactif de Python (p.ex. pour charger des modules...). On le nomme en général `.pythonrc` et on le place dans le répertoire personnel de l'utilisateur. L'activation de ce prologue s'effectue par la définition de la variable d'environnement suivante dans son prologue shell (`~/.bashrc`) : `export PYTHONSTARTUP=~/.pythonrc`

Mais pour un usage interactif fréquent, les possibilités de l'interpréteur de base sont très limitées, et on a bien meilleur temps d'utiliser l'interpréteur IPython présenté ci-après.

4.2.2 IPython (interpréteur interactif)

IPython est l'interpréteur Python interactif le plus réputé en raison de ses vastes possibilités interactives et graphiques ainsi que ses fonctionnalités de *notebook*.

Pour démarrer IPython, passez la commande : `ipython` (ou `ipython3`, selon votre installation, pour utiliser l'interpréteur Python v3). Voici en outre les options de lancement les plus importantes (pour davantage de détails frappez : `ipython --help` ou `ipython --help-all`) :

- `qtconsole` : lancement de IPython dans une **fenêtre graphique** Qt (ce qui nécessite les packages PyQt et PySide)
- `notebook` : lancement de IPython en mode **Notebook** (voir chapitre suivant)
- `--pylab=inline` : active le mode **pylab** (i.e. chargement des librairies Numpy et Matplotlib en mode interactif, identique à la *magic fonction* `%pylab inline`)

Pour sortir de IPython, passez la commande `exit` ou `quit`, ou frappez `<ctrl-D>` et confirmez.

La rédaction de la suite de ce chapitre (*magic fonctions*...) est en cours, merci de patienter !

4.2.3 Notebooks IPython

IPython permet aussi de travailler en ***mode notebook*** à la façon d'autres logiciels scientifiques (Mathematica, Maple...), c'est-à-dire de créer des documents interactifs composés de code Python, de texte formaté (Markdown, HTML et LaTeX) et de graphiques. La plupart des fonctionnalités IPython décrites au chapitre précédent sont utilisables dans les *cellules* de code.

L'interface notebook est **basée web**, c'est-à-dire qu'au démarrage de IPython en mode Notebook un serveur web est automatiquement démarré localement sur votre machine, et vous interagissez avec le *notebook* depuis votre navigateur web.

Pour lancer IPython en mode Notebook, passez la commande `ipython notebook` (ou `ipython3 notebook`, selon votre installation, pour utiliser l'interpréteur Python v3). Il est pratique de passer cette commande depuis le répertoire où se trouvent vos fichiers notebooks afin que IPython les affiche directement et que les sauvegardes s'effectuent à cet endroit. Vous pouvez sinon désigner l'emplacement de vos notebooks en complétant la commande avec le paramètre `--notebook-dir=chemin`

Les liens ci-dessous vous permettent de télécharger 2 notebooks que nous avons élaborés. Sauvegardez-les sur votre machine avec `<clic-droite> Save link as` :

- ipython-notebook.ipynb : notebook d'introduction à l'utilisation de **IPython Notebook**
- python-bases.ipynb : notebook d'introduction rapide aux **bases du langage** Python

La rédaction de la suite de ce chapitre est en cours, merci de patienter !

4.3 Spyder (IDE)

Spyder (Scientific PYthon Development EnviRonment) est un IDE orienté vers un usage scientifique de Python et doté de fonctionnalités avancées d'édition, debugging, introspection et profiling. L'utilisateur MATLAB ou GNU Octave GUI se retrouvera dans un environnement familier, avec notamment des fenêtres Console, Editor, Variable explorer, File explorer, History, Online help... Disponible sur tous les systèmes d'exploitation, cet IDE est lui-même écrit en Python et s'appuie sur le toolkit graphique multiplateforme Qt (nécessitant les packages PyQt et PySide).

Cet IDE s'installe sous GNU/Linux de façon standard (voir ce chapitre), et est notamment intégré aux *bundles* WinPython et Anaconda présentés plus loin. Notez que Spyder ≤ 2.2.x est orienté Python v2, et Spyder ≥ 2.3.x est nécessaire pour Python v3.

La rédaction de la suite de ce chapitre est en cours, merci de patienter !

5. Installation d'un environnement Python v3 complet

Python n'est pas installé par défaut sous Windows (où dominent les environnements de programmation propriétaires de Microsoft: PowerShell, .NET). Sous GNU/Linux et Mac OS X par contre, l'*interpréteur* de base Python ainsi que la *librairie standard* sont intégrés au système :

- depuis GNU/Linux Ubuntu 14.04 on dispose en parallèle de Pyhon v2 et v3
- sous Mac OS X 10.8 (*Mountain Lion*, 2012), 10.9 (*Mavericks*, 2013) et 10.10 (*Yosemite*, 2014), Apple est très conservateur et ne propose que Python 2.7

Ce support de cours étant axé sur Python v3, les chapitres qui suivent ont pour objectif de vous aider à installer sur votre propre machine, selon votre système d'exploitation et de façon non intrusive (i.e. dans un contexte utilisateur), l'environnement de développement **Scientific Python v3** de base (packages Numpy, Scipy et Matplotlib notamment, ainsi que IPython et Spyder). Nous avons pour cela opté en faveur de :

- pour Windows : **WinPython** : distribution spécifique à Windows et très facile à mettre en oeuvre
- pour GNU/Linux Ubuntu ≥ 14.04 : installation standard via les **dépôts officiels Ubuntu** de Canonical
- pour Mac OS X : **Anaconda** : distribution Python multiplateforme très répandue dans les milieux scientifiques

5.1 WinPython : bundle Python recommandé sous Windows

Il serait possible d'installer l'interpréteur Python v3 de base (et sa librairie standard) sous Windows en utilisant l'installeur proposé par le site Python, mais l'ajout des autres outils n'est alors pas évidente. Nous nous orientons donc plutôt vers un bundle complet.

WinPython est un *bundle* Scientific Python assez récent qui est entièrement libre mais spécifique à Windows. Il intègre notamment : IPython, Spyder, NumPy, SciPy, MatplotLib, Pandas, SymPy, PIP... (voir la liste complète des packages). Extrêmement simple à installer, il présente en outre l'intérêt d'être entièrement portable (donc par exemple installable sur un média USB amovible). Une autre alternative sous Windows serait d'installer le *bundle* Anaconda (méthode présentée plus bas pour Mac OS X).

Suivez simplement la procédure d'installation qui suit :

1. Allez sur le site http://sourceforge.net/projects/winpython/files/
2. Descendez dans l'arborescence WinPython 3.4 et téléchargez l'installeur `WinPython-plateforme-3.4.x.y.exe` *(env. 280 MB)* ;
   choisissez, entre les versions `-64bit-` et `-32bit-`, celle qui correspond à votre système d'exploitation (en principe 64bit sur les machines récentes)
3. Lancez cet installeur, et choisissez l'emplacement d'installation à votre convenance

Une fois l'installation terminée, il reste à faire un tout petit peu de configuration/personnalisation :

4. Allez à la racine du dossier d'installation *(où env. 800 MB ont été déposés)*, et créez sur le bureau des raccourcis de lancement pour les applications suivantes : "IPython Qt Console", "IPython Notebook", "WinPython Command Prompt" et "Spyder"
5. S'agissant du raccourci "IPython Notebook", éditez ses Propriétés (en cliquant avec le bouton de <droite> de la souris) : dans son onglet "Raccourci", dans le champ Cible, ajoutez après `"...\IPython Notebook.exe"` un caractère <espace> puis l'option `--notebook-dir=chemin` indiquant le *chemin* du répertoire dans lequel vous stockez vos notebooks
6. Si vous désirez finalement associer au niveau de l'Explorateur Windows les extensions `.py`, `.pyc` et `.pyo` à cette distribution, ouvrez le "WinPython Control Panel" et faites Advanced>Register distribution.

Pour ajouter ou supprimer des packages ultérieurement, cela pourra se faire avec le gestionnaire WPPM (WinPython Package Manager, accessible depuis le "WinPython Control Panel"), ou avec le gestionnaire PIP qui est également intégré à cette distribution.

Outre l'éditeur intégré à Spyder, si vous souhaitez installer d'autres éditeurs adaptés à l'édition de code Python, voyez notre chapitre "Éditeurs de programmation".

5.2 Scientific Python v3 : installation standard sous GNU/Linux Ubuntu

La procédure standard décrite ici fonctionne complètement à partir de Ubuntu 14.04 LTS. Pour d'autres distributions GNU/Linux, référez-vous aux indications de l'éditeur. Quelle que soit la distribution, une bonne alternative consiste à installer le *bundle* Anaconda (méthode décrite plus bas pour Mac OS X).

Les outils et librairies dont nous avons besoin étant, depuis Ubuntu 14.04, entièrement portés sous Python v3 et *packagés* dans les dépôts standards, l'installation s'effectue de façon classique (les indications ci-après se rapportant précisément à Ubuntu 14.04) :

1. S'agissant de l'interpréteur Python v3 et sa *librairie standard*, rien à faire de spécial, car Ubuntu 14.04 embarque déjà Python 3.4 (accessible sous la commande `python3`) en parallèle à Python 2.7 (accessible avec `python`)
2. `sudo apt install python3-pip python3-dev` : installe PIP pour Python v3, qu'on utilisera avec la commande `pip3` (la commande `pip`, quant à elle, agissant sur l'environnement Python v2).
3. `sudo pip3 install ipython[all]` : installe IPython pour Python v3 (intégrant les fonctionnalités *notebook* et *qtconsole*), qu'on démarrera indifféremment avec la commande `ipython3` ou `ipython`
   (Une installation standard via les dépôts Ubuntu, avec "sudo apt install ipython3 ipython3-notebook ipython3-qtconsole", installerait une ancienne version IPython 1.2.x au lieu de la dernière en date 2.x)
4. `sudo apt install python3-numpy python3-scipy python3-matplotlib python3-pandas` : installe les librairies Scientific Python v3 de base
5. `sudo apt install spyder3 python3-zmq` : installe l'IDE Spyder 2.3, qu'on lancera avec la commande `spyder3`
   (Une autre manière de l'installer, pour disposer de la version la plus récente, serait de le faire avec PIP)

Notez que si vous souhaitez faire du virtualenv Python, Ubuntu 14.04 offre, sans rien installer, la commande `pyvenv-3.4`

Outre l'éditeur intégré à Spyder, vous pouvez installer l'un ou l'autre des éditeurs de programmation classiques suivants :

- Gedit : `sudo apt install gedit gedit-plugins`
- Geany : `sudo apt install geany geany-plugins`
- Atom : définition du dépôt alternatif: `sudo add-apt-repository ppa:webupd8team/atom` puis installation: `sudo apt update && sudo apt install atom`

5.3 Anaconda : bundle Python recommandé sous Mac OS X (voire autres OS)

Il serait possible d'installer l'interpréteur Python v3 de base (et sa librairie standard) sous Mac OS X en utilisant l'installeur proposé par le site Python, mais l'ajout des autres outils n'est alors pas évidente. Nous nous orientons donc plutôt vers un bundle complet.

Anaconda est un *bundle* Python actuellement très en vogue dans le monde scientifique. Développé par la société Continuum Analytics, non "libre" mais gratuit dans sa version de base, il a l'avantage d'être multiplateforme (Windows, Mac OS X et GNU/Linux) et d'intégrer une grande quantité d'outils et packages Python, notamment : IPython, Spyder, NumPy, SciPy, MatplotLib, Pandas, Sympy, PIP... (voir la liste complète).

5.3.1 Installation de base

Nous allons ici installer Anaconda 3 afin de disposer d'un environnement Scientific Python v3. La procédure, fonctionnant pour Mac OS X 10.7 à 10.10, est la suivante :

1. Allez sur la page http://continuum.io/downloads
2. Votre système d'exploitation (Mac OS X) devrait être automatiquement détecté, mais on nous propose par défaut l'environnement Anaconda pour Python v2. Cliquez donc plutôt sur "**I WANT PYTHON 3.x**"
3. Deux possibilités sont offertes : un installeur en mode graphique ou un installeur en ligne de commande Choisissez le premier en cliquant sur le bouton **[Mac OS X - 64-Bit Python 3.x Graphical Installer]**
4. Le fichier `Anaconda3-version-MacOSX-x86_64.pkg` (env. 280 MB) est téléchargé (notez le `3` qui correspond à Python v3)
5. Lancez l'installation proprement dite en double-cliquant sur ce fichier-package. Accepter la licence ainsi que l'emplacement d'installation par défaut (qui est `/Users/username/anaconda`)
6. Une fois l'installation terminée, vous pouvez jeter ce fichier-package

Notez enfin ce qui suit :

- Depuis une fenêtre terminal, vous pouvez maintenant (votre PATH ayant été automatiquement mis à jour) :
  - utiliser l'interpréteur Python v3 avec la commande `python`
  - lancer l'interpréteur interactif IPython avec la commande `ipython` (et les paramètres `qtconsole` pour avoir une fenêtre dédiée, ou `notebook` pour travailler sur des *notebooks* dans votre navigateur web)
  - démarrer l'IDE Spyder avec la commande `spyder &`
- Anaconda dispose de son propre *gestionnaire de package* que l'on invoque avec la commande `conda`. Vous pouvez notamment :
  - mettre à jour Python et le gestionnaire de packages Conda avec : `conda update conda`
  - mettre à jour tous les packages Anaconda avec : `conda update anaconda` (dans cette commande, `anaconda` représente un *méta-package* rassemblant l'ensemble des packages proposés par Anaconda)
  - installation d'un `package` Anaconda spécifique avec : `conda install package`
- Cet environnement Anaconda Python v3 est installé de façon totalement stand-alone dans l'espace-fichier de l'utilisateur. Si vous souhaitez le désinstaller, il suffit de jeter le dossier `/Users/username/anaconda` (qui pèse env. 900 MB)

Outre l'éditeur intégré à Spyder, si vous souhaitez installer d'autres éditeurs adaptés à l'édition de code Python, voyez notre chapitre "Éditeurs de programmation".

5.3.2 Définition d'environnements de développement Anaconda/Python (pour les curieux seulement)

De façon analogue à la technique *virtualenv* Python, il est possible de définir des environnements de développement Anaconda/Python distincts avec leurs propres versions Python, leurs propres modules/packages...

Voyons tout d'abord quelques commandes de base du package manager `conda`. Les parenthèses autour du paramètre `(-n env)` indiquent que celui-ci est facultatif, nécessaire uniquement pour cibler un environnement `env` Python alternatif :

- `conda create -n env packages` : crée un nouvel environnement Python nommé *env* et y installe les *packages* spécifiés
- `conda install (-n env) packages` : installe les *packages* spécifiés dans l'environnement par défaut ou dans l'environnement *env*
- `conda list (-n env)` : affiche la liste des packages installés et leur version courante
- `conda update (-n env) packages` : met à jour les *packages* spécifiés
- `conda remove (-n env) packages` : supprime les *packages* spécifiés
- `conda search query` : affiche tous les packages (installés ou disponibles) correspondant à *query*

Pour créer par exemple un *environnement* Python 2.7 nommé `py2` et contenant Spyder, on ferait ceci :

1. `conda create -n py2 python=2` : création de cet environnement contenant Python 2.7.x ainsi que les packages dépendants (openssl, readline, sqlite, system, tk, zlib)
2. `conda install -n py2 spyder` : installe Spyder dans cet environnement ainsi que les packages dépendants ipython, libpng, pygments, pyside, pyzmq, qt, shiboken, util-linux, zeromq (et encore python.app sous Mac OS X)

Pour utiliser cet environnement, on procéderait alors ainsi :

- `source activate py2` : bascule dans cet environnement (modification du PATH du shell courant)
- si l'on lance `python`, on est alors dans Python 2.7 ; on peut aussi lancer `ipython qtconsole` ; démarrons finalement `spyder` : si l'on va sous le menu `? > Optional Dependencies`, on constate qu'il faudrait idéalement encore installer (à l'aide de `conda`) les packages suivants pour que Spyder soit pleinement utilisable : matplotlib, pep8, pyflakes, rope, sphinx et évent. sympy.
- `source deactivate` : revient dans l'environnement de base (rétablissement du PATH du shell courant)
- si l'on lance `python`, on est bien de retour dans Python v3 !

Remarque technique : ce que fait `source activate py2`, c'est simplement ajouter, pour le shell courant, le chemin `...anaconda/envs/py2/bin` en tête de la variable d'environnement PATH. Inversement, `source deactivate` enlève ce chemin. Cela signifie que vous pouvez, pour vous simplifier la vie, lancer directement les outils Python de cet environnement `py2` (Python, IPyton, Spyder...) sans faire d'`activate` en faisant précéder le nom du chemin `...anaconda/envs/py2/bin` ; ainsi que créer en conséquence, dans votre prologue de shell (sous Ubuntu le fichier `~/.bashrc`, et sous Mac OS X `~/.bash_profile`), les alias de commande correspondants !

5.4 Gestion d'environnements Python multiples avec virtualenv

Particulièrement utile pour les développeurs, `virtualenv` permet d'installer, dans un contexte utilisateur, plusieurs environnements de développement Python parallèles (avec différentes versions de Python et/ou de modules/packages) et de passer aisément de l'un à l'autre.

 Rédaction de ce chapitre en cours, merci de patienter !

5.5 Packages et modules additionnels

Bien que la *librairie standard* soit déjà extrêmement riche, Python est aussi réputé pour le très grand nombre de modules et packages additionnels existant et librement accessibles, notamment à travers le dépôt **PyPI** (*Python Package Index*).

Il existe plusieurs façons d'installer des modules/packages supplémentaires, selon la procédure/distribution qui a été utilisée pour installer Python :

- Lorsque Python est lié au système d'exploitation (p.ex. versions de Python intégrées à GNU/Linux et Mac OS X), c'est l'OS qui prend en charge ses mises à jour de même que l'installation des packages additionnels les plus courants. Sous Ubuntu par exemple, les modules les plus connus (provenant de la distribution Debian) sont *packagés* pour Python v3 dans les dépôts Ubuntu sous le nom `python3-package` (respectivement `python-package` pour Python v2) ; on les installe ainsi avec le gestionnaire de packages standard (commande `sudo apt install python3-package`).
- Lorsque Python est installé sous forme d'une distribution complète (*bundle*), celle-ci dispose de ses propres dépôts et propose en général son propre gestionnaire de paquets, par exemple `conda` sous Anaconda, WPPM sous WinPython...
- Et l'on peut finalement utiliser le gestionnaire de packages **PIP** présenté ci-après.

5.5.1 Le gestionnaire de packages PIP

PIP (Python Install Python) est le gestionnaire de packages Python le plus répandu actuellement (supplantant l'ancien mécanisme nommé *easy_install*). Il permet d'installer en ligne de commande les packages provenant du dépôt **PyPI** (*Python Package Index*). On utilise cette méthode pour les packages non offerts par les distributions *bundle* ou dépôts Linux, ou lorsque l'on s'intéresse aux dernières versions de ces packages.

PIP est aussi intégré aux *bundles* Anaconda et WinPython. Pour disposer de PIP sous Linux/Ubuntu, on l'installe pour Python v3 par la commande : `sudo apt install python3-pip python3-dev`

Voici les commandes PIP principales (`pip3` pour Python v3, `pip` pour Python v2) :

- `pip3 help` : affiche l'aide sur la commande `pip`
- `pip3 freeze` : affiche la liste des packages Python installés
- `pip3 install package` : installe depuis PyPI le *package* spécifié avec ses éventuelles dépendances
- `pip3 install package==version` : force l'installation *package* à la *version* spécifiée
- `pip3 install --upgrade package` : met à jour le *package* installé
- `pip3 install --index-url http://repository package` : installe un *package* depuis un *repository* alternatif
- `pip3 uninstall package` : désinstalle un *package*
- `pip3 show package` : affiche les infos relatives au *package* installé
- `pip3 search query` : recherche, dans le dépôt PyPI, tous les packages dont le nom ou la description contient *query*

Notez que les commandes `pip` et `pip3` relatives à l'installation ou à la suppression de packages nécessitent des privilèges d'administration si ceux-ci sont installés au niveau système, donc ces commandes doivent être passées sous Ubuntu avec `sudo pip3 ...`. Mais cela n'est pas le cas s'ils sont installés dans le contexte utilisateur (avec *virtualenv*).

5.6 Autres distributions ou outils Python

L'inventaire ci-dessous n'est pas exhaustif.

Distributions

Enthought Canopy (ex- Enthought Python Distribution) est un *bundle* Scientific Python multiplateforme. Commercial, il est cependant gratuit pour un usage académique ou dans la version de base Canopy Express. Il n'inclut pas l'IDE Spyder.

Active Python (de la société ActiveState Software Inc.) est une distribution Python multiplateforme importante, mais moins orientée vers un usage scientifique. Proposée en deux versions : Community (gratuite) et Business (payante).

Python(x,y) est un *bundle* Scientific Python libre pour Windows. Plus ancien que WinPython, il tourne sous Windows 64bit mais en mode 32bit. Basé sur l'IDE Eclipse.

Portable Python est un *bundle* Scientific Python libre pour Windows. Portable comme WinPython, il s'appuie sur l'IDE PyScripter.

Cygwin, l'environnement libre bien connu d'émulation Unix pour Windows, permet l'installation et utilisation de Python. Mais cette solution n'a de sens que pour ceux utilisent/maîtrisent déjà Cygwin.

Sous Mac OS X il est aussi possible d'installer Python via l'un des systèmes de packaging **Fink**, **Homebrew** ou **MacPorts**. Mais c'est relativement lourd (nécessite l'environnement de développement Apple XCode) et pas évident lorsque l'on veut intégrer certains outils Python annexes (Qt, IDE, etc...)

IDE's

Parmi les IDE's bien adaptés à Python, outre Spyder présenté plus haut, mentionnons (voir ce lien pour une liste plus complète) :

Eclipse, le célébrissime IDE libre et multiplateforme, se prête bien entendu aussi au développement Python, complété par l'extension **PyDev**.

PyScripter est un IDE Python libre pour Windows qui semble intéressant.