Python : Outils pour le packaging et le déploiement

Pierre Navaro

navaro@math.unistra.fr

Institut de Recherche Mathématique Avancée, Strasbourg

Journée DevelopR6, 14 juin 2012

`http://www-irma.u-strasbg.fr/~navaro/besancon.pdf`

Pourquoi un package ?

- Simplicité.
- Stabilité.
- Distribution.
- Gestion des dépendances.

Les outils de packaging disponibles sous Python

- Très nombreux (trop ?)
- Facile d’utilisation (pas toujours).

Le module Python

- Soit `MyModule.py` un fichier Python contenant des fonctions et des classes.
- Ces fonctions et ces classes sont accessibles avec :

```
import MyModule
from MyModule import f1,f2,MyClass1
from MyModule import *
```

- Plusieurs fichiers python peuvent être regroupés et constituent un «package».
- Le nom d'un module est lié au nom du fichier, le nom du package sera lié au nom du répertoire
- Python reconnait un package à la présence dans le répertoire du fichier : `__init__.py`
- Ce fichier peut être vide et doit exister dans chacun des sous répertoires.

L’emplacement standard d’installation

- Défini par la variable PYTHONPATH

```
$ python -c "import sys; print sys.path"
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-linux2',
'/usr/lib/python2.7/lib-tk', '/usr/lib/python2.7/lib-old',
'/usr/lib/python2.7/lib-dynload',
 '/usr/lib/python2.7/dist-packages', ...]
```

- Pour installer ses modules dans le répertoire «officiel» :

```
#!/usr/bin/env python
import sys, shutil
ver = sys.version[0:3] # version of Python
libdir = os.path.join(sys.prefix, 'lib', 'python'+ver, 'site-packages')
module_file = sys.argv[1]
shutil.copy(module_file, libdir)
```

NE PAS UTILISER CETTE METHODE !

Un code python qui dit Bonjour !

- hello.py

```
def helloworld():
    print 'Hello World'
if __name__ == "__main__":
    import hello
    hello.helloworld()
```

- __init__.py

```
from hello import helloworld
```

```
hello-1.0/
    PKG-INFO
    MANIFEST.in
    setup.py
    hello/
        __init__.py
        hello.py
    doc/
        index.rst
        html/
            index.html
```

Le fichier MANIFEST.in

Sert à spécifier les fichiers non-exécutables à inclure dans la distribution.

```
include *.py

include hello/*.py
include src/wrapper/*.cpp

recursive-include doc/*.rst *.html
include doc/Makefile

include setup.py
include README.txt
```

Choisir son site d'hébergement

Site personnel, labo ou :

- Sourceforge
- Google Code
- GitHub

Choisir son gestionnaire de versions

- Bazaar (bzr) : `http://bazaar-vcs.org/`
- Git (git) : `http://git-scm.com/`
- Mercurial (hg) : `http://mercurial.selenic.com/wiki/`
- Subversion (svn) : `http://subversion.tigris.org/`

Formats disponibles pour la distribution

- archive tar (unix)
- fichier zip (windows,unix)
- binaire egg (universel)
- rpm (redhat, fedora)
- deb (ubuntu, debian)
- pkg (solaris, macos)

Un outil de compilation : distutils

- Distutils a été initié par Greg Ward en 1998.
- Organise un project Python autour d'un fichier `setup.py`.
- Distutils permit de configurer, compiler, deployer, ...
- Possède des extensions pour l'interface avec C, Fortran.
- Ajouté à la librairie standard de Python 1.6 en 2000.

```
from distutils.core import setup

setup(name='Distutils',
      version='1.0',
      description='Python Distribution Utilities',
      author='Greg Ward',
      author_email='gward@python.net',
      url='http://www.python.org/sigs/distutils-sig/',
      packages=['distutils', 'distutils.command'],     )
```

Notre fichier setup.py

```
from setuptools import setup

setup(name='hello',
      version='1.0',
      distribution= "Bonjour!",
      author = "Pierre Navaro",
      author_email = "navaro@math.unistra.fr",
      packages=['hello',],
      url="http://www-irma.u-strasbg.fr/~navaro/hello",
      packages = ['hello'],
      scripts  = ['hello_script']      )
```

```
$ python setup.py --help-commands
  install            install everything from build directory
  bdist              create a built (binary) distribution
  bdist_rpm          create an RPM distribution
  bdist_wininst      create an executable installer for MS Windows
  bdist_egg          create an "egg" distribution
  build_sphinx       Build Sphinx documentation
```

Packaging et dépôt sur PyPi (Python Package Index)

Depuis 2003 c'est un index central accessible avec Distutils, creé par Richard Jones et connu aussi sous le nom «Cheeseshop».

- Créer un compte sur `http://pypi.python.org/pypi`

```
$ python setup.py register
```

- Création du fichier .pypirc :

```
[pypirc]
servers = pypi
[server-login]
username:your_awesome_username
password:your_awesome_password
```

- Déposer sa distribution "source" :

```
$ python setup.py sdist upload
```

Installation du package avec Distutils

- Télécharger et déployer l'archive puis :

```
$ sudo python setup.py install
```

- Cette technique permet de déployer le logiciel dans le répertoire site-packages situé dans le répertoire d'installation de Python.
- Attention cette commande n'installe pas les dépendances.
- Effets de bords possibles avec votre système.
- Pas de commande pour désinstaller.

Setuptools

- Initié par Phillip Eby en 2005, c'est une extension des Distutils.
- Permet de créer des packages au format binaires (eggs).
- Un outil d'installation automatique (`easy_install`).
- dépendances optionnelles
- Génère automatiquement le fichier MANIFEST.in avec l'aide du gestionnaire de versions,
- Capable de fouiller sur le web pour trouver des packages,
- gère correctement les numérotations complexes de versions.

`http://pypi.python.org/pypi/setuptools`

easy_install

- Usage classique :

  easy_install SQLObject

- Installation à partir d'un lien donné via une URL :

  easy_install -f http://pythonpaste.org/package_index.html SQLObje

- Télécharge, compile et installe automatiquement :

  easy_install http://example.com/path/to/MyPackage-1.2.3.tgz

- Installe le package le format binaire egg déjà téléchargé :

  easy_install /my_downloads/OtherPackage-3.2.1-py2.3.egg

- Mise à jour d'un package :

  easy_install --upgrade PyProtocols

- Installation à partir du répertoire source du package :

  easy_install .

Distribute

- En 2008, Setuptools est une partie vitale de Python mais son développement ralentit.
- Cet autre projet voit le jour et supporte Python 3 (Ian Bicking)
- Setuptools et Distribute sont deux packages distincts et concurrents.
- Il est fortement conseillé de choisir entre ces deux outils (pas de cohabitation).

`http://packages.python.org/distribute/index.html`

pip (package distribute)

```
$ pip install hello
```

- Tous les paquets sont téléchargés au préalable.
- Affiche des messages plus clairs.
- Dans le cas d'une dépendance celle-ci est tracée par l'installation.
- Meilleur code.
- Les packages ne sont pas forcément installés au format binaire egg.
- Support natif pour les systèmes de gestion de version (Git, Mercurial ou Bazaar)
- Possibilité de desinstaller.

Un nouveau package `distutils2` est en développement...

Les problèmes de l’installation globale

- Problème de volume.
- Nécessite de travailler avec les droits super utilisateur.
- Il n’est pas bon de mettre tous les modules dans le même répertoire.
- Certaines dépendances exigent des numéros de versions différentes du même module.

D’autres solutions sont apparues en 2006.

- Buildout (2006)
- Virtualenv (2007)

Virtualenv

Cet outil permet de cloner en environnement Python pour y installer vos propres modules sans effets de bord sur votre système.

```
$ sudo pip install virtualenv
$ virtualenv ENV
```

- Un répertoire ENV/lib/pythonX.X/site-packages est créé
- Un interpréteur Python également dans ENV/bin/python
- Il installe également les setuptools ou distribute.

`http://pypi.python.org/pypi/virtualenv`

Buildout

- Outil de construction haut niveau (Jim Fulton 2006)
- construit sur Setuptools et easy_install.
- Permet d’encapsuler un groupe de packages comme virtualenv
- Plus ambitieux, il permet de construire un environnement de dépendances complet pour des projets complexes.
- Toutes les dépendances sont installées automatiquement.
- Toutes les installations sont possibles même celles qui ne sont pas écrites en Python.
- On peut par exemple générer une documentation ou installer MySQL.

buildout.cfg

```
[buildout]    #indispensable
parts = test #La compilation est décomposée en parties (peut être vide)
develop = . #Pour définir le répertoire ou seront créés les fichier eggs.
[test]
recipe = zc.recipe.testrunner
eggs = xanalogica.tumbler
```

Malheureusement la documentation manque de clarté.

http://www.buildout.org/

Références

A guide to Python packaging

`http://www.ibm.com/developerworks/opensource/library/os-pythonpackaging/`

The Hitchhiker's Guide to Packaging

`http://guide.python-distribute.org/`

Python documentation for distribution

`http://docs.python/distutils/`

PyPI project example

`http://pypi.python.org/pypi/an_example_pypi_project`

Concept de Python eggs

`http://cours-plone-niveau-1.ecreall.com/integrateur/concepts_python_eggs.html`

Outils de compilation Python : SCons

Fichier de description SConstruct

```
Program('program', ['main.c', 'hello.c'])
```

```
$ scons
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
gcc -o main.o -c main.c
gcc -o hello.o -c hello.c
gcc -o hello main.o hello.o
scons: done building targets.
```

- Simple à utiliser.
- Très bien documenté.
- Limité et trop lent parfois.

SCons

Pour

- Syntaxe Python.
- Fonctionne sur tous les OS.
- Possibilité d'auto-configuration.
- Le langage Python est sa seule dépendance.
- Pas d'étape intermédiaire.

Contre

- Vous devez connaitre un minimum de Python et Python doit être installé.
- Un peu limité.
- Code confus pour une utilisation bas niveau.
- Lent, surtout lorsque les répertoires contenant les sources sont nombreux.

Outils de compilation Python : waf

Fichier de description : wscript

```
def configure(conf):
    conf.load('compiler_c')
def build(bld):
    bld.program(source =['main.c', 'hello.c'], target = 'hello',
                        features = 'c cprogram')
```

```
$ ./waf configure
Setting top to                           : /Users/navaro/DevelopR6
Setting out to                           : /Users/navaro/DevelopR6/build
Add options(opt): opt.load('compiler_c')
(complete log in /Users/navaro/DevelopR6/build/config.log)
$ ./waf
Waf: Entering directory '/Users/navaro/DevelopR6/build'
[1/3] c: main.c -> build/main.c.1.o
[2/3] c: hello.c -> build/hello.c.1.o
[3/3] cprogram: build/main.c.1.o build/hello.c.1.o -> build/hello-wor
Waf: Leaving directory '/Users/navaro/DevelopR6/build'
'build' finished successfully (0.106s)
```

Waf

Pour

- Plus rapide que SCons.
- La configuration est separée de la compilation.
- Les sorties à l'écran sont configurables (couleur, format,...)
- Plus proche du code Python.
- Outil de compilation de KDE.

Contre

- Pas encore aussi répandu que SCons.
- Moins populaire donc moins d'utilisateurs.
- Produit jeune.
- Ne supporte pas les anciennes versions de Python

Outils de compilation Python : fbuild

Fichier de description fbuildroot.py (Python 3 !)

```
import fbuild.builders.c

def build(ctx):
    builder = fbuild.builders.c.guess_static(ctx)
    exe = builder.build_exe('helloworld', ['main.c','hello.c'])
```

```
$ fbuild
determining platform     : {'bsd', 'darwin', 'macosx', 'posix'}
looking for gcc          : ok /usr/local/bin/gcc
checking gcc             : ok
looking for ar           : ok /usr/bin/ar
looking for ranlib       : ok /usr/bin/ranlib
checking if gcc can make objects : ok
checking if gcc can make libraries : ok
checking if gcc can make exes      : ok
checking if gcc can link lib to exe : ok
 * gcc                             : main.c -> build/main.o
 * gcc                             : hello.c -> build/hello.o
 * gcc                             : build/hello.o build/main.o -> b
```

Python, un bon choix !

- Progression importante dans le mileu académique.
- Langage sans limite.
- Propose des outils pour le web, la compilation, la doc, l'IHM...
- S'interface extrêmement facilement.
- Propose des outils de compilation pour tous les langages.
- Cet outil de compilation peut être fourni avec le logiciel.
- SAGEMATH (`http://www.sagemath.org`) est un bel exemple des possibilités de Python.

Références

Make alternatives

`http://freecode.com/articles/make-alternatives`

Benchmarks of various C++ build tools

`http://psycle.svn.sourceforge.net/viewvc/psycle/branches/bohan/wonderbuild/benchmarks/time.xml`