Formation Python : Installation, packaging, env. virtuel

par Bernard CHAMBON

Subatech, Nantes - France

23, 24 janvier 2018

- Actions et outils
  - Installation (module ou distribution) : `easy_install`, `pip`
  - Format de packaging
  - Packaging avec `setup.py`
  - Environnement virtuel : `virtualenv`
- Mise en pratique

- Outil d'installation : easy_install et pip
  - `easy_install`, de 2004
    Outil qui permet d'installer des packages
    Peut fonctionner avec `PyPI` (Python Package Index), repository de package python
    Outil ancien (2004), qui ne reconnait que le format `eggs`
  - `pip`, de 2008 (`pip3` avec Python3)
    C'est l'outil d'installation de package
    pip permet d'installer, de désinstaller, d'upgrader, de lister
    Ne supporte que le format `wheels`, ne supporte pas le format `egg` (voir convertisseur)

    Voir comparaison `easy_install VS pip`
  - `PyPI` : Python Package Index : https://pypi.python.org/pypi
- Formats de packaging : `eggs` | `wheels`
  - `eggs` (.egg)
    "A 'Python egg' is a logical structure embodying the release of a specific version of a Python project, comprising its code, resources, and metadata.
    Ancien, date de 2004
  - `wheels` (.whl)
    "wheel is currently considered the standard for built and binary packaging for Python."
    Wheels are the new standard of python distribution and are intended to replace eggs.
    Date de 2012

- Outils de packaging
  Les setuptools avec en particulier le fichier de configuration `setup.py`
- Mais encore ...
  Il existe plein d'outil d'installation et de packaging, par exemple `conda` avec la distribution Python `Anaconda`
  Tout sur l'installation et le packaging Space of Python installation and packaging

- Mise en place d'env. isolé
  `virtualenv` outil permettant de mettre en place un env. isolé
  (par exemple pour installer un module dans votre $HOME, sans droit d'admin et/ou sans perturber votre système)

- Installation avec `easy_install` puis `pip`
  - On dispose d'une lib python avec un module m1 dans un package pck1
    La lib est composée de deux fonctions `get_hello_msg()` et `get_goodbye_msg()`
    La lib est disponible aux formats `egg` (`HelloBye-0.1-py2.7.egg`) et `wheel` (`HelloBye-0.1-py27-none-any.whl`)
    (fichiers crées en utilisant setup.py - vu dans un 2eme temps)

```
>ls /tmp/REPO/
HelloBye-0.1-py27-none-any.whl
HelloBye-0.1-py2.7.egg
```

  - On dispose d'un mini code de vérification, utilisant ce module

```
>ls /tmp/CHECK/
check_m1.py

>more check_m1.py
#!/usr/bin/env python

from pck1.m1 import *

def small_talk() :
  print(get_hello_msg())
  print("What about you ?, bla bla bla" )
  print(get_goodbye_msg())

if __name__ == "__main__":
  small_talk()
```

- Installation avec `easy_install`, localement (/tmp/EI) via l'option `-install-dir`
  - Installation

```
> mkdir  /tmp/EI/ ; setenv PYTHONPATH /tmp/EI/

>easy_install --install-dir=/tmp/EI/ /tmp/REPO/HelloBye-0.1-py2.7.egg

Creating /tmp/EI/site.py
Processing HelloBye-0.1-py2.7.egg
Copying HelloBye-0.1-py2.7.egg to /tmp/EI
Adding HelloBye 0.1 to easy-install.pth file

Installed /tmp/EI/HelloBye-0.1-py2.7.egg
Processing dependencies for HelloBye==0.1
Finished processing dependencies for HelloBye==0.1
```

  - Vérification

```
>ll /tmp/EI/
total 16
-rw-r--r-- 1 bchambon ccin2p3 3007 May 16 15:22 HelloBye-0.1-py2.7.egg
-rw-r--r-- 1 bchambon ccin2p3 2418 May 17 08:18 site.py
drwxr-xr-x 2 bchambon ccin2p3 4096 May 17 08:18 __pycache__
-rw-r--r-- 1 bchambon ccin2p3  210 May 17 08:18 easy-install.pth

>setenv PYTHONPATH /tmp/EI ; /tmp/CHECK/check_m1.py
Good morning
What about you ?, bla bla bla
Bye bye
```

- Installation avec `pip`, localement (/tmp/PIP) via l'option `-target`
  - Installation

```
> mkdir  /tmp/PIP/; setenv PYTHONPATH /tmp/PIP/

> pip install --target /tmp/PIP/ /tmp/REPO/HelloBye-0.1-py27-none-any.whl

Unpacking ./REPO/HelloBye-0.1-py27-none-any.whl
Installing collected packages: HelloBye
Successfully installed HelloBye
Cleaning up...
```

  - Vérification

```
>ll /tmp/PIP/
total 12
drwxr-xr-x 2 bchambon ccin2p3 4096 May 16 16:09 test_pck1
drwxr-xr-x 2 bchambon ccin2p3 4096 May 16 16:09 pck1
drwxr-xr-x 2 bchambon ccin2p3 4096 May 16 16:09 HelloBye-0.1.dist-info

>setenv PYTHONPATH /tmp/PIP/;  /tmp/CHECK/check_m1.py
Good morning
What about you ?, bla bla bla
Bye bye
```

- Packaging (mais pas que) avec `setup.py`
  - Fichier `setup.py` minimal

```
> cat setup.py
#!/usr/bin/env python

from setuptools import setup, find_packages

setup()
```

  - On peut alors disposer de l'aide `-help` ou `-help-commands`

```
> python setup.py --help-commands
Standard commands:
  build             build everything needed to install
  install           install everything from build directory
  upload            upload binary package to PyPI
  sdist             create a source distribution(tarball, zip file, etc.)
  bdist_rpm         create an RPM distribution
...
Extra commands:
...
  test              run unit tests after in-place build
  bdist_egg         create an "egg" distribution
  bdist_wheel       create a wheel distribution (si package wheel dispo, sinon 'pip install wheel')
```

- Objectif : créer les fichier `.egg` | `.whl` précédents avec `setup.py`

Fichiers et répertoires

```
pck1/
  __init__.py
  m1.py

test_pck1/:
  __init__.py
  test_m1.py

main.py

setup.py README
```

Fichier de config setup.py

```
from setuptools import setup, find_packages

setup(
 name = 'HelloBye',
 version = '0.1',
 packages= find_packages(),
 author ='Bernard Chambon',
 author_email='bernard.chambon@cc.in2p3.fr',
 description='To learn ...',
 long_description=open('README').read(),
 url='w.x.y.z'  ,
 test_suite='test_pck1'
)
```

- Jouons avec `setup.py`

```
> python setup.py --description
To learn ...
>  python setup.py --long-description
Here are important notes

> python setup.py --contact-email
bernard.chambon@cc.in2p3.fr
> python setup.py --contact
Bernard Chambon

>python setup.py test
running test
...
test_1 (test_pck1.test_m1.MyTestCase) ... ok
test_2 (test_pck1.test_m1.MyTestCase) ... ok
----------------------------------------------------------------------
Ran 2 tests in 0.000s
```

- Packaging en format `egg` et `wheel`
  - Fichier `HelloBye-0.1-py2.7.egg`

```
> python setup.py bdist_egg
running bdist_egg
...
creating dist
creating 'dist/HelloBye-0.1-py2.7.egg' and adding 'build/bdist.macosx-10.11-intel/egg' to it
removing 'build/bdist.macosx-10.11-intel/egg' (and everything under it)

>ls -l dist/
total 8
-rw-r--r--  1 bchambon  staff  2932 20 sep 17:09 HelloBye-0.1-py2.7.egg
```

  - Fichier `HelloBye-0.1-py2-none-any.whl`

```
> python setup.py bdist_wheel
running bdist_wheel
running build

> ls -l dist/
total 16
-rw-r--r--  1 bchambon  staff  2932 20 sep 17:09 HelloBye-0.1-py2.7.egg
-rw-r--r--  1 bchambon  staff  2575 20 sep 17:12 HelloBye-0.1-py2-none-any.whl
```

- Les setuptools (setup.py) permettent de faire plein d'autre choses, par ex :
  - Compiler du code C (voir commandes build*)
  - Uploader sur un repo PyPI (upload)
  - Installer (voir commandes install*) comme fait précédemment via easy_install | pip install
  - Exécuter des tests (commande test)

- Création d'environnement isolé : `virtualenv`
  Pour pourvoir installer des packages hors système
  (sans disposer des droits admin, sans perturber votre machine)
  - Installation de virtualenv

```
- download de virtualenv (virtualenv-15.1.0-py2.py3-none-any.whl)\\

- install local ds /tmp/virtualenv
>pip install --target /tmp/virtualenv /tmp/virtualenv-15.1.0-py2.py3-none-any.whl
Unpacking ./virtualenv-15.1.0-py2.py3-none-any.whl
Installing collected packages: virtualenv
Successfully installed virtualenv

ou  plus simplement
>pip install --target /tmp/virtualenv  virtualenv
```

  - Création et activation de l'environnement virtuel

```
>mkdir /mnt/ephemeral/DEMO_VE

>/tmp/virtualenv/virtualenv.py /mnt/ephemeral/DEMO_VE
Using base prefix '/usr/local/python/python-3.3'
New python executable in /mnt/ephemeral/DEMO_VE/bin/python
Installing setuptools, pip, wheel...done.

# Activation de l'env. virtuel
>source /mnt/ephemeral/DEMO_VE/bin/activate.csh

# Vérification
>which python
/mnt/ephemeral/DEMO_VE/bin/python

>python --version
Python 3.3.3

>which pip
/mnt/ephemeral/DEMO_VE/bin/pip
```

- Utilisation de l'environnement virtuel crée (et activé) précédement
  - Installation d'une distrib 'HelloBye-0.1-py33-none-any.whl'

```
>pip --verbose install /tmp/REPO/HelloBye-0.1-py33-none-any.whl
Processing /tmp/REPO/HelloBye-0.1-py33-none-any.whl
Installing collected packages: HelloBye

Successfully installed HelloBye-0.1
Cleaning up...

>which python
/mnt/ephemeral/DEMO_VE/bin/python

# Vérification de la lib installée dans l'env. virtuel
>python
Python 3.3.3(default, Feb  5 2014, 16:04:28)
[GCC 4.4.7 20120313(Red Hat 4.4.7-3)] on linux

>>> from pck1.m1 import *
>>> print(get_hello_msg())
Good morning

>>> from pck1.m1 import *
>>> print("{} and {}".format(get_hello_msg(), get_goodbye_msg() ) )
Good morning and Bye bye
```

  - Désactivation de l'environnement virtuel

```
>deactivate

>which python
/usr/local/bin/python

>python --version
Python 2.7.3
```

- A propos de la version de Python disponible avec `virtualenv`
  - Par défaut
    La version de Python disponible dans `virtualenv` est celle avec lequel (`virtualenv`) il a été installé
  - Mais ...
    Il est possible de définir une autre version (`virtualenv -python ...`), pour peu que cette version soit disponible sur votre machine
    $\Leftrightarrow$ pour disposer d'une nouvelle version de python en utilisant `virtualenv` , il faut d'abord disposer localement (télécharger) cette nouvelle version de Python

Exercise 5