Comment packager et distribuer son paquet Python

Yanis Guenane

T'es qui toi ?

- **Yanis Guenane**
  - **irc (freenode): spredzy**
  - **Github: Spredzy**
  - **Twitter: @YanisGuenane**

- **Infrastructure/Software Engineer @Red Hat**

- **Aime:**
  - **Automation**
  - **Packaging**
  - **Burgers**
  - **Café**
  - **Voyage**

Pourquoi ?

Différences ?

```
#!/bin/python

def helloworld():
    """Print hello"""

    print('hello')

def main():
    helloworld()

if __name__ == '__main__':
    main()
```

Pourquoi ?

- **Distribution facile**
- **Intégration natives aux différents OS**
- **Installation facile, intégration avec les outils d'automatisation (Puppet, Ansible, etc…)**
- **Faciliter les mises à jour, correctifs, cycle de vie d'un projet**

Comment ?

~~Les outils~~ l'outil :

- **~~distutils~~**
- **setuptools**

foo.py

```
#!/bin/python

def helloworld():
    """Print hello"""

    print('hello')

def main():
    helloworld()

if __name__ == '__main__':
    main()
```

```
jdoe@polaris(~/foo)$> ls -a
foo.py .git
jdoe@polaris(~/foo)$> python foo.py
hello
jdoe@polaris(~/foo)$> git add foo.py
jdoe@polaris(~/foo)$> git commit -m "foo: display hello"
jdoe@polaris(~/foo)$> git push origin master
```

foo.py → the foo project

```
jdoe@polaris(~/foo)$> ls -lR
.:
total 0
drwxrwxr-x. 2 jdoe jdoe  60 Mar  8 08:04 bin
drwxrwxr-x. 2 jdoe jdoe 100 Mar  8 08:04 foo
-rw-rw-r--. 1 jdoe jdoe   0 Mar  8 08:03 README.rst
drwxrwxr-x. 2 jdoe jdoe  80 Mar  8 08:04 tests

./bin:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 foo

./foo:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 bar.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 foo.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:03 __init__.py

./tests:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 test_bar.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 test_foo.py
```

setuptools et setup.py

```
import setuptools

setuptools.setup(
    name='foo',
    version=0.1.5,
    packages=setuptools.find_packages(exclude=['tests', 'docs']),
    license='Apache v2.0',
    url='https://github.com/polaris/foo',

    author='John Doe',
    author_email='jdoe@polaris.com',
    description='My very own first python library',
    long_description='My very own first python library',

    install_requires=['requirement1', 'requirement2'],
    classifiers=[
        'Environment :: Console',
        'Intended Audience :: Developers',
        'License :: OSI Approved :: Apache Software License',
        'Operating System :: POSIX :: Linux',
        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3.4',
    ],
    entry_points={
        'console_scripts': [
            'foo=foo:main',
        ],
    },
)
```

The foo project avec setup.py

```
jdoe@polaris(~/foo)$> ls -lR
.:
total 0
drwxrwxr-x. 2 jdoe jdoe  60 Mar  8 08:04 bin
drwxrwxr-x. 2 jdoe jdoe 100 Mar  8 08:04 foo
-rw-rw-r--. 1 jdoe jdoe   0 Mar  8 08:03 README.rst
-rw-rw-r--. 1 jdoe jdoe 987 Mar  8 08:03 setup.py
drwxrwxr-x. 2 jdoe jdoe  80 Mar  8 08:04 tests

./bin:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 foo

./foo:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 bar.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 foo.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:03 __init__.py

./tests:
total 0
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 test_bar.py
-rw-rw-r--. 1 jdoe jdoe 0 Mar  8 08:04 test_foo.py
```

python setup.py all the way

```
Standard commands:

  build             build everything needed to install
  build_py          "build" pure Python modules (copy to build directory)
  build_ext         build C/C++ extensions (compile/link to build directory)
  build_clib        build C/C++ libraries used by Python extensions
  build_scripts     "build" scripts (copy and fixup #! line)
  clean             clean up temporary files from 'build' command
  install           install everything from build directory
  install_lib       install all Python modules (extensions and pure Python)
  install_headers   install C/C++ header files
  install_scripts   install scripts (Python or otherwise)
  install_data      install data files
  sdist             create a source distribution (tarball, zip file, etc.)
  register          register the distribution with the Python package index
  bdist             create a built (binary) distribution
  bdist_dumb        create a "dumb" built distribution
  bdist_rpm         create an RPM distribution
  bdist_wininst     create an executable installer for MS Windows
  upload            upload binary package to PyPI
  check             perform some checks on the package
```

python setup.py all the way

```
Extra commands:

  saveopts            save supplied options to setup.cfg or other config file
  compile_catalog     compile message catalogs to binary MO files
  develop             install package in 'development mode'
  upload_docs         Upload documentation to PyPI
  easy_install        Find/get/install Python packages
  init_catalog        create a new catalog based on a POT file
  test                run unit tests after in-place build
  update_catalog      update message catalogs from a POT file
  flake8              Run flake8 on modules registered in setuptools
  setopt              set an option in setup.cfg or another config file
  nosetests           Run unit tests using nosetests
  install_egg_info    Install an .egg-info directory for the package
  rotate              delete older distributions, keeping N newest files
  egg_info            create a distribution's .egg-info directory
  alias               define a shortcut to invoke one or more commands
  extract_messages    extract localizable strings from the project code
  bdist_egg           create an "egg" distribution
  build_sphinx        Build Sphinx documentation
```

Share, gather feedback, rinse, repeat

~/.pypirc

```
[distutils]
index-servers=pypi

[pypi]
repository = https://pypi.python.org/pypi
username = <username>
password = <password>
```

Register le projet

```
jdoe@polaris (~/foo) $> twine register dist/*
```

Upload le projet

```
jdoe@polaris (~/foo) $> twine upload dist/*
```

Le projet est disponible, téléchargez le !

```
jdoe@polaris (~/foo) $> pip install foo
```

One step closer to world domination !

```
Name:           python-foo
Version:        1.2.3
Release:        1%{?dist}
Summary:        An example python module
License:        MIT
URL:            http://pypi.python.org/pypi/foo
Source0:        http://pypi.python.org/packages/source/f/foo/foo-1.2.3.tar.gz
BuildArch:      noarch
BuildRequires:  python2-devel

%description
An python module which provides a convenient example

%prep
%autosetup -n %{srcname}-%{version}

%build
%py2_build

%install
%py2_install

%check
%{__python2} setup.py test

%files
%license LICENSE
%doc README.rst
%{python2_sitelib}/*
%{_bindir}/foo

%changelog
* Tue March 8 2016 John Doe  <jdoe@polaris.com> 1.2.3-1
- Initial commit
```

Links

Liens propre au packaging python:

- Python Package Authority - https://packaging.python.org/
- Setuptools – https://pypi.python.org/pypi/setuptools/

Guidelines du packaging python selon les distributions:

- Fedora/CentOS/RH - https://fedoraproject.org/wiki/Packaging:Python
- Debian – https://wiki.debian.org/Python/Packaging

Conclusion

Don't believe the myth, Python packaging is easy !