Packaging Python Code: an introduction

Muzzamil LUQMAN (L3i) and Antoine FALAIZE (LaSIE)

LaSIE Seminar
23/11/2017
Université de La Rochelle

What is a Python package

Recall a module is a single file (or files) that are imported under one import

```
import my_module
```

A package is a collection of modules in directories that give a package hierarchy

```
from my_package.timing import afunction
```

But, a “Python package” can refer to the distribution sources as well (confusion)

Why packaging your Python code?

- To import a module (or a package), it must be in your Python path
- Without a package structure, you need
  - to copy all your modules in the current working directory or
  - to append the path to your modules in the Python path

```
import sys

sys.path.append('/path/to/my/modules')
```

- With a package structure, you can
  - let all you code in a single directory (e.g. a GIT repository!)
  - use **`setuptools`** to install globally and import your modules from any location
  - distribute your work through **PyPI** so that it installs with **`pip`**

Why publishing your code on PyPI ?

from the website:

The Python Package Index is a repository of software for the Python programming language. There are currently 122544 packages here.

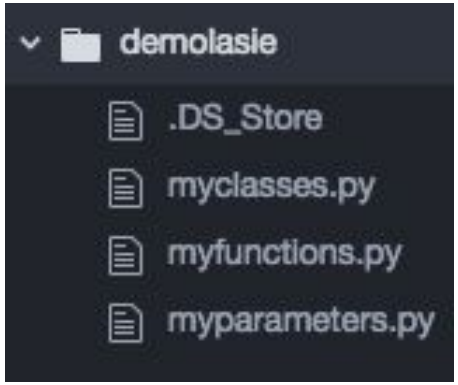
- A package uploaded on PyPI can be installed easily on every platforms
- This gives **visibility** and **accessibility** to your work!

Requirements

1. some Python code
   - obviously
2. setuptools: To install your package locally
   - Usually available in Python distributions
3. Pip: To easily install packages and to connect to PyPI
   - Usually available in Python distributions
4. A PyPI account
   - To upload your package to PyPI
5. Twine: To securely push your code to PyPI
   - install with `pip install twine`

1. Make a package

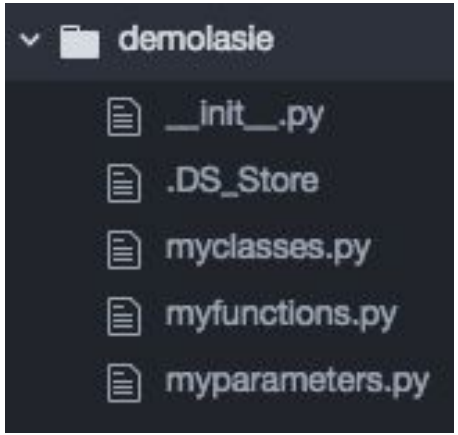
Suppose you have some code in a folder:



bunch of python scripts

Simply add an empty python script

`__init__.py`



a python package

Details on what to put in this script

2. Make a package installable

1. Rebase your package in a **`project_directory`**
2. Add a **`setup.py`**
   - **`to configure the installation`**
3. Add a **`setup.cfg`**
   - **`to configure the build of sources`**
4. add a **`README.rst`**
   - **`to explain why/how to use your package`**
5. add a **`MANIFEST.in`**
   - **`to distribute additional material`**
6. add a **`LICENCE.txt`**
   - **`to explain what can be done with your package`**
   - **`we use the French Academic Licence CeCILL`**
7. add **`AUTHORS`**, **`CHANGELOG`**, and **`requirement.txt`** files (optional)

2. Make a package installable

- Now you can install your package globally with

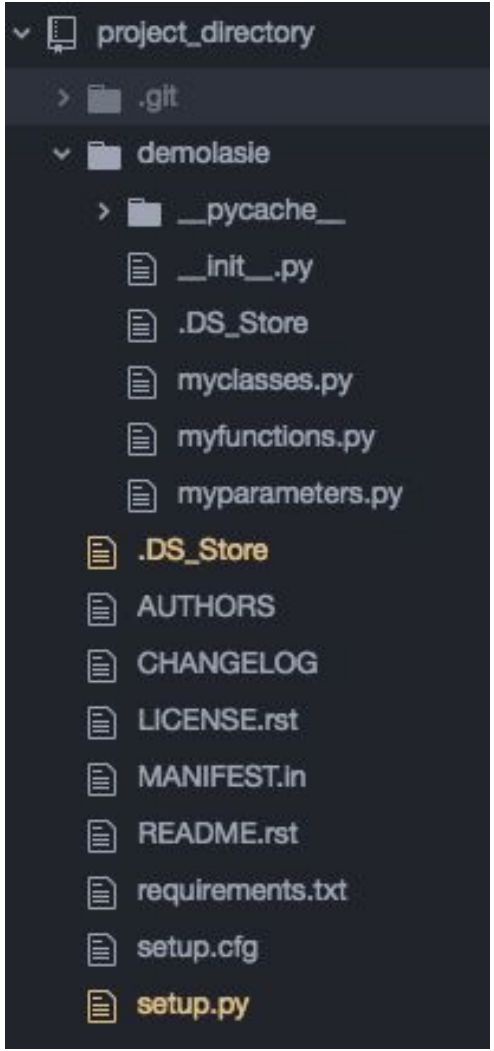
```
pip install .
```

- To allow further editing of the code, use

```
pip install -e .
```

(both from the `project_directory`)

[Details on packages installation]



3. Prepare the distribution sources

To have your project installable from a Package Index like PyPI, you'll need to create a Distribution (aka "Package") for your project.

This is done with

```
python setup.py sdist
```

This build the sources in the `project_directory`

Details on sources building (this is the real packaging)

4. Push your Package to PyPI

If all the preceding succeed, you are now ready to push your code to PyPI.

Simply run (still from the `project_directory`)

```
twine upload dist/*
```

And that's all!

Now anyone can install your project from anywhere with

```
pip install your_project
```

Details on uploading sources

Further documentation

About Python packages installation:

Installing Packages

About distributing on PyPI

Packaging and Distributing Projects