Share your Python code

Raphael Bacher

12 02 2019

Plan

- Aspects Juridiques
- Partager un code ça implique quoi ?
- Publier un code Python comment faire ?

Aspects Juridiques

Questions

- Point sur la Loi pour une république numérique
- Qu'ai-je le droit de faire de mon code ?
- Quelle démarche à suivre pour diffuser mon code ?
- Quelle licence choisir ?

Loi pour une république numérique

Promulguée en Octobre 2016

Deux impacts majeurs pour la recherche :

- Protection des publications Open Access
- “Forte incitation” à la publication des codes

Qu'ai-je le droit de faire avec mon code

- Le code produit appartient à ma tutelle (sauf en stage !)
- Sauf cas spécial (partenariat privé,. . .) -> encouragé à le rendre public
- Possibilité de faire un dépot de logiciel

Note : les articles de recherche appartiennent au chercheur

La démarche à suivre

- Demande auprès du directeur de labo/unité
- Si dépot de logiciel -> service valorisation tutelle

Le choix d'une licence

THE website : https://choosealicense.com/

Mettre mon package python en ligne

Publier un code

- Dépot des sources
- Package
- Licence
- Documentation
- Tests
- CI
- Contributions

Publier un code Python

- sur github
- sur binder
- sur pypi
- sur conda-forge

Créer son paquet python

Module/paquet

- Un module : un fichier name.py
- Un paquet : une collection de modules Python, un dossier de fichiers-modules Python avec un fichier **init**.py
- Un paquet d'installation : un paquet python distribuable et installable

Historique

- easy_install
- distutils
- eggs
- ...

L'approche "standard" actuelle

- pip pour installer les paquets
- setuptools pour construire et installer un paquet
- wheel pour construire les paquets binaires
- twine pour uploader votre package sur PyPI

Le guide de référence

https://packaging.python.org/

Conda-forge

Construire un paquet conda à partir d'un paquet Pypi

https://conda.io/docs/user-guide/tutorials/build-pkgs-skeleton.html

Publier sur conda-forge

- https://conda-forge.org
- https://conda-forge.org/docs/recipe.html

Documenter mon code

- pep8, flake8
- docstrings, . . .
- sphinx [https://www.pythonforthelab.com/blog/documenting-with-sphinx-and-readthedocs/#id1]
- nb_sphinx [https://nbsphinx.readthedocs.io/en/0.3.5/usage.html]
- Readthedocs

Tester mon code

- pytest https://pytest.readthedocs.io/en/latest/
- Integration continue

Tutoriel :

http://katyhuff.github.io/python-testing/

Intégration Continue (CI)

Permet d'automatiser les tests (entre autres)

- Le développeur envoie son code sur le dépot.
- le processus de CI construit le projet et fait tourner les tests
- Le développeur est notifié du résultat

Plateformes

- Travis (avec Github)
- gitlab-ci (avec Gitlab)
- et d'autres (Jenkins, Appveyor...)

Gitlab-ci

File .gitlab-ci.yml

```
# requirements for testing
before_script:
- pip install pytest
# script #1
python:
script:
  - python -m pytest
```

Maintenir son paquet

- Gérer les pull requests/issues sur Github
- Faire des releases
- Utiliser des tests/integration continue

Cookiecutter

https://cookiecutter-pypackage.readthedocs.io/en/latest/tutorial.html