Publishing my Python packages

Making open source 101

Who am I?

Felipe Martin

- 1987
- CTO at Saluspot
- Dev + IT + OP + Manager...
  - aka: *Problem solver*
- fmartingr.com
- **@fmartingr** on most social.

Disclaimer

- I'm no expert.
- This are just some tricks or stuff I found out when publishing my own packages.
- There's no stupid question, feel free to ask anything.
- Same for corrections, if I'm wrong just tell me!

What are we going to learn

1. Create a python package/module
2. Publish your code to Github (or gitlab)
3. Upload your code to PyPI
4. Improve the experience for the community

1

Creating a python module

A python implementation of left-pad

Because yeah, we need that.

1.1 Create your module

```
# leftpad.py

def leftpad(str, fillchar='0', length=5):
    return str.rjust(length, fillchar)
```

1.2 Add some docstrings! PEP 257

```
# leftpad.py

def leftpad(string, fillchar='0', length=5):
    """
    Take str and fill from the left with the provided characters

    Args:
        string (str) The string to perform te fill into
        fillchar (str) The character used to fill the string
        length (int) The total lenght of the resulting string

    Returns:
        string
        The filled string
    """
    return string.rjust(length, fillchar)
```

1.2.1 Improve docstrings

- PEP 0484 Type Annotations for Python 3.5+.
- pydoc Builtin documentation generator.
- Sphinx-doc Special formatting for auto-generating documentation from source code.

There are more out there, just find out which one is best for you and your team.

1.3 Create tests to check your code

```
# tests.py
from unittest import TestCase

import leftpad

class LeftPadTestCase(TestCase):
    def test_no_arguments_use_defaults(self):
        test_str = '1'
        self.assertEquals(leftpad.leftpad(test_str), '00001')

# ...
```

1.4 Create a proper README file

```
pyleftpad
=========

A python implementation for leftpad

# Summary
...

# Installation
...

# Usage
...

# License
...
```

The tree right now

```
leftpad/
|-leftpad.py
|-README.md
`-tests.py
```

2

Publish your code to Github (or gitlab)

I'm assuming everyone knows this one

So I'm not going into detail here

3

Upload your code to PyPI

3.1 Create a PyPI account

- Go to the PyPI register page
- Create an account
- Woah! That was fast!

3.2 Create a `setup.py` file in your project

```
import os
from setuptools import find_packages, setup

setup(
    name='pyleftpad',  # Name of your package (pip install pyleftpad)
    version='0.1.0', # Version for your package
    packages=find_packages(exclude=['tests.py']), # Packages to include
    description='Leftpad implementation in python',
    url='https://github.com/fmartingr/pyleftpad.git',
    author='Felipe Martin',
    author_email='me@fmartingr.com',
    classifiers=[
            # See https://pypi.python.org/pypi?%3Aaction=list_classifiers
    ],
)
```

3.3 MANIFEST.in vs setup.py `packages`

- MANIFEST file is used to include extra files our package needs in addition to the python modules.
- setup.py `packages` parameters determines which packages are going to be included within the installation.
  - ⚠️ Beware! It's not recursive.
  - Use `find_packages()` with exclusion parameter.

3.4 Ignore not needed files from setuptools

When creating a new distribution setuptools will use our projects folder to store the files, ignore them so you don't accidentally commit them into the codebase.

```
# .gitignore
# ...
# setuptools
dist/
build/
MANIFEST
```

3.5 Publish your release

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

Handy helper:

```
# Add this to your `setup.py`
if sys.argv[-1] == 'publish':
    os.system("python setup.py sdist upload")
    args = {'version': VERSION}
    print("You probably want to also tag the version now:")
    print("  git tag -a %(version)s -m 'version %(version)s'" % args)
    print("  git push --tags")
    sys.exit()
```

```
$ python setup.py publish
```

3.99 Resources

- Setuptools documentation
- Django: How to write reusable apps

4

Improve the experience for the community

4.1 Choose a license

- Some people will ask you to put a license to the project before making a issue or a merge request, so choose one before publishing it.
- Typical are MIT or GPLv2.
- Use choosealicense.com

4.2 Explain how to properly create issues/MR

- Let people know how to correctly fill an issue before the first communication it's made, this will be way faster rather than asking for more stuff.
- This will also prevent friction when users create incomplete or unrelated issues.
- Github and Gitlab provide templates for this.

4.3 Create a changelog and proper versioning

- Stablish a semantic versioning for your project
- Use git tags properly (`git tag -h`)
- Create a proper changelog
  - Github provides a releases tab
  - You can also create a `CHANGELOG.md`

4.4 Automated unit tests with tox & travis-ci

- Tox
- Travis-ci

There are others:

- Nose
- py.test
- CircleCI
- Codeship
- ...

4.4.1 Configure tox

- Install tox `pip install tox`
- Create a `tox.ini` in your project:

```
[tox]
envlist = py26, py27, py33, py34, py35, pypy

[testenv]
commands =
    nose tests.py
deps = -r{toxinidir}/test_requirements.txt
```

4.4.2 Configure travis

- Create a `.travis.yml` file in the root of your project

```
language: python
env:
  - TOXENV=py26
  - TOXENV=py27
  - TOXENV=py33
  - TOXENV=py34
  - TOXENV=py35
  - TOXENV=pypy
install:
  - travis_retry pip install tox==2.3.1
script:
  - travis_retry tox
```

4.4.3 Sign up for Travis-CI and activate project

- Go to Travis-CI and create an account.
- Search and enable your project for builds
- Push a new commit or create a new pull request
- Done!

Now you can be a cool kid too and add the super cool build status badge to your `README` file!

4.5 Create an `AUTHORS` file

- Make people add themselves when creating a MR
  - Say this in the contribution guidelines

Conclusion

- Don't be afraid of publishing your code.
- Follow the rule: If it's useful for you, it may be useful for others.
- Same rules apply when you make merge requests.
- Always learn something.
- Automate everything.
- Be kind to the community.

Thank you!

Q&A