

Python ~~packaging~~ (best practices and community building) October 8, 2019

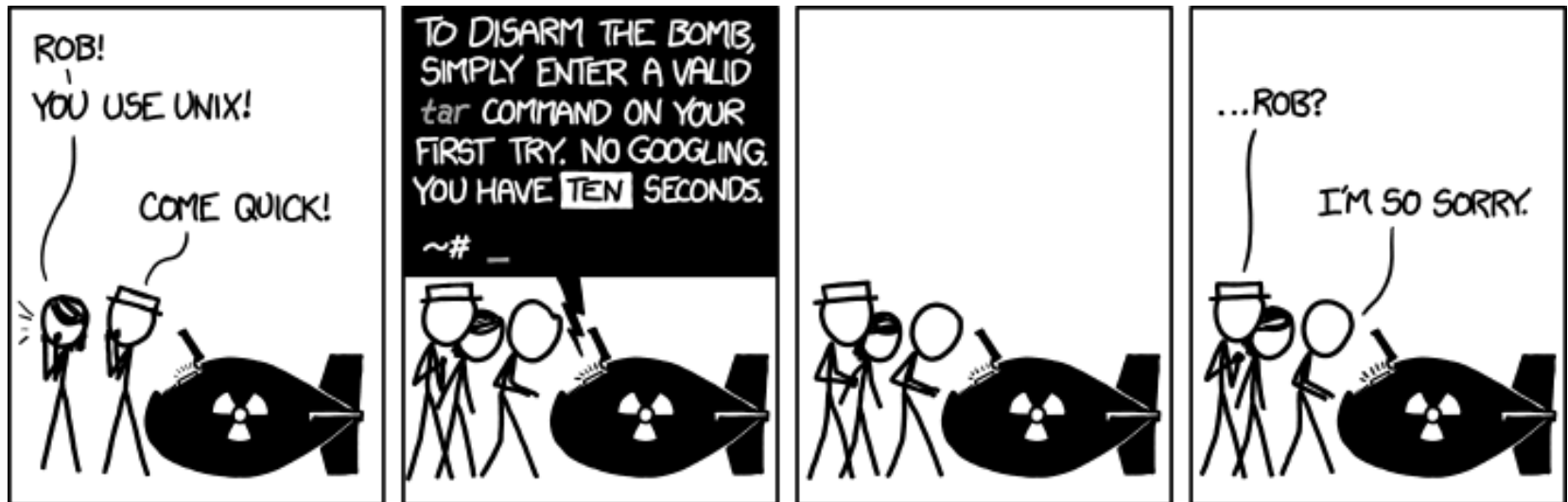


Before we start

<https://hacktoberfest.digitalocean.com>



Motivation

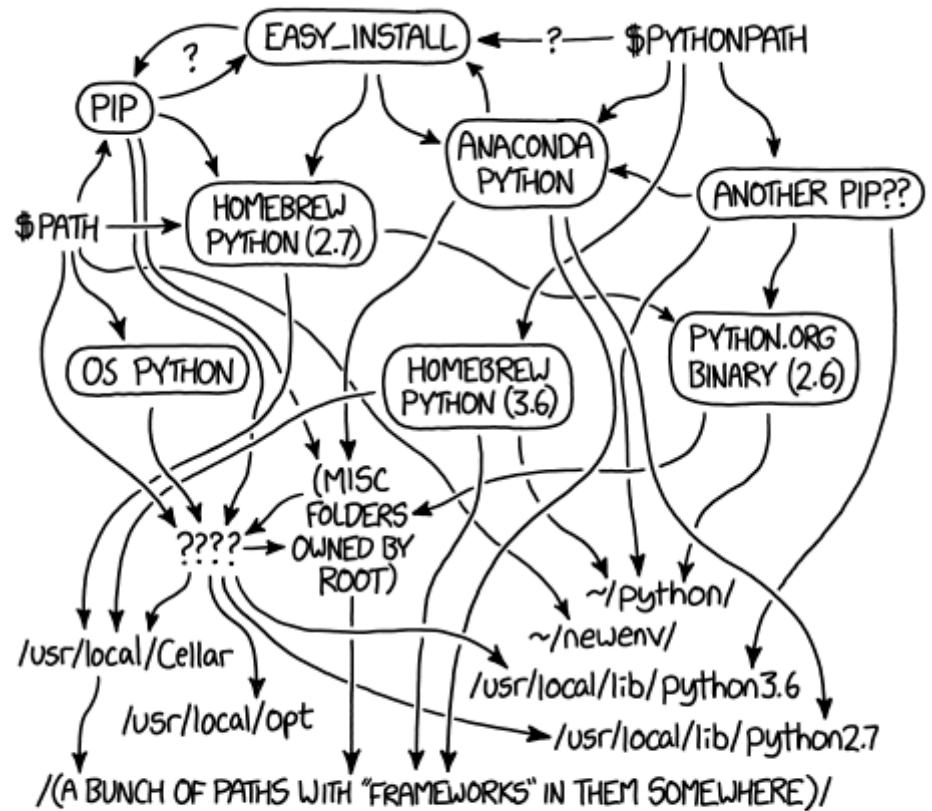


How bad is the packaging situation?

INSTALL.SH

```
#!/bin/bash
```

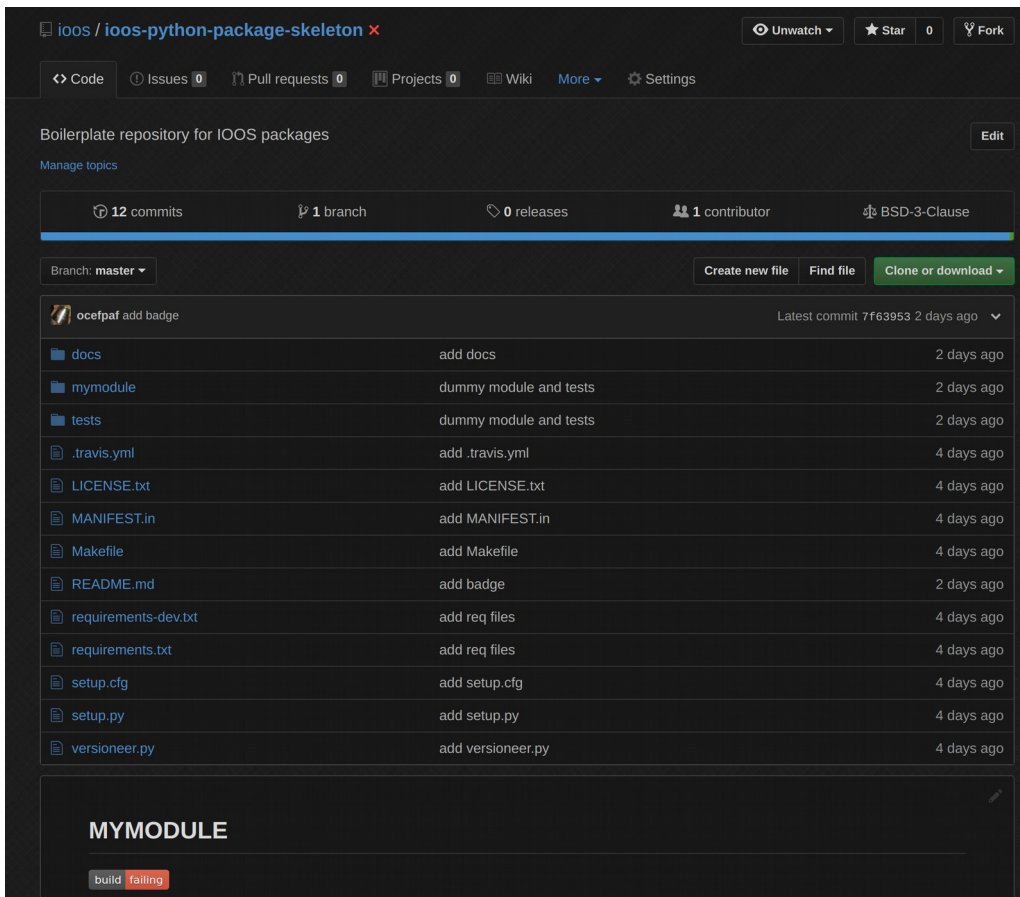
```
pip install "$1" &  
easy_install "$1" &  
brew install "$1" &  
npm install "$1" &  
yum install "$1" & dnf install "$1" &  
docker run "$1" &  
pkg install "$1" &  
apt-get install "$1" &  
sudo apt-get install "$1" &  
steamcmd +app_update "$1" validate &  
git clone https://github.com/"$1"/"$1" &  
cd "$1"; ./configure; make; make install &  
curl "$1" | bash &
```



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

One-stop shop for packaging needs

<https://github.com/ioos/ioos-python-package-skeleton>



The screenshot shows the GitHub repository page for `ioos / ioos-python-package-skeleton`. The repository is a boilerplate for IOOS packages, currently on the `master` branch. It has 12 commits, 1 branch, 0 releases, and 1 contributor. The repository is licensed under BSD-3-Clause. The file list includes:

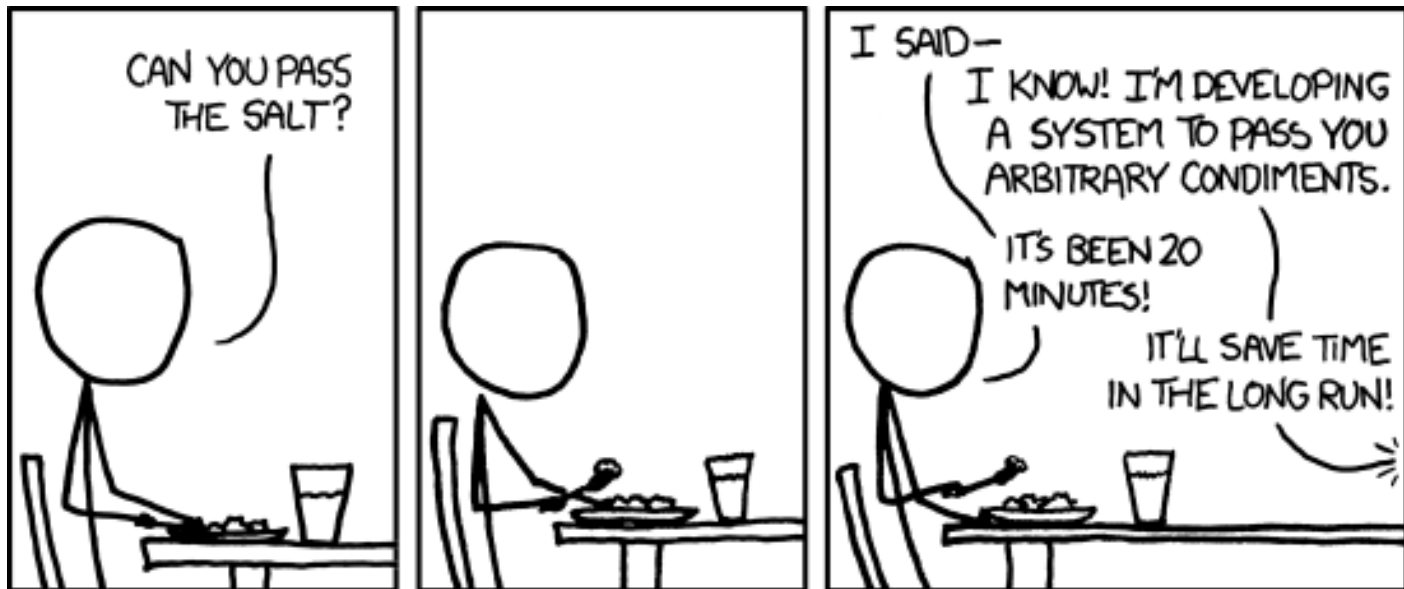
File	Description	Commit Date
<code>docs</code>	add docs	2 days ago
<code>mymodule</code>	dummy module and tests	2 days ago
<code>tests</code>	dummy module and tests	2 days ago
<code>.travis.yml</code>	add .travis.yml	4 days ago
<code>LICENSE.txt</code>	add LICENSE.txt	4 days ago
<code>MANIFEST.in</code>	add MANIFEST.in	4 days ago
<code>Makefile</code>	add Makefile	4 days ago
<code>README.md</code>	add badge	2 days ago
<code>requirements-dev.txt</code>	add req files	4 days ago
<code>requirements.txt</code>	add req files	4 days ago
<code>setup.cfg</code>	add setup.cfg	4 days ago
<code>setup.py</code>	add setup.py	4 days ago
<code>versioneer.py</code>	add versioneer.py	4 days ago

At the bottom, there is a section for `MYMODULE` with a `build failing` status.



Are we doing this right?

(or “Why not a cookie-cutter?”)



Project structure

| -docs

| | -source

| | | -_static

| | -build

| -tests

| -mymodule

setup.py

```
setup(  
    name="mymodule",  
    python_requires='>=3.6',  
    version=versioneer.get_version(),  
    description="My Awesome module",  
    license="BSD-3-Clause",  
    long_description=f'{read("README.md")}',  
    long_description_content_type="text/markdown",  
    author="AUTHOR NAME",  
    author_email="AUTHOR EMAIL",  
    packages=find_packages(),  
    install_requires=install_requires,  
    cmdclass=versioneer.get_cmdclass(),  
)
```


setup.py

```
with open("requirements.txt") as f:  
    requires = f.readlines()
```

```
install_requires = [req.strip() for req in requires]
```

requirements.txt

OWSLib>=0.8.3

Jinja2>=2.7.3

functools32==3.2.3-2; python_version < '3.2' #conda: functools32 (only python=2)

setup.cfg

```
[versioneer]
```

```
...
```

```
[tool:pytest]
```

```
flake8-max-line-length = 105
```

```
flake8-ignore =
```

```
    versioneer.py ALL
```

```
    mymodule/_version.py ALL
```

```
[metadata]
```

```
description-file = README.md
```

```
license_file = LICENSE.txt
```

```
[isort]
```

```
atomic=true
```

```
[check-manifest]
```

```
ignore =
```

```
    .travis.yml
```

Code pause

NEVER HAVE I FELT SO
CLOSE TO ANOTHER SOUL
AND YET SO HELPLESSLY ALONE
AS WHEN I GOOGLE AN ERROR
AND THERE'S ONE RESULT
A THREAD BY SOMEONE
WITH THE SAME PROBLEM
AND NO ANSWER
LAST POSTED TO IN 2003



PEP 517/518

- standardized non-executable config file;
- many backends, one spec:
 - poetry, setuptools, pipenv(?), flit, conda, etc;
 - all should support pip installs.

Refs.:

<https://www.python.org/dev/peps/pep-0517>

<https://www.python.org/dev/peps/pep-0518>

<https://medium.com/@grassfedcode/pep-517-and-518-in-plain-english-47208ca8b7a6>

MANIFEST.in

```
include *.txt
```

```
include LICENSE
```

```
include README.md
```

```
recursive-include mymodule *.py
```

```
include versioneer.py
```

README.md

You should always have a README in your projects!

LICENSE

Copyright 2017 AUTHOR NAME

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY

WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

.travis.yml

language: minimal

sudo: false

env:

 global:

 - secure: "TOKEN"

matrix:

 fast_finish: true

 include:

 - name: "python-3.6"

 env: PY=3.6

 - name: "python-3.7"

 env: PY=3.7

 - name: "coding_standards"

 env: PY=3.7

 - name: "tarball"

 env: PY=3.7

 - name: "docs"

 env: PY=3.7

.travis.yml

before_install:

```
# Install miniconda and create TEST env.  
- |  
  wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O  
  miniconda.sh  
  bash miniconda.sh -b -p $HOME/miniconda  
  export PATH="$HOME/miniconda/bin:$PATH"  
  conda config --set always_yes yes --set changeps1 no --set show_channel_urls true  
  conda update --quiet conda  
  conda config --add channels conda-forge --force  
  conda config --set channel_priority strict  
  conda config --set safety_checks disabled  
  conda create --name TEST python=$PY --file requirements.txt --file requirements-  
dev.txt  
  source activate TEST  
  conda info --all
```

install:

```
- pip install -e . --no-deps --force-reinstall
```

requirements-dev.txt

black
check-manifest
doctr
flake8
flake8-builtins
flake8-comprehensions
flake8-mutable
flake8-print
isort
nbsphinx
pylint
pytest
pytest-cov
pytest-flake8
pytest-xdist
sphinx
twine
wheel

.travis.yml

script:

- if [[\$TRAVIS_JOB_NAME == python-*]]; then
 cp -r tests/ /tmp ;
 pushd /tmp && pytest -n 2 -rxs --cov=mymodule tests && popd ;
fi
- if [[\$TRAVIS_JOB_NAME == 'tarball']]; then
 pip wheel . -w dist --no-deps ;
 check-manifest --verbose ;
 twine check dist/* ;
fi
- if [[\$TRAVIS_JOB_NAME == 'coding_standards']]; then
 pytest --flake8 -m flake8 ;
fi

.travis.yml

```
- |
  if [[ $TRAVIS_JOB_NAME == 'docs' ]]; then
    set -e
    cp notebooks/{quick_intro.ipynb,searchfor.ipynb} docs/source/
    pushd docs
    make clean html linkcheck
    popd
    if [[ -z "$TRAVIS_TAG" ]]; then
      python -m doctr deploy --build-tags --key-path key.enc --built-docs
docs/_build/html dev
    else
      python -m doctr deploy --build-tags --key-path key.enc --built-docs
docs/_build/html "version-$TRAVIS_TAG"
      python -m doctr deploy --build-tags --key-path key.enc --built-docs
docs/_build/html .
    fi
  fi
```

.travis.yml

```
deploy:  
  skip_cleanup: true  
  provider: pypi  
  user: ioos  
  password:  
    secure: "TOKEN"  
  distributions: sdist bdist_wheel  
  upload_docs: no  
  on:  
    repo: ioos/mymodule  
    tags: true  
    all_branches: master  
    condition: '$TEST_TARGET == "docs"'
```

Must have:

- README
 - install instructions
- License
- docs
- unittest tests
- CIs

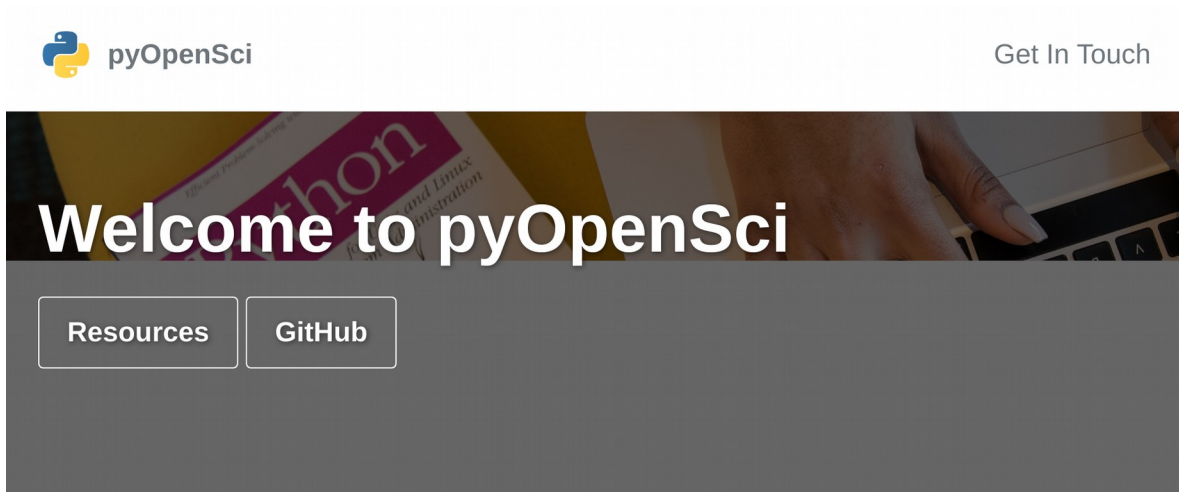
Nice to have:

- automatic version number from tags
- auto-publish docs and tarball
- tarball automated checks
- standard style: black, lints, isort
- integration tests
- Windows CI
- Your pkg in conda-forge

Also nice to have:

- CONTRIBUTING.rst
- .github/

Extra: pyOpenSci



About pyOpenSci

pyOpenSci promotes open and reproducible research through peer-review of scientific Python packages. We also build technical capacity by providing a curated repository of high-quality packages and enabling scientists to write and share their own software. We hope to foster a greater sense of community among scientific Python users so that we can help each other become better programmers and researchers.

pyOpenSci is being modeled after the successful [rOpenSci](#) community.

That's all folks!

The screenshot shows a GitHub repository page for 'bouk / monkey'. At the top, there are navigation links for 'Code', 'Pull requests 1', 'Actions', 'Releases 2', and 'More'. On the right, there are buttons for 'Watch', 'Star 1,408', and 'Fork'. Below the navigation, the current branch is 'master' and the file path is 'monkey / LICENSE.md'. There are buttons for 'Find file' and 'Copy path'. A commit by 'bouk' is shown, titled 'Update LICENSE.md', with commit hash 'ca6af77' and date 'on May 27'. Below the commit, it says '1 contributor'. The file details show '6 lines (3 sloc)' and '180 Bytes'. There are buttons for 'Raw', 'Blame', and 'History'. The content of the LICENSE.md file is displayed in a dark theme:

```
Copyright Bouke van der Bijl

I do not give anyone permissions to use this tool for any purpose. Don't use it.

I'm not interested in changing this license. Please don't ask.
```