Python Packaging

for users and devs

2024-05-17

Guilhem Saurel

This presentation

Available at

`https://homepages.laas.fr/gsaurel/talks/python-packaging.pdf`

Under License

This presentation (continued)

Source

`https://gitlab.laas.fr/gsaurel/talks : python-packaging.md`

Discussions

https://matrix.to/#/room/#allo-pi2:laas.fr

Introduction

Goals

1. Use python packages from other people
2. Provide your own python packages to other people
3. Get an overview of different Package Managers for that

OS Scope

1. linux
2. macos, *BSD
3. ~~windows~~

OS Scope

1. linux
2. macos, *BSD
3. windows
   - in pure python
   - with WSL

Part 1: Use python packages

Do you really need it ?

Dependency ≈ Addiction

Do you really need it ?

Dependency ≈ Addiction

- is this dependency essential ?
- can it be made optional ?
- what about its own dependencies ?

Is it good enough ?

- is it pure python ?
- are you confident in its future ?
- are you sure you will be able to handle its updates ?

Document your dependencies

Either:

1. `README.md`
2. `requirements.txt`
3. `pyproject.toml`

Document your dependencies

Either:

1. `README.md`
2. `requirements.txt`
3. `pyproject.toml`

So that:

1. you won't forget
2. you can `pip install -r requirements.txt`
3. you can `pip install .`

Install them

- Use a venv
- Troubleshooting: `$PYTHONPATH`

Install them

- Use a venv
- Troubleshooting: `$PYTHONPATH`

Also troubleshooting:

```python
import sys
print(sys.path)
```

Dependencies versions

- keep them up to date
- document your needs
- document what won’t work

Versions constraints & pinning

- constraint graphs grow quickly
- solutions can change over time
- use a lock file with your current working solution

Versions constraints & pinning

- constraint graphs grow quickly
- solutions can change over time
- use a lock file with your current working solution
- `pip freeze > requirements.lock`

```
Django==4.2.11
httpx==0.27.0
ipython==8.23.0
jedi==0.19.1
Jinja2==3.1.3
matplotlib-inline==0.1.6
numpy==1.26.4
pandas==2.2.1
tqdm==4.66.2
```

Part 2: Distribute your packages

Follow community standards

- `ruff format`
- `ruff check`

Follow community standards

- `ruff format`
- `ruff check`

Use those in your IDE, git hooks, and/or CI

Choose a license

https://spdx.org/licenses/

eg.: BSD / MIT / Apache / GPL

Choose a package builder

- setuptools
- poetry
- flit

Write a pyproject.toml

- name
- version
- authors
- license
- urls
- dependencies
- entrypoints
- tooling configuration

ref. your builder docs

Test your packaging

- `python -m build`
- `pip install .`
- in your CI

Test your packaging

- `python -m build`
- `pip install .`
- in your CI

This is enough for other people to use eg.:

```
pip install \
    git+https://gitlab.laas.fr/gsaurel/ndh
```

Create a release

- decide a version number: https://semver.org/
- document changes between versions: https://keepachangelog.com/
- publish a git tag (bonus points if signed)
- publish package archives (bonus points if signed)

(Optionnal) Publish on PyPI

- twine
- `flit publish`
- `poetry publish`
- github.com/pypa/gh-action-pypi-publish

Part 3: Some python package managers

Your distribution package manager

- apt
- pacman
- rpm

This is the most simple and most stable solution.

pip

This is the incontournable standard solution.

pip-tools

https://github.com/jazzband/pip-tools

Simple dependency constraint declaration + solution pinning

poetry

https://python-poetry.org/

Full feature and widest adoption.

pipenv

Should be deprecated in favor of poetry.

pdm

https://pdm-project.org/latest/

A bit more modern than poetry, but narrower adoption and support.

conda / mamba

This will eat your home.

uv

https://github.com/astral-sh/uv

The new cool kid.

nix

The perfection you didn’t ask for, yet.

nix

The perfection you didn’t ask for, yet.

Come to the next formations to know more !

Questions ?

Links

Prior art

- Managing Python Packages (2019)
- Python Tooling (2022)

This presentation

`https://homepages.laas.fr/gsaurel/talks/python-packaging.pdf`

https://matrix.to/#/room/#allo-pi2:laas.fr