Building Python Projects

PEP 517 & PEP518 to the rescue

Dr. Cristián Maureira-Fredes
Software Engineer @ TQtC

@cmaureir

The ~~Chicken~~Wheel and Egg Problem

- Eggs come from setuptools in 2004.
- Wheels were a formal PEP (PEP427) in 2012.
- Other differences:
  - Distribution vs Distribution/Runtime installation
  - No .pyc vs including .pyc files
  - richer file naming convention
  - versioning
  - sysconfig path type organization

Cool, so we use the default standard tool for packaging

Not so fast...

distutils limitations

- Project dependencies
- Configure files/plugins to be included (sdist)
- Entry points
- Windows command line executable at installation (instead of prebuild)
- Consistent behavior across Python versions

Good news This are features of setuptools.

setuptools: the recommended module

- fully-feature, actively-maintained, and stable!
- Package metadata
- Test hooks
- Platform specific details
- Python 3 support
- pip uses it by default

Good, that was easy

There were other attempts:

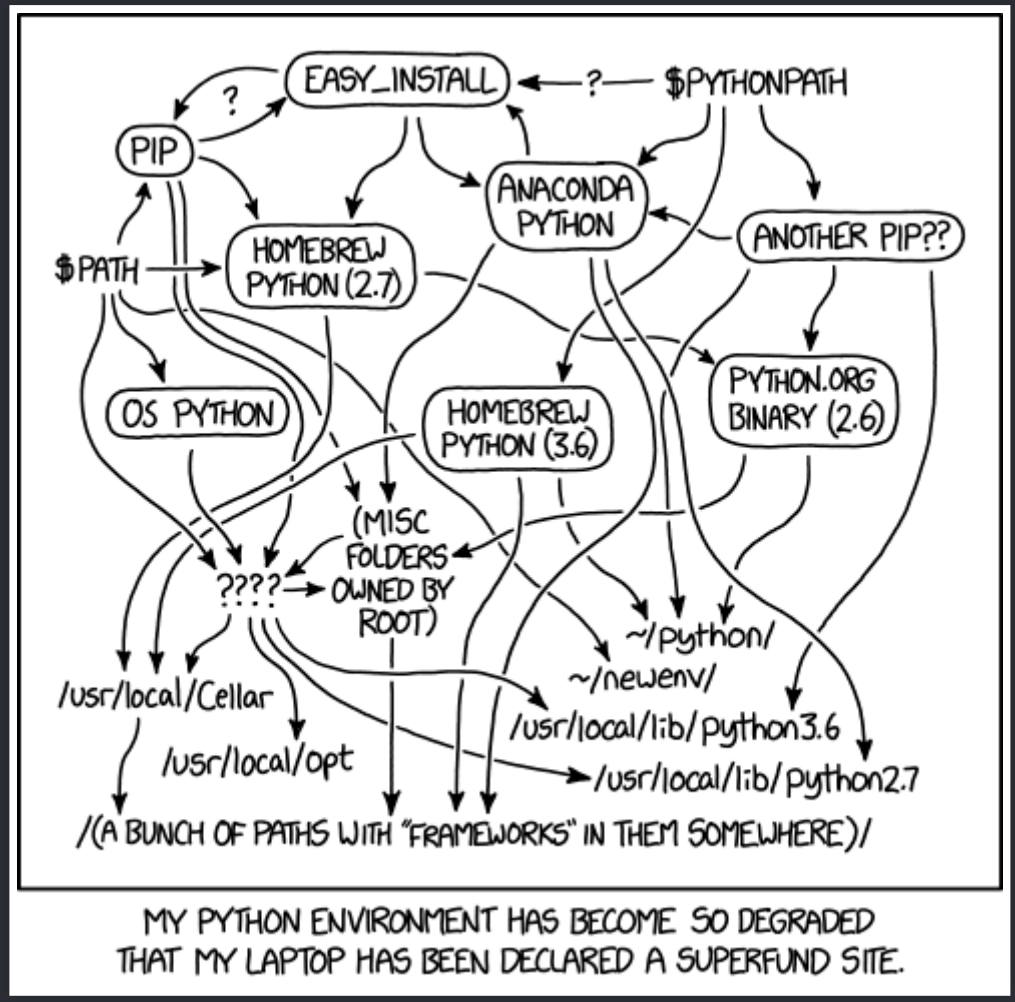
- distribute (fork of setuptools, merged back)
- distutils2 (distutils+setuptools+distribute)

OK, that's fair, so we stick with setuptools

There are alternatives:

- distlib
- bento
- buildout
- conda
- flit
- poetry
- ...

The problem is not only building



Python Environment - https://xkcd.com/1987/

What are the main problems with setuptools then?

- Missing features
- Difficult extension
- Entanglement with distutils/setuptools...PEP517

PEP517 - A build-system independent format for source trees

- Get rid of distutils-sig (gatekeeper for Python build systems)
- Easy build system for wheels and sdists.

Then we need a new style of source tree defined on a file...pyproject.toml

PEP518 - Specifying Minimum Build System Requirements for Python Projects

- setup.py, dependencies?
- What about setup_requires()?
  - no tooling can access the info besides setuptools
  - implicit installation (installing twice...)
  - a lot of workarounds to delay the dependencies imports
- Solution steps to produce a built artifact:
  - Source checkout, build system installation, build system execution.

TOML files

```
# This is a TOML document.
title = "TOML Example"

[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00-08:00 # First class dates

[database]
server = "192.168.1.1"
ports = [ 8001, 8001, 8002 ]
connection_max = 5000
enabled = true

[servers]
```

https://github.com/toml-lang/toml

An example

```
[build-system]
# Defined by PEP 518:
#requires = ["setuptools", "wheel"]
requires = ["flit"]
# Defined by this PEP:
build-backend = "flit.api:main"
```

What about all the backend interface?

```
def build_wheel(wheel_directory, config_settings=None, metadata_directory=None)
```

```
def build_sdist(sdist_directory, config_settings=None)
```

Optional

```
def prepare_metadata_for_build_wheel(...)
```

```
def get_requires_for_build_sdist(...)
```

```
. . .
```

Thanks, so we have yet another file?

- requirements.txt
- setup.py
- setup.cfg

What now?

- Keep maintaining your setuptools-based processes.
- Don't forget that pip will interact with the pyproject.toml
- Learn a bit about Flint and Poetry
- Don't be confused by other tools extending pyproject.toml
  - Black, coverage.py, towncrier, and tox.

Where do I sign?

```
[build-system]
requires = ["setuptools >= 40.6.0", "wheel"]
build-backend = "setuptools.build_meta"
```

Can I drop my setup.py then?

- Yes
  - And you keep your setup.cfg for build details
  - Although, you will not have editable installs (setuptools develop mode) pip install -e ...
- No, but!
  - Add a small one

```
import setuptools
if __name__ == "__main__":
    setuptools.setup()
```

Packaging Platypus

https://monotreme.club/

References

- https://www.python.org/dev/peps/pep-0517/
- https://www.python.org/dev/peps/pep-0518/
- https://packaging.python.org/key_projects/
- https://snarky.ca/what-the-heck-is-pyproject-toml/
- https://snarky.ca/clarifying-pep-518/
- https://caremad.io/posts/2013/07/setup-vs-requirement/
- https://discuss.python.org/t/the-packaging-platypus/1939

Building Python Projects

PEP 517 & PEP518 to the rescue

Dr. Cristián Maureira-Fredes
Software Engineer @ TQtC

@cmaureir