Rethinking Python Packaging

A thought experiment

@pradyunsg

Who?

Pradyun Gedam
@pradyunsg
pradyunsg.me

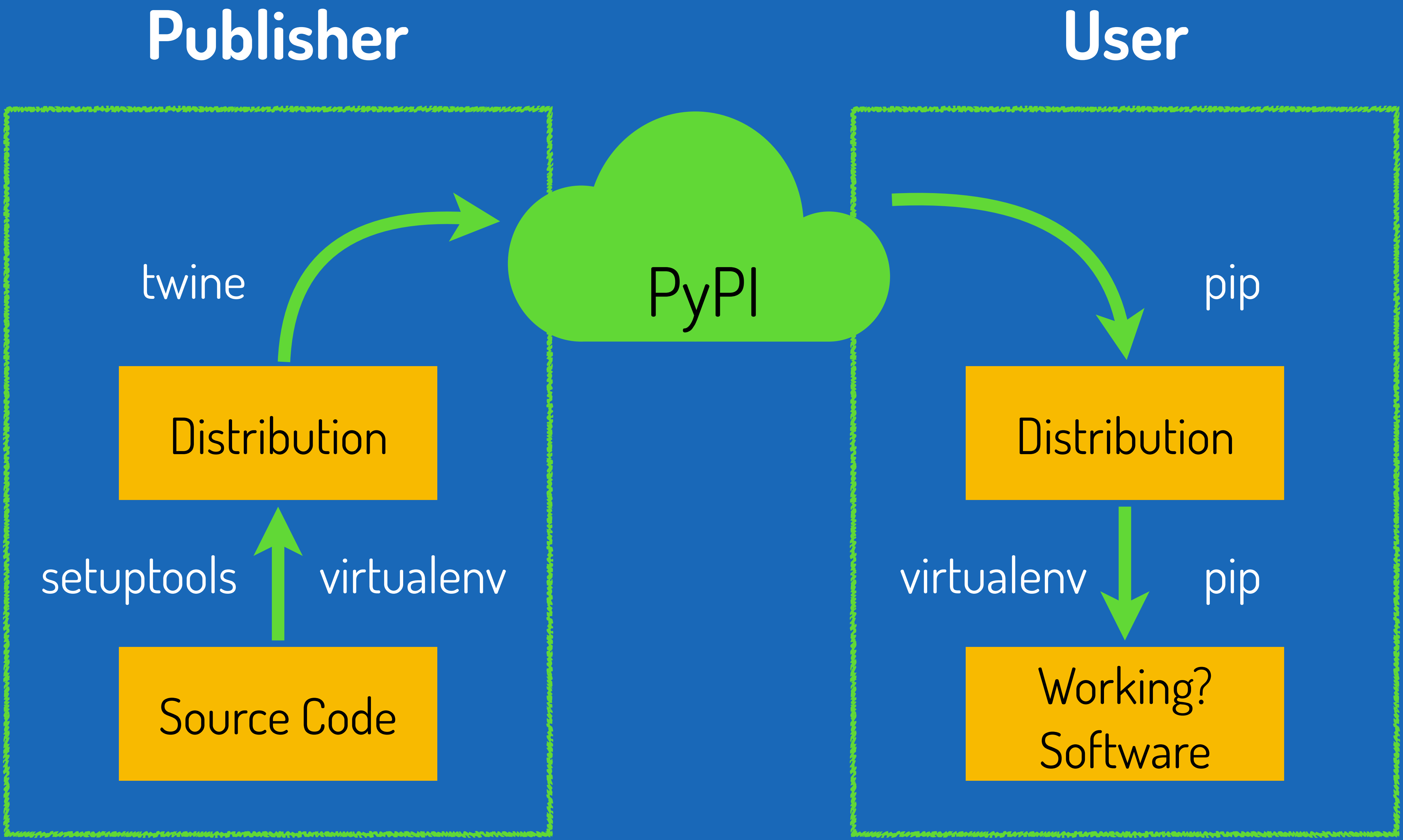
Member of
Python Packaging Authority

Maintainer of
pip, virtualenv, packaging and more

PSF Fellow

College student!

Rethinking
Python Packaging


Publisher
User
twine
PyPI
pip
Distribution
Distribution
setuptools
virtualenv
virtualenv
pip
Source Code
Working?
Software

Rethinking Python Packaging

Free to ignore existing tooling

Try to not break everything

Easier to understand, use and maintain

Constraints

- No removal of functionality for:
  - infrastructure — PyPI
  - published packages
  - existing PyPA standards

Disclaimers

complete speculation
(sort of)

other folks' ideas + my opinions

not a "UX person"

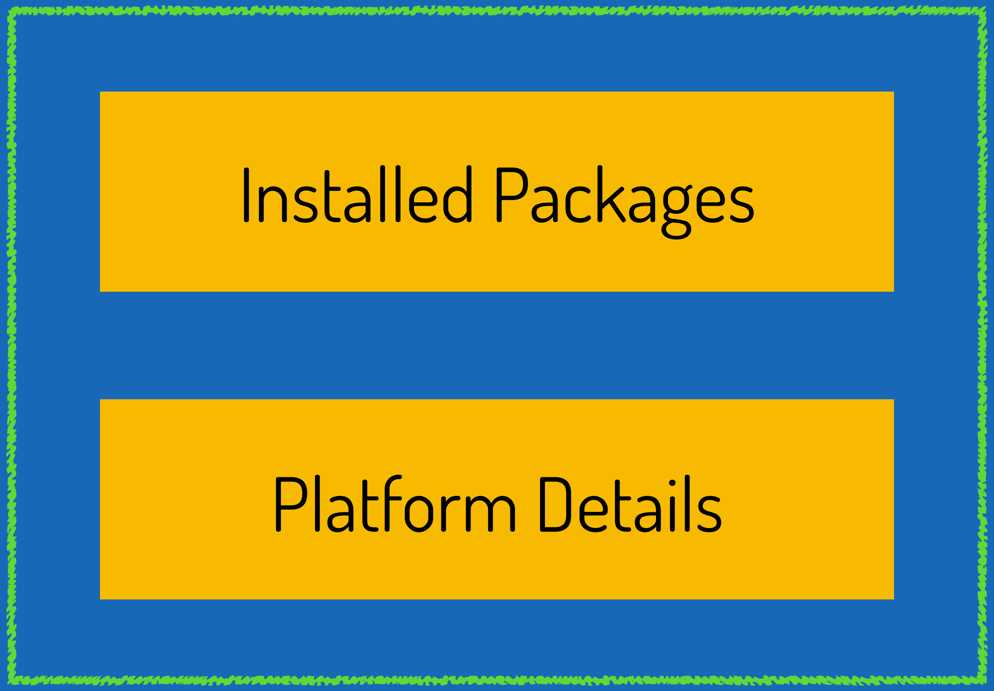
"Things"

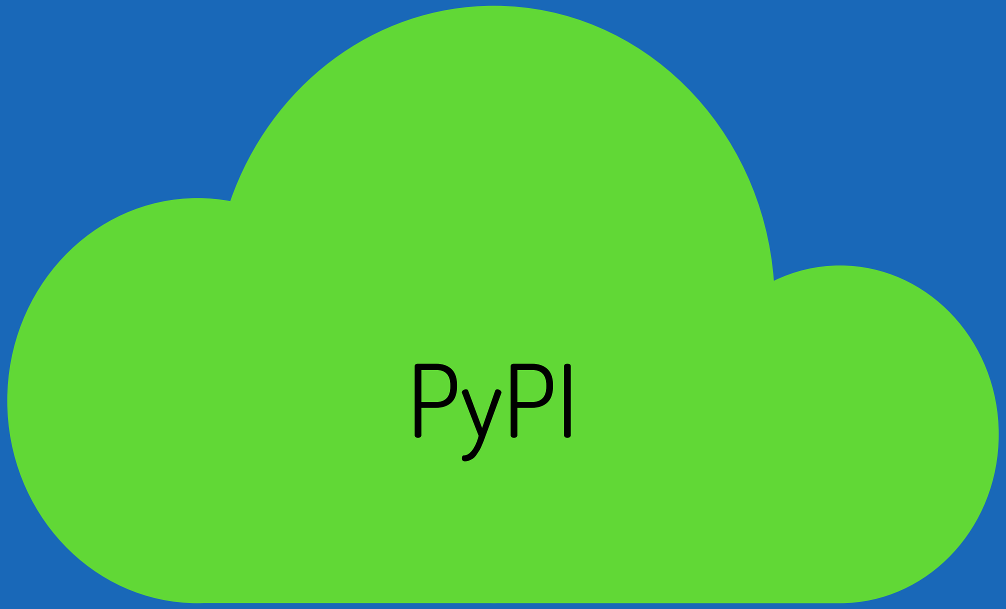
Source Tree

Source Distribution

Wheel Distribution

Environments



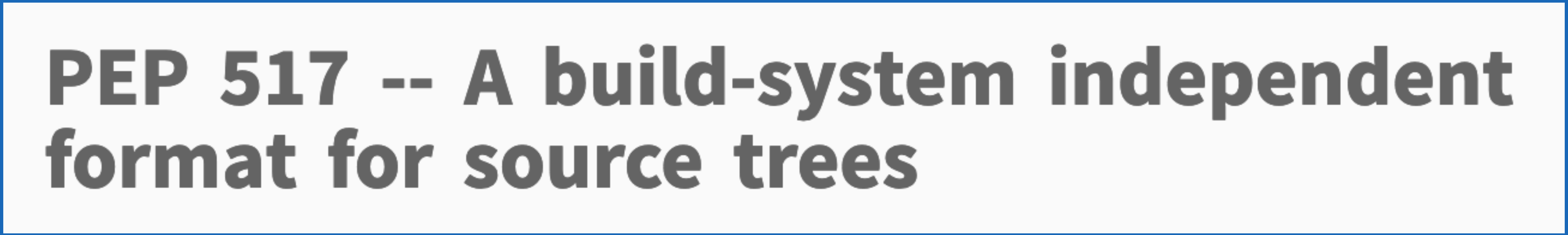

PyPI

Uploading Distributions

- Two Phase uploads would enable new workflows.
    - https://github.com/pypa/warehouse/issues/726

PEP 517's model

PEP 517's MODEL


PEP 517 -- A build-system independent format for source trees

Modern Packaging Tooling's model

Modern Packaging Tooling's model

- Build Backend — like setuptools, flit etc
  - Produce distributions from source trees

- Build Frontend — like pip
  - is user-facing
  - manages environment
  - handles distributions
  - orchestrates build backends

Backend Operations

- Produce distributions from source trees
  - Handling of Python code
  - Handling of non-Python code / artefacts
  - Metadata generation
  - Properly place files into a distribution

Properly place files into a distribution

- Idempotent source distributions
    - unpack + build-sdist → same as initial
- Build non-Python "stuff" only for wheels

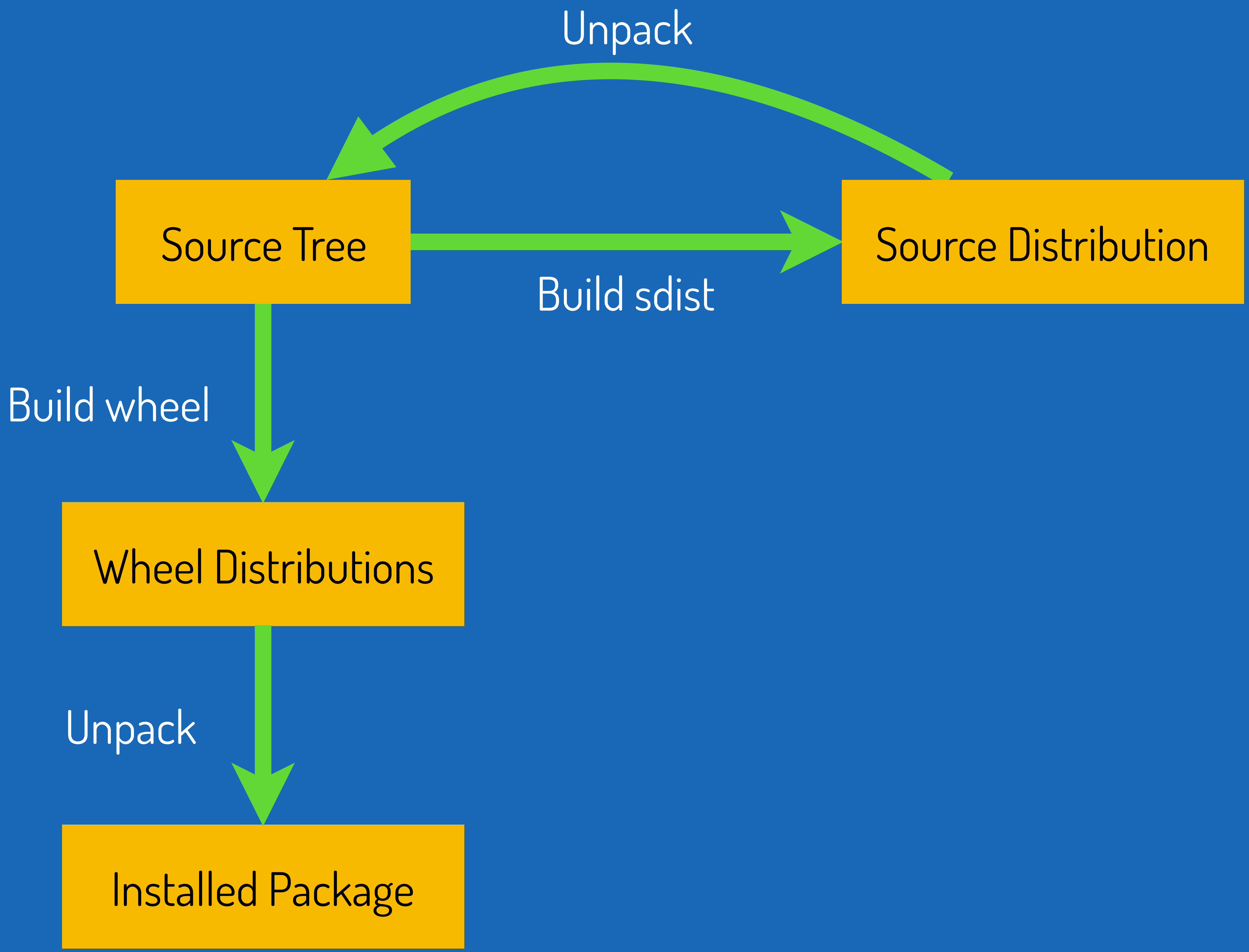
Metadata

- setuptools allows for arbitrary logic to generate metadata
- What we'd want
  - Static metadata, by default
  - Dynamic metadata, by opt-in
- Keep metadata in pyproject.toml
  - Allow specifying a script for enhancing it

Handling of non-Python code / artefacts

- setuptools invokes the compilers directly
    - complicated build process → complicated setup.py

- What we'd want
    - Generate build instructions for *other* build tools
    - Invoke the build tools


Unpack
Source Tree
Source Distribution
Build sdist
Build wheel
Wheel Distributions
Unpack
Installed Package

BUILD SDIST

- Load information from pyproject.toml
- Generate metadata
- Package relevant files into a .tar.gz file

BUILD WHEEL

- Load information from pyproject.toml
- Build non-Python code into binaries
- Generate metadata
- Package relevant files into a .whl file

Frontend Operations

- Environment management
  - Dependency resolution (hah!)
  - Installation / Uninstallation / Upgrade

- Handling distributions
  - Discovering and fetching distributions
  - Unpacking distributions correctly

- Orchestrates build backends
  - Manage source tree → distribution

Environment Management

- Binary Compatibility
- Reproducibility
- Simpler workflows