Python metapackages

Roberto Pastor Muela

Ansys

Introduction

Organizations experience problems when distributing multiple packages. What if you could easily distribute all your packages in one single package? Python *metapackages* are here to solve your problems!

The *metapackage* concept

Python *metapackages* are empty Python libraries that contain only a version attribute. However, they use a "dependencies" section to declare all libraries that are required for the installation. This trick can be used to install all the desired projects of a large community.

Example use case

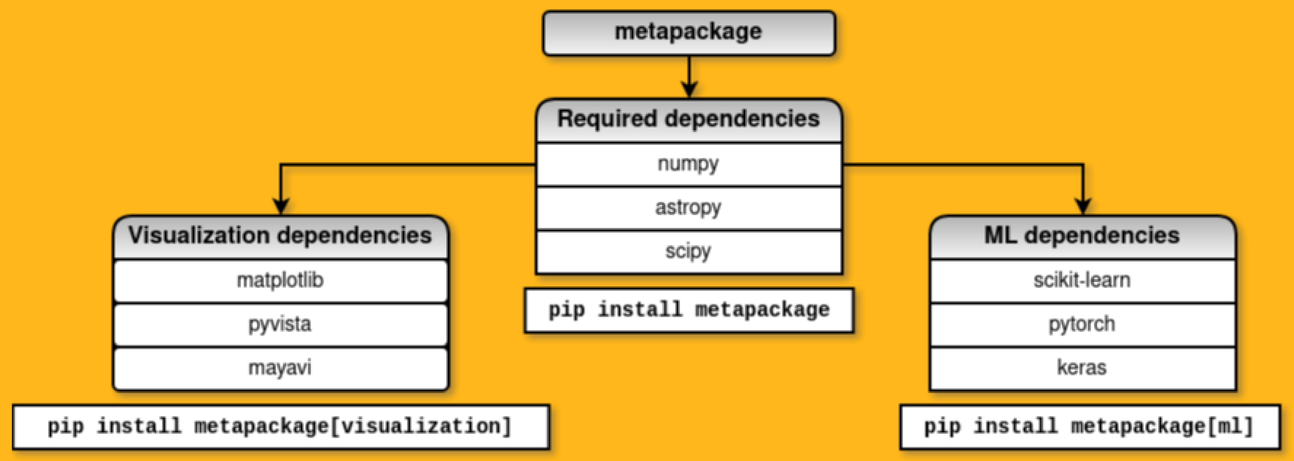
Say you have a Python metapackage called *my-package*. By installing it, users get your defined dependencies and also have access to additional targets you define. See the graph on the main section of this poster.

File structure

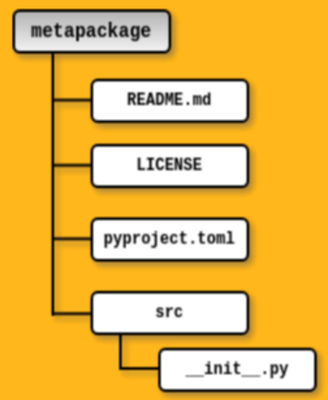
- The `src/<my-package>` folder has an `__init__.py` file that simply contains your metapackage version.
- A build system requirements file, `pyproject.toml`, `setup.py`), contains with your dependencies and extra targets. Dependency versions can be pinned down. For example, `numpy==1.21.0`) or flexible (`numpy`).

Metapackage structure

Required and extra dependencies groups



Metapackage layout




```toml
[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildap"

[project]
name = "metapackage"
version = "1.0.0"
description = "A demo metapackage for SciPyConf 2023"
readme = "README.md"
requires-python = ">=3.8,<4"
license = {file = "LICENSE"}
authors = [{name = "ANSYS, Inc.", email = "pyansys.core@ansys.com"}]
maintainers = [{name = "Ansys, Inc", email = "pyansys.core@ansys.com"}]
dependencies = ["astropy", "numpy", "scipy"]

[project.optional-dependencies]
ml = ["scikit-learn", "pytorch", "keras"]
visualization = ["matplotlib", "pyvista", "mayavi"]
```

Want to see an example repository?

✓Visit https://github.com/ansys/pyansys

Any questions?

✓Don't be shy and start the conversation!

Benefits of using a metapackage

- **One-stop shop**: All your Python packages are delivered together and are easily made available to end users.
- **Dependencies compatibility**: No incompatibility issues amongst dependencies can occur (when using CI/CD for building the package).
- **Easier install process**: Rather than installing each package individually, install all packages with only one installation command.
- **Multiple targets**: The metapackage may not only have *required* dependencies, it may also have extra targets (additional dependencies) for other purposes.
- **Pinned versions** (optional): Dependency updates sometimes lead to incompatibilities that users are not aware of. By having a metapackage that pins down your dependencies to a certain version, you make sure that for a given version your scripts are compatible with all the dependent libraries. This makes dependency handling much easier for end users.

PyAnsys

The PyAnsys project is a collection of Python packages that enable the use of Ansys products through Python.

Any questions?

Contact us at pyansys.core@ansys.com.

See our docs for more information on PyAnsys: https://docs.pyansys.com