Python metapackages

Roberto Pastor Muela

Ansys

Introduction

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

The *metapackage* concept

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

Example use case

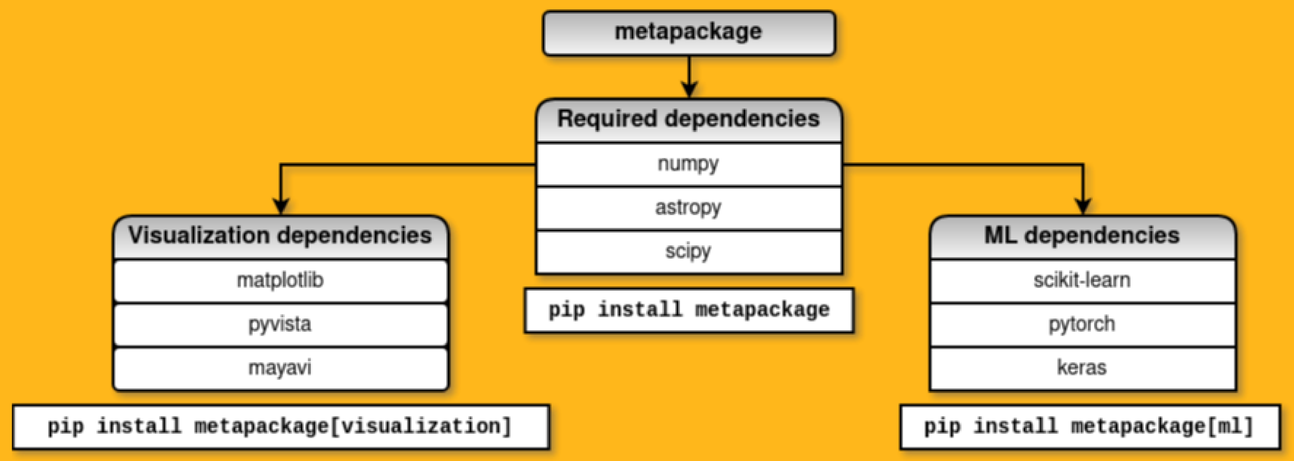
Say you have a Python metapackage called *my-package*. By installing it, users would get your defined dependencies, and also have access to additional targets you define. See the graph on the main section of this poster.

File structure

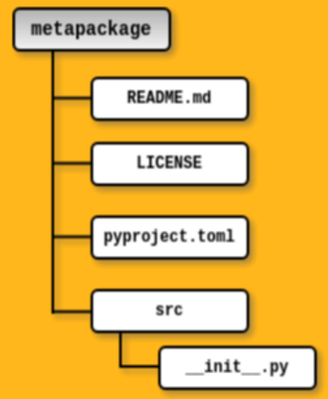
- A `src/<my-package>` folder with an `__init__.py` file. This file could simply contain your metapackage version.
- A build system requirements file (i.e. `pyproject.toml`, `setup.py`), with your dependencies and extra targets. Dependency versions can be pinned down (for example, `numpy==1.21.0`) or 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?

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

Any doubts?

✓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.
- **Ensure dependencies compatibility**: we ensure no incompatibility issues amongst its dependencies occur (by using CI/CD for building the package).
- **Easier install process**: rather than installing each package individually, getting all of them together with one install command makes your life easier.
- **Multiple targets**: the metapackage does not only need to have *required* dependencies. It may also have extra targets (i.e. additional dependencies) for other purposes!
- **Pinned versions** (optional): dependency updates sometimes leads to incompatibilities which 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 doubts?

Contact us at pyansys.core@ansys.com!

Check our docs for more information on PyAnsys!

https://docs.pyansys.com