Packaging python programs using pip

Why do we need packaging

1. Python is a general purpose language which is used in a variety of applications: games, web sites, industrial robots, stc.
2. Not every library can be bundled together with the python standard library, hence we need a standardised way to share, download, and install useful python packages.

Various ways of distributing python packages #1

1. **Python Modules**: If the script is small, depends on only the standard library, and the target python version is known, the file modules can be used directly.
2. However, this won't scale for projects that consist of multiple files, need additional libraries, or need a specific version of Python.

Various ways of distributing python packages #2

1. **Source distributions**: If the code contains only python code, and we know your deployment environment supports the corresponding Python version, then we can use Python's native packaging tools to create a *source* distribution package, or *sdist* for short.
2. *sdists* are compressed archives (.tar.gz files) containing one or more packages or modules.
3. However, this pattern won't scale for projects that consist of multiple files of various languages.

Various ways of distributing python packages #3

1. **Binary distributions**: These can be used when our code contains C, C++, python etc.
2. Python created the Wheel, a package format designed to ship libraries with compiled artifacts. In fact, Python's package installer, pip, always prefers wheels because installation is always faster.
3. A Python .whl file is a built distribution which is essentially a ZIP (.zip) archive, such that it comes in a ready-to-install format and allows us to skip the build stage required with source distributions.
4. It is a good idea to have both source and binary distributions available for a project, as by uploading sdist, you are allowing developers to build it for themselves.

How to publish a python package to PyPI

1. **PyPI (/ˌpaɪpiˈaɪ/)**, also known as the “Cheese Shop” primarily hosts Python packages in the form of archives: sdists (source distributions) or precompiled wheels.
2. Here we will learn to:
   a. Prepare out Python package for publication
   b. Think about versioning
   c. Upload your package to PyPI

Versioning the package

Semantic versioning is a good default scheme to use. The version number is given as three numerical components, for instance 0.1.2. The components are called MAJOR, MINOR, and PATCH, and there are simple rules about when to increment each component:

- Increment the MAJOR version when you make incompatible API changes.
- Increment the MINOR version when you add functionality in a backwards-compatible manner.
- Increment the PATCH version when you make backwards-compatible bug fixes.

How to publish a python package to PyPI

1. Create a project with valid setup.py files
2. pip install twine
3. python setup.py sdist bdist_wheel
4. twine check dist/*
5. twine upload dist/*

Important Links:

1. An Overview of Packaging for Python: https://packaging.python.org/overview/
2. Python Wheels: https://realpython.com/python-wheels/#python-packaging-made-better-an-intro-to-python-wheels
3. How to publish to pypi:
   https://realpython.com/pypi-publish-python-package/#preparing-your-package-for-publication
4. How to publish to pypi 2: https://github.com/MichaelKim0407/tutorial-pip-package
5. Using anaconda and pip: https://www.anaconda.com/blog/understanding-conda-and-pip
6. For testing pip uploads: https://test.pypi.org/
7. For final pip uploads: https://pypi.org/
8. More about pip: https://realpython.com/what-is-pip/

Thank You