Packaging Python Projects

This tutorial walks you through how to package a simple Python project. It will show you how to add the necessary files and structure to create the package, how to build the package, and how to upload it to the Python Package Index.

A simple project

This tutorial uses a simple project named `example_pkg`. If you are unfamiliar with Python's modules and import packages, take a few minutes to read over the Python documentation for packages and modules. Even if you already have a project that you want to package up, we recommend following this tutorial as-is using this example package and then trying with your own package.

To create this project locally, create the following file structure:

```
packaging_tutorial
└── example_pkg
    └── __init__.py
```

Once you create this structure, you'll want to run all of the commands in this tutorial within the top-level folder - so be sure to `cd packaging_tutorial`.

`example_pkg/__init__.py` is required to import the directory as a package, and can simply be an empty file.

Creating the package files

You will now create a handful of files to package up this project and prepare it for distribution. Create the new files listed below and place them in the project's root directory - you will add content to them in the following steps.

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Creating a test folder

`tests/` is a placeholder for unit test files. Leave it empty for now.

Creating setup.py

`setup.py` is the build script for setuptools. It tells setuptools about your package (such as the name and version) as well as which code files to include.

Open `setup.py` and enter the following content. Update the package name to include your username (for example, `example-pkg-theacodes`), this ensures that you have a unique package name and that your package doesn't conflict with packages uploaded by other people following this tutorial.

```
import setuptools

with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()

setuptools.setup(
    name="example-pkg-YOUR-USERNAME-HERE", # Replace with your own username
    version="0.0.1",
    author="Example Author",
    author_email="author@example.com",
    description="A small example package",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/pypa/sampleproject",
    packages=setuptools.find_packages(),
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires='>=3.6',
)
```

`setup()` takes several arguments. This example package uses a relatively minimal set:

- `name` is the *distribution name* of your package. This can be any name as long as only contains letters, numbers, `_` , and `-`. It also must not already be taken on pypi.org. **Be sure to update this with your username,** as this ensures you won't try to upload a package with the same name as one which already exists when you upload the package.
- `version` is the package version see **PEP 440** for more details on versions.
- `author` and `author_email` are used to identify the author of the package.
- `description` is a short, one-sentence summary of the package.
- `long_description` is a detailed description of the package. This is shown on the package detail page on the Python Package Index. In this case, the long description is loaded from `README.md` which is a common pattern.
- `long_description_content_type` tells the index what type of markup is used for the long description. In this case, it's Markdown.

- `url` is the URL for the homepage of the project. For many projects, this will just be a link to GitHub, GitLab, Bitbucket, or similar code hosting service.
- `packages` is a list of all Python import packages that should be included in the Distribution Package. Instead of listing each package manually, we can use `find_packages()` to automatically discover all packages and subpackages. In this case, the list of packages will be `example_pkg` as that's the only package present.
- `classifiers` gives the index and pip some additional metadata about your package. In this case, the package is only compatible with Python 3, is licensed under the MIT license, and is OS-independent. You should always include at least which version(s) of Python your package works on, which license your package is available under, and which operating systems your package will work on. For a complete list of classifiers, see https://pypi.org/classifiers/.

There are many more than the ones mentioned here. See Packaging and distributing projects for more details.

Creating README.md

Open `README.md` and enter the following content. You can customize this if you'd like.

```
# Example Package

This is a simple example package. You can use
[Github-flavored Markdown](https://guides.github.com/features/mastering-markdown/)
to write your content.
```

Creating a LICENSE

It's important for every package uploaded to the Python Package Index to include a license. This tells users who install your package the terms under which they can use your package. For help picking a license, see https://choosealicense.com/. Once you have chosen a license, open `LICENSE` and enter the license text. For example, if you had chosen the MIT license:

```
Copyright (c) 2018 The Python Packaging Authority

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Generating distribution archives

The next step is to generate distribution packages for the package. These are archives that are uploaded to the Package Index and can be installed by pip.

Make sure you have the latest versions of `setuptools` and `wheel` installed:

```
python3 -m pip install --user --upgrade setuptools wheel
```

Tip: IF you have trouble installing these, see the Installing Packages tutorial.

Now run this command from the same directory where `setup.py` is located:

```
python3 setup.py sdist bdist_wheel
```

This command should output a lot of text and once completed should generate two files in the `dist` directory:

```
dist/
  example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
  example_pkg_YOUR_USERNAME_HERE-0.0.1.tar.gz
```

Note: If you run into trouble here, please copy the output and file an issue over on packaging problems and we'll do our best to help you!

The `tar.gz` file is a Source Archive whereas the `.whl` file is a Built Distribution. Newer pip versions preferentially install built distributions, but will fall back to source archives if needed. You should always upload a source archive and provide built archives for the platforms your project is compatible with. In this case, our example package is compatible with Python on any platform so only one built distribution is needed.

Uploading the distribution archives

Finally, it's time to upload your package to the Python Package Index!

The first thing you'll need to do is register an account on `Test PyPI`. Test PyPI is a separate instance of the package index intended for testing and experimentation. It's great for things like this tutorial where we don't necessarily want to upload to the real index. To register an account, go to https://test.pypi.org/account/register/ and complete the steps on that page. You will also need to verify your email address before you're able to upload any packages. For more details on Test PyPI, see Using TestPyPI.

Now you'll create a PyPI API token so you will be able to securely upload your project.

Go to https://test.pypi.org/manage/account/#api-tokens and create a new API token; don't limit its scope to a particular project, since you are creating a new project.

Don't close the page until you have copied and saved the token — you won't see that token again.

Now that you are registered, you can use twine to upload the distribution packages. You'll need to install Twine:

```
python3 -m pip install --user --upgrade twine
```

Once installed, run Twine to upload all of the archives under `dist`:

```
python3 -m twine upload --repository testpypi dist/*
```

You will be prompted for a username and password. For the username, use `__token__`. For the password, use the token value, including the `pypi-` prefix.

After the command completes, you should see output similar to this:

```
Uploading distributions to https://test.pypi.org/legacy/
Enter your username: [your username]
Enter your password:
Uploading example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
100%|████████████████████████████████| 4.65k/4.65k [00:01<00:00, 2.88kB/s]
Uploading example_pkg_YOUR_USERNAME_HERE-0.0.1.tar.gz
100%|████████████████████████████████| 4.25k/4.25k [00:01<00:00, 3.05kB/s]
```

Once uploaded your package should be viewable on TestPyPI, for example, https://test.pypi.org/project/example-pkg-YOUR-USERNAME-HERE

Installing your newly uploaded package

You can use pip to install your package and verify that it works. Create a new virtualenv (see Installing Packages for detailed instructions) and install your package from TestPyP


```
python3 -m pip install --index-url https://test.pypi.org/simple/ --no-deps example-pk
```

Make sure to specify your username in the package name!

pip should install the package from Test PyPI and the output should look something like this:

```
Collecting example-pkg-YOUR-USERNAME-HERE
  Downloading https://test-files.pythonhosted.org/packages/.../example-pkg-YOUR-USERN
Installing collected packages: example-pkg-YOUR-USERNAME-HERE
Successfully installed example-pkg-YOUR-USERNAME-HERE-0.0.1
```

Note: This example uses `--index-url` flag to specify TestPyPI instead of live PyPI. Additionally, it specifies `--no-deps`. Since TestPyPI doesn't have the same packages as the live PyPI, it's possible that attempting to install dependencies may fail or install something unexpected. While our example package doesn't have any dependencies, it's a good practice to avoid installing dependencies when using TestPyPI.

You can test that it was installed correctly by importing the package. Run the Python interpreter (make sure you're still in your virtualenv):

```
python
```

and from the interpreter shell import the package:

```
>>> import example_pkg
```

Note that the Import Package is `example_pkg` regardless of what name you gave your Distribution Package in `setup.py` (in this case, `example-pkg-YOUR-USERNAME-HERE`).

Next steps

Congratulations, you've packaged and distributed a Python project! ✨ 📦 ✨

Keep in mind that this tutorial showed you how to upload your package to Test PyPI, which isn't a permanent storage. The Test system occasionally deletes packages and accounts. It is best to use Test PyPI for testing and experiments like this tutorial.

When you are ready to upload a real package to the Python Package Index you can do much the same as you did in this tutorial, but with these important differences:

- Choose a memorable and unique name for your package. You don't have to append your username as you did in the tutorial.
- Register an account on https://pypi.org - note that these are two separate se[illegible] login details from the test server are not shared with the main server.

- Use `twine upload dist/*` to upload your package and enter your credentials for the account you registered on the real PyPI. Now that you're uploading the package in production, you don't need to specify `--repository`; the package will upload to https://pypi.org/ by default.
- Install your package from the real PyPI using `pip install [your-package]`.

At this point if you want to read more on packaging Python libraries here are some things you can do:

- Read more about using setuptools to package libraries in Packaging and distributing projects.
- Read about Packaging binary extensions.
- Consider alternatives to setuptools such as flit, hatch, and poetry.