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.

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

```
/example_pkg
  /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 example_pkg`.

You should also edit `example_pkg/__init__.py` and put the following code in there:

```
name = "example_pkg"
```

This is just so that you can verify that it installed correctly later in this tutorial.

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 - you will add content to them in the following steps.

```
/example_pkg
  /example_pkg
    __init__.py
  setup.py
  LICENSE
  README.md
```

v: latest

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, you can personalize the values if you want:

```
import setuptools

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

setuptools.setup(
    name="example_pkg",
    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",
    ],
)
```

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

- `name` is the name of your package. This can be any name as long as only contains letters, numbers, `_` , and `-`. It also must not already taken on pypi.org.
- `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 package 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 pro[illegible] v: latest [illegible]l 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` tell 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/ma
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 obta
of this software and associated documentation files (the "Softwa
in the Software without restriction, including without limitatio
to use, copy, modify, merge, publish, distribute, sublicense, ar
copies of the Software, and to permit persons to whom the Softwa
furnished to do so, subject to the following conditio

The above copyright notice and this permission notice shall be i
```

```
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANT
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVEN
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DE
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-0.0.1-py3-none-any.whl
  example_pkg-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.

v: latest

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 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`:

```
twine upload --repository-url https://test.pypi.org/legacy/ dist
```

You will be prompted for the username and password you registered with Test PyPI. 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-0.0.1-py3-none-any.whl
100%|█████████████████████| 4.65k/4.65k [00:01<00:00, 2.88kB/s]
Uploading example_pkg-0.0.1.tar.gz
100%|█████████████████████| 4.25k/4.25k [00:01<00:00, 3.05kB/s]
```

> **Note:** If you get an error that says `The user '[your username]' isn't allowed to upload to project 'example-pkg'`, you’ll need to go and pick a unique name for your package. A good choice is `example_pkg_your_username`. Update the `name` argument in `setup.py`, remove the `dist` folder, and regenerate the archives.

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

v: latest

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 TestPyPI:

```
python3 -m pip install --index-url https://test.pypi.org/simple/
```

Note: If you used a different package name in the preview step, replace `example_pkg` in the command above with your package name.

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

```
Collecting example_pkg
  Downloading https://test-files.pythonhosted.org/packages/.../e
Installing collected packages: example-pkg
Successfully installed example-pkg-0.0.1
```

You can test that it was installed correctly by importing the module and referencing the `name` property you put in `__init__.py` earlier.

Run the Python interpreter (make sure you're still in your virtualenv):

```
python
```

And then import the module and print out the `name` property. This should be the same regardless of what you name you gave your distribution package in `setup.py` because your import package is `example_pkg`.

```
>>> import example_pkg
>>> example_pkg.name
'example_pkg'
```

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 and Test PyPI is ephemeral. It's not unusual for packages and accounts to be deleted occasionally. If you want to upload your package to the real Python Package Index you can do it by registering an account on https://pypi.org and following the same instructions, however, use `twine upload dist/*` to upload [illegible] and enter your credentials for the account you registered on the real PyPI. You can 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.