

CREATING A PYTHON PACKAGE

Author: G00Z-G00Z

Mail: edygomezgg4@gmail.com

Date: 2023-07-18

WHY DO YOU NEED YOUR CODE TO BE ?

- When you are **reusing** a lot of the same functions/classes
- When you want split your code into specific packages that are **independent** from each other
- **When you want to share your code with others,** the best way is to package your code

BENEFITS OF PACKAGING



- Packages can be **independent** from each other. Allowing you to test and distribute code easily and allows independent development from other parts of the code
- Can be managed by `pip` as any other dependency
- **Reduces code duplication**
- Can be labeled with **versions**, so that they can be easily identified and versioned in the code base

CREATING A PACKAGE →
INSTALLING A PACKAGE ↓



INSTALLATION TYPES



Install locally

```
$ pip install -e <cloned_repo_dir_in_computer>
```



Install from **Github**

```
$ pip install git+<clone_url>
```



Install from **PyPI** and **Test PyPI**

```
$ pip install <package-name>
```



HOW TO CREATE A PACKAGE?

Watch this video explaining how to create python packages

Both the video and this section is based on [ArjanCodes's Youtube video](#).










EXAMPLE PACKAGE REPO

Clone this [repository](#) if you want the structure of a python package



HOW TO CREATE A PACKAGE



1.  Structure folder
2.  Write some code
3.  Test it
4.  Document it
5.  Write `setup.py`
6.  Build it
7.  Publish it



STRUCTURE YOUR FOLDERS

```
- README.md
- setup.py
- package_name
  - README.md
  - __init__.py
  - tests/
    - __init__.py
    - test_function_1.py
  - sub_module_1
    - __init__.py
    - function_1.py
    - ...
  - sub_module_2
    - __init__.py
    - function_2.py
```

In a directory, you must include this files as an example. The folder with the `package_name` is the package your are going to publish. Inside of it there're the functions and tests.

Remember to always add `__init__.py` inside each folder in the package



WRITE CODE INSIDE THE PACKAGE & TEST IT !

Try to write *modular*. This will improve the quality of the code and allow the clients of the package to have a nicer experience

Write *tests* for your code that test the *functionality* of your code. Even though it is not critical at the moment, the future developer (or the future you) will definitely be grateful



DOCUMENT THE CODE

Document the functions by adding *descriptions* and *type hints*, and fill in the `README.md` inside the package (will use it later in the `setup.py`).

Optionally, generate documents or websites with information on how to use your package



SETUP.PY WITH SETUPTOOLS

Setuptools is a collection of tools and utilities that facilitate the packaging, distribution, and installation of Python packages. It simplifies the process of distributing software and its dependencies, making it easier for developers to share their code with others

To install it:

```
$ pip install setuptools wheel
```

MAKE YOUR SETUP.PY

This file tells python how to install the package in your system. It has *metadata, requirements, tags, descriptions ...* It is very important to fill it correctly

You can fill up the *name of the package, version, requirements, testing parameters, metadata, categories, tags, descriptions etc*

This basically tells `pip` and other developers how your package is intended to be installed

Example of a `setup.py` from the [example repository](#)

```
from setuptools import find_packages, setup

with open("./package_name/README.md", "r") as f:
    long_description = f.read()

setup(
    name="monkey_lab_example_lib",
    packages=find_packages(include=["monkey_lab_example_lib"]),
    version="0.0.1",
    description="Monkey Lab example library for understanding h",
    long_description=long_description,
    long_description_content_type="text/markdown",
    author="g00z-g00z",
    license="MIT",
    install_requires=[].
```



BUILD YOUR PROJECT

To build your project run the following commands:

```
$ cd <root_directory_of_repo>  
$ python setup.py bdist_wheel sdist
```

This will generate the `dist/` folder with your package information ready to be used!

PUBLISH YOUR PACKAGE



USING PYPI →



USING GITHUB ↓

REPOSITORY IN GITHUB

If you followed the structure from the [video](#) or the [repository](#) you can upload the repo to GitHub and installed directly from there with the commands listed in [here](#)

The advantage of this is that the repository can be private and you do not need to build your project first.

PYPI

PyPI This is the official site for python packages. After installing it here, anyone can install it using `pip`

TEST PYPI

Test PyPI is a website where you can publish packages as prototypes, without committing it to **PyPI**

TWINE

This tool will allow you to publish your package to [PyPI](#) from the command line.

```
$ pip install twine
```

CREATE AN ACCOUNT

For uploading packages to [PyPI](#), you must create account in the following websites:

- [PyPI](#): This is the official site for python packages!
Must be used only for the release versions
- [Test PyPI](#) : This is where you can upload your packages without affecting the search indexes
(great for prototyping)

You must have your credentials to upload it

TO TEST PYPI

```
$ twine upload -r testpypi dist/*
```

Here you can test your package to see if it works before
uploading it to [PyPI](#)

TO PYPI

```
$ twine upload dist/*
```

Look on how to use the package [here](#)

Example on how it looks in [Test PyPI example library](#)

HAPPY CODING!

