**Step 1: Get the python scripts ready**

First step is, of course, getting your python program (which you want to publish on PyPI) ready! It could be any python script.

**Step 2: Getting the package-directory structure ready**

This is the most important step. Now, we have to follow some pre-defined structure for our package’s directory.

The directory structure has to be like this :

Lightbox

Here is how my setup.py looks like:

from setuptools import setup

# reading long description from file

with open('DESCRIPTION.txt') as file:

long\_description = file.read()

# specify requirements of your package here

REQUIREMENTS = ['requests']

# some more details

CLASSIFIERS = [

'Development Status :: 4 - Beta',

'Intended Audience :: Developers',

'Topic :: Internet',

'License :: OSI Approved :: MIT License',

'Programming Language :: Python',

'Programming Language :: Python :: 2',

'Programming Language :: Python :: 2.6',

'Programming Language :: Python :: 2.7',

'Programming Language :: Python :: 3',

'Programming Language :: Python :: 3.3',

'Programming Language :: Python :: 3.4',

'Programming Language :: Python :: 3.5',

]

# calling the setup function

setup(name='mygmap',

version='1.0.0',

description='A small wrapper around google maps api',

long\_description=long\_description,

url='https://github.com/nikhilkumarsingh/mygmap',

author='Nikhil Kumar Singh',

author\_email='nikhilksingh97@gmail.com',

license='MIT',

packages=['geo'],

classifiers=CLASSIFIERS,

install\_requires=REQUIREMENTS,

keywords='maps location address'

)

**setup.py** : It is the most important file. It’s the file where various aspects of your project are configured. The primary feature of setup.py is that it contains a global setup() function. The keyword arguments to this function are how specific details of your project are defined.

**name:** It is the name of your project. Your package will be listed by this name on PyPI.

**version:** It is a string in which you can specify the current version of your project.

It is totally your choice how you want to set the scheme of the series of versions (You may use ‘1.0’ or ‘0.1’ or even ‘0.0.1’).

This version is displayed on PyPI for each release if you publish your project. Every-time you upload a new version, you will have to change this argument as well.

**description:** A short description about the package. You can use long\_description argument to write long descriptions.

**long\_description**: We can use rich text for better description of our files. The default file format is reStructuredText . You can have a look at DESCRIPTION.txt to get a feel of the syntax.

**url:** A homepage URL for your project. This makes it easier for people to follow or contribute to your project.

**author, author\_email:** Details about the author

**license:** specify the type of license you are using.

**classifiers:** It is a list of strings in which we can specify more details about our project like its development status, topic, license and supported python versions for your project. You can see more classifiers here.

**install\_requires:** It can be used to specify what third-party libraries your package needs to run. These dependencies will be installed by pip when someone installs your package.

**keywords:** List keywords to describe your project.

**DESCRIPTION.txt :** This file contains the long description about our package to show on the PyPI page. We use reStructuredText file format here. Check the file used in our package here.

**LICENSE.txt**: It is a good practice to set a license for the usage of your project. You can use any of the freely available templates. Most commonly used is the MIT license.

The license I am using for this project is available here.(You may replace the Author’s name for using this license in your projects)

**README.md:** This file has got nothing to do with our PyPI package. It contains description to be shown on the Github page. You can use it for PyPI page too but it will need some more modifications to our code. For now, lets keep it simple.

**\_\_init\_\_.py:** The primary use of \_\_init\_\_.py is to initialize a python package.

The inclusion of this file in a directory indicates to the Python interpreter that the directory should be treated like a Python package. You can leave this file empty.

**Step 3:** Create your accounts

Now, its time to create an account on PyPI and Test PyPI. Test PyPI is just a testing site where we will upload our code first to see if everything works correctly or not.

Once accounts have been made, create this .pypirc file in home directory of your system and enter the account details.

[distutils]

index-servers =

pypi

pypitest

[pypi]

repository=https://pypi.python.org/pypi

username= your\_username

password= your\_password

[pypitest]

repository=https://testpypi.python.org/pypi

username= your\_username

password= your\_password

**Note:** If you are on a windows system, just type echo %USERPROFILE% in command prompt to know the home directory of your PC. Put the .pypirc file there.

**Step 4: Upload the package**

Finally, we are ready to upload our package on PyPI!

First of all, we will check if our package installs correctly on Test PyPI.

Open command prompt/terminal in the root directory of your package.

Run this in terminal:

**python setup.py register -r pypitest**

This will attempt to register your package against PyPI’s test server, just to make sure you’ve set up everything correctly.

Now, run this:

**python setup.py sdist upload -r pypitest**

You should get no errors, and should also now be able to see your library in the test PyPI repository.

Once you’ve successfully uploaded to PyPI Test, perform the same steps but point to the live PyPI server instead.

To register on PyPI, run:

**python setup.py register -r pypi**

Then, run:

**python setup.py sdist upload -r pypi**

And you’re all done! Your package is now publicly available on PyPI and could be easily installed by a simple pip command!

The package we created using this tutorial is available here.

Just type on terminal,

**pip install your\_package\_name**

to check if installation process is getting completed successfully.