**CREATING A PYTHON PACKAGE FROM YOUR CODE**

**Prerequisites:**

1. PyPi account
2. Latest version of setuptools, wheel, twine installed (This can be done using pip install)

**Steps:**

1. Get the code ready
   1. Some optional but recommended steps:
      1. Remove all print statement from the code. If you want to inform or log something, use logging.
      2. Remove all code that stays outside of a class or function. Such code (if really necessary), put it under \_\_main\_\_ function
   2. Put all required files and folders into a folder with the name of the package.

Add a \_\_init\_\_.py file to the folder. This will mark it as a Python package.

Add the \_\_init\_\_.py file to any other subfolder in the package which contains Python files that need to be referenced in the code.

The \_\_init\_\_.py files will be as below

from . import <python file name>

…

\_\_all\_\_ = [

'<python file name>',

…

]

* 1. Add the below files outside the main package folder. These are necessary to upload the package to PyPi.
     1. LICENSE file.

The License file contains the copyright and permission notice.

The content of the MIT license can be taken from <https://opensource.org/licenses/MIT>

* + 1. README.md file. (optional but recommended)

The Readme file contains information about the python package.

* + 1. HISTORY.md file. (optional but recommended)

The History file contains information about previous versions of the package.

* + 1. setup.py

The setup.py file contains information about the package that PyPi needs, like its name, a description, the current version etc.

The setup.py file will be as below:

from setuptools import setup, find\_packages

with open('README.md') as readme\_file:

README = readme\_file.read()

with open('HISTORY.md') as history\_file:

HISTORY = history\_file.read()

setup\_args = dict(

name='<package name>',

version='<package version>',

description='<short description>',

long\_description\_content\_type="text/markdown",

long\_description=README + '\n\n' + HISTORY,

license='MIT',

packages=find\_packages(),

author='<author name>',

author\_email='<author email>',

keywords=[<comma separated list of keywords>],

url='<github url of the package>',

download\_url='https://pypi.org/project/<package name>',

)

install\_requires = [

<comma separated list of dependent packages to be installed with your package>

]

if \_\_name\_\_ == '\_\_main\_\_':

setup(\*\*setup\_args, install\_requires=install\_requires, include\_package\_data=True)

Add the names of the other packages that were needed to be installed for your code in the install\_requires field.

Add include\_package\_data=True to setup(…)to include data files (templates, config etc)

Create a MANIFEST.in in the same location as the setup.py and list all the data files to be included as below

include <folder name>\<file name>

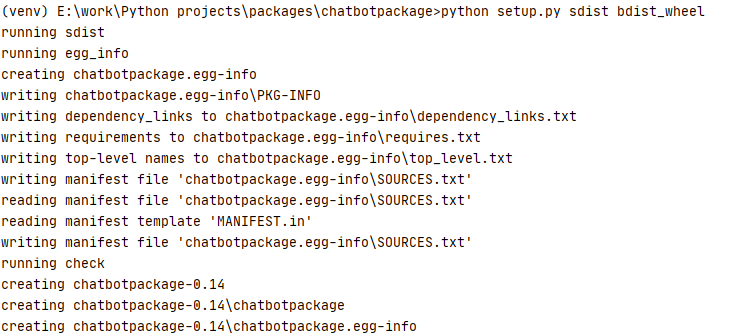
To include all subfolders replace include with recursive-include.

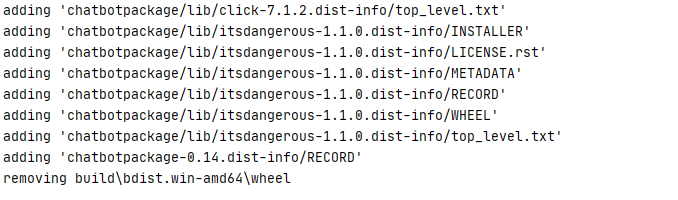
* 1. Generate distribution archives and upload to PyPi.

To generate distribution archives run the below command from the same directory where setup.py is located

python setup.py sdist bdist\_wheel

This command will produce output like below and once completed will generate a dist folder with a tar and wheel file in it, and also a build and <package name>.egg-info folder





To upload all the archives to PyPi run the below command from the same directory as the dist directory.

twine upload dist/\*

This will prompt for your PyPi username and password.

After successfully uploading the url to view you package will be given in the output

Your Python package can now be installed using pip

pip install <package name>

Newer versions of the package can updated later with

pip install <your\_package\_name> --upgrade