# Build a Python

## Creating a package

First, we need to think of a way to structure our code, so that others can access our code functionalities. In Python, to make a package, we need to add an __init__.py to the directory. Here, we are going to make a package called test_package.

write the __init__.py

```
from collections import Counter

  def count_in_list(l, word):

  c = Counter(l)

  return c[word]
```

Now, create a directory called test_package, and in it keep the __init__.py file. That's all, our package is ready. All our package does count the number of occurrences of a word in a list. Now, to use our package, create a run.py file outside the test_package directory. Inside the run.py simply import the newly created package and use the count_in_list function. We can write the code as shown below.

```
from test_package import count_in_list



l = ["gfg", "dsa", "gfg"]

count = count_in_list(l, "gfg")

print(count)
```

That's all. We have just created our first package. Our directory structure should look something like this

```
package-folder

├── run.py

└── test_package

└── __init__.py
```

```
1 directory, 2 files
```

To test it out, simply type python3 run.py. And our output should be as follows:

```
package-folder
├── LICENSE
├── README.md
├── setup.py
└── test_package
└── __init__.py

1 directory, 4 files
```

Make sure to remove the previously created run.py, it was intended for manual testing only. For the LICENSE we would recommend going with the MIT LICENSE as it offers the most flexibility. We can read up about various types of licenses online.

Next, create the README.md file. It would essentially contain a complete description of the package. If we are new to markdown-style writing, we recommend reading this article. Once, we have your README.md file ready, we need to write the setup.py. This is the most important part.

```
import setuptools

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

setuptools.setup(
    name="test-package",
    version="0.0.1",
    author="GeeksforGeeks",

    author_email="contact@gfg.com",

    packages=["test_package"],

    description="A sample test package",

    long_description=description,

    long_description_content_type="text/markdown",

    url="https://github.com/gituser/test-tackage",

    license='MIT',

    python_requires='>=3.8',

    install_requires=[]

)
```

In the above code, you need to change the

author with your name

author_email  with your email

url with your GitHub URL of the package

Our package is now ready.

Register our package to PyPI

Now that we developed our python package, we need to register it on PyPI.

## 1. Upload to GitHub

Create a new GitHub repository and push all our code there. If you don't know how to push code to a GitHub repo, you can head over and read this article. Also, do not forget to update our URL in setup.py with the newly created GitHub repo URL. Our repo should be publicly accessible.

## 2. Create an account in PyPI

We are going to publish the package in PyPI, we need an account. To do that, simply visit PyPI and create your account.

## 3. Generating distributions

Distribution archives are needed for hosting it as a package. To generate these packages, we need to install two additional packages.

```
pip3 install setuptools wheel
```

Now that we installed them, run the below command to generate archives:

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

It will generate folders build, dist, test_package.egg-info. Now your directory structure should look something like this.

```
package-folder
├── build
|   ├── bdist.linux-x86_64
|   └── lib
|   └── test_package
|   └── __init__.py
```

```
├── dist
|   ├── test_package-0.0.1-py3-none-any.whl
|   └── test-package-0.0.1.tar.gz
├── LICENSE
├── README.md
├── setup.py
├── test_package
|  └── __init__.py
└── test_package.egg-info
├── dependency_links.txt
├── PKG-INFO
├── SOURCES.txt
└── top_level.txt

7 directories, 11 files
```

**4. Deploy**

To upload to PyPI we need the twine package installed.

```
pip3 install twine
```

Now, upload the package with the proper version defined in the setup.py file. To do so, run the below command

```
twine upload --repository pypi dist/*
```

That was all about developing and deploying packages in python.

## Using our package

Once the package is developed, we should be able to use it right? After all, it was developed to reuse the same logic at different codebases. Create a completely new directory and create a virtual environment inside it. To so, simply type

```
python3 -m venv env

source env/bin/activate
```

Now, install your newly deployed package to the virtual environment. You need to use pip to be able to install it.

```
pip install test-package
```

Now, let’s see how we can use this package in our code. The idea is simple. This is after all a package, right? So, all we need to do is import it as a package.

```
from test_package import count_in_list
```

Now, we have our function. Let’s try to use it.

```
print(count_in_list(["gfg", "dsa", "gfg"], "gfg")) # output: 2

print(count_in_list(["gfg", "dsa", "gfg"], "maths")) # output: 0
```

That’s all.  If we put it all together,

```
from test_package import count_in_list


print(count_in_list(["gfg", "dsa", "gfg"], "gfg")) # output: 2

print(count_in_list(["gfg", "dsa", "gfg"], "maths")) # output: 0
```

**Output:**

So that was all to developing a package and deploying it to PyPI.