Packages

What is package?
Package is collection of modules (.py files) (OR) python programs.
Technically package is nothing but folder, which consist of .py files.
Packing is nothing but grouping set of related programs and referring with one name.

numpy → package
sys.py → module
os.py
calendar.py
pandas → package

Advantage of packages
1. Grouping and referring with one name
2. Maintaining of modules becomes easy
3. Hiding modules
4. Creating app installer or compressing in single file (.whl)

Packages are 2 types
1. Predefined packages
2. User defined packages

Predefined packages (libraries)
The existing packages are called predefined packages. These packages are provided python or third party vendors.
Example: numpy, pandas, tkinter, django, flask,..

User defined packages
The packages created by programmer are called user defined packages. These packages are application specific.
Example: emp, college, chess,…

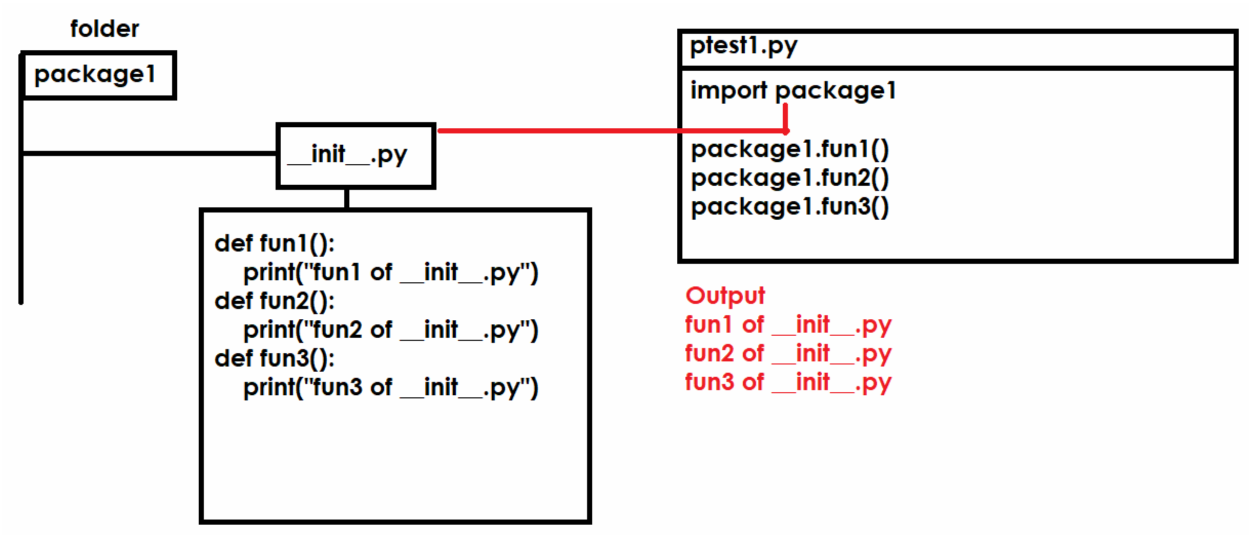
How to create package?
Creating package is nothing but creating folder and storing .py files
Every package must contain a special program or file called __init__.py

What is __init__.py?

__init__.py is a special file in Python that serves two primary purposes:

Package Declaration:
It marks a directory as a Python package. Without this file, Python would not recognize the directory as a package, and you wouldn't be able to import modules from it.
Initialization:
It can contain code that is executed when the package is imported. This code can be used to initialize the package, set up configurations, or perform any other necessary tasks.

This __init__.py is imported automatically when package is imported
__init__.py is a special module, this module is referred with package name.

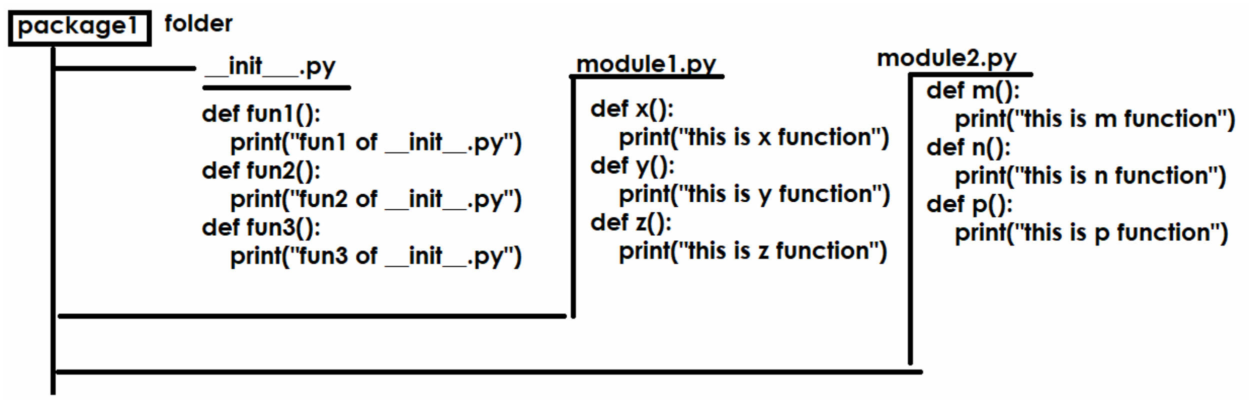


```
def fun1():
   print("fun1 of __init__.py")
def fun2():
   print("fun2 of __init__.py")
def fun3():
   print("fun3 of __init__.py")
```

```
import package1

package1.fun1()
package1.fun2()
package1.fun3()
```

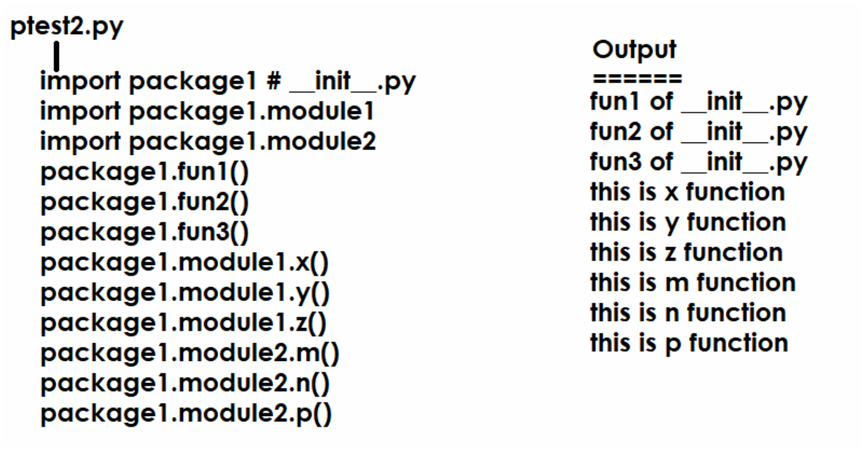
```
Output
fun1 of __init__.py
fun2 of __init__.py
fun3 of __init__.py
```



```
def fun1():
   print("fun1 of __init__.py")
def fun2():
   print("fun2 of __init__.py")
def fun3():
   print("fun3 of __init__.py")
```

```
def x():
   print("this is x function")
def y():
   print("this is y function")
def z():
   print("this is z function")
```

```
def m():
   print("this is m function")
def n():
   print("this is n function")
def p():
   print("this is p function")
```



```
ptest2.py

import package1 # __init__.py
import package1.module1
import package1.module2
package1.fun1()
package1.fun2()
package1.fun3()
package1.module1.x()
package1.module1.y()
package1.module1.z()
package1.module2.m()
package1.module2.n()
package1.module2.p()
```

```
Output
======
fun1 of __init__.py
fun2 of __init__.py
fun3 of __init__.py
this is x function
this is y function
this is z function
this is m function
this is n function
this is p function
```

Example:

```
import package1 # __init__.py
from package1.module1 import *
from package1.module2 import *
package1.fun1()
package1.fun2()
package1.fun3()
x()
y()
z()
m()
n()
p()
```

Example:

```
import package1 # __init__.py
from package1 import module1
from package1 import module2
package1.fun1()
package1.fun2()
package1.fun3()
module1.x()
module1.y()
module1.z()
module2.p()
module2.m()
module2.n()
```

__init__.py is used,

1. Import modules from within the package, making them directly accessible when the package is imported. (hiding modules)

__init__.py
```
from package1.module1 import *
from package1.module2 import *
def fun1():
   print("fun1 of __init__.py")
def fun2():
   print("fun2 of __init__.py")
def fun3():
   print("fun3 of __init__.py")
```

ptest3.py

```
import package1 # __init__.py

package1.fun1()
package1.fun2()
package1.fun3()
package1.x()
package1.y()
package1.z()
package1.p()
package1.m()
package1.n()
```

2. Define variables or functions that are available at the package level.

Create package with name pacakge2
Inside package2 add the following modules

__init__.py
```
x=100 # pacakge level variable
def fun1(): # package level function
   print("package level function1")
def fun2(): # package level function
   print("package level function2")
```

m1.py
```
import package2

def fun3():
    print("function3")
    package2.fun1()
    print(package2.x)
```

m2.py

```
import package2

def fun4():
    print("function4")
    package2.fun2()
    pacakge2.fun1()
    print(pacakge2.x)
```

Create the following module outside the package for testing

ptest4.py
```
import package2.m1
import package2.m2

package2.m1.fun3()
package2.m2.fun4()
```

Output
```
function3
package level function1
100
function4
package level function2
package level function1
100
```

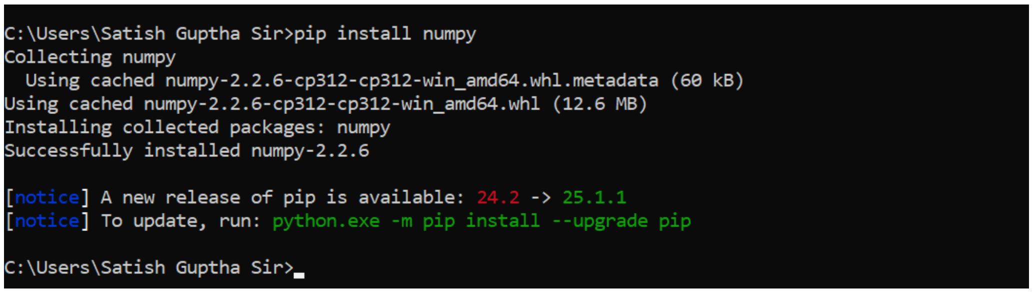
How to manage predefined packages?
Python software does not provides application specific libraries (numpy, pandas, matplotlib, keras, scipy, flask, django,…)

For managing predefined packages, python provides a tool called PIP.

PIP (Python Package Installer): A tool for installing and managing Python package. It is a command line utility which is executed from command prompt.

Installing package

pip install package-name



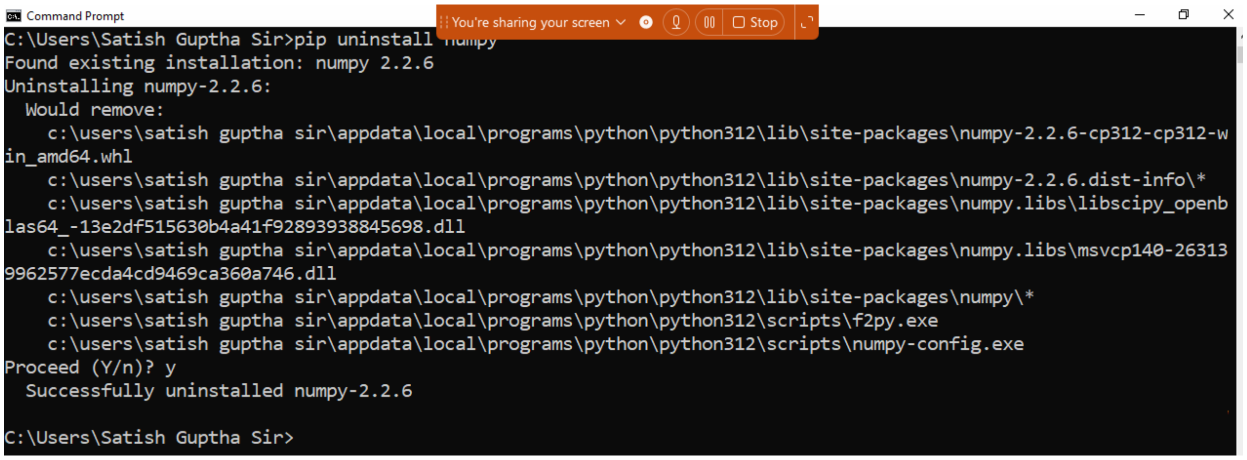
```
C:\Users\Satish Guptha Sir>pip install numpy
Collecting numpy
  Using cached numpy-2.2.6-cp312-cp312-win_amd64.whl.metadata (60 kB)
Using cached numpy-2.2.6-cp312-cp312-win_amd64.whl (12.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.2.6

[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Satish Guptha Sir>_
```

Uninstalling package

Pip uninstall package-name

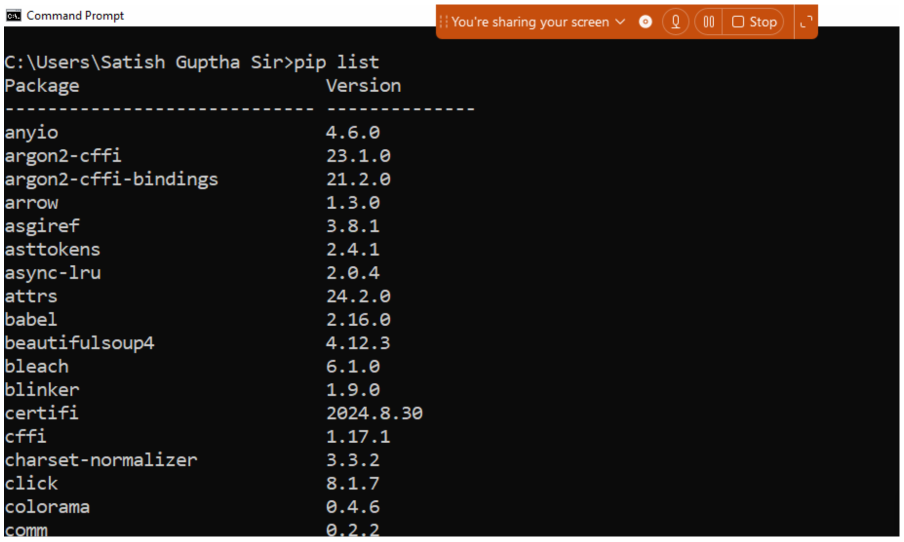


```
C:\Users\Satish Guptha Sir>pip uninstall numpy
Found existing installation: numpy 2.2.6
Uninstalling numpy-2.2.6:
  Would remove:
    c:\users\satish guptha sir\appdata\local\programs\python\python312\lib\site-packages\numpy-2.2.6-cp312-cp312-w
in_amd64.whl
    c:\users\satish guptha sir\appdata\local\programs\python\python312\lib\site-packages\numpy-2.2.6.dist-info\*
    c:\users\satish guptha sir\appdata\local\programs\python\python312\lib\site-packages\numpy.libs\libscipy_openb
las64_-13e2df515630b4a41f92893938845698.dll
    c:\users\satish guptha sir\appdata\local\programs\python\python312\lib\site-packages\numpy.libs\msvcp140-26313
9962577ecda4cd9469ca360a746.dll
    c:\users\satish guptha sir\appdata\local\programs\python\python312\lib\site-packages\numpy\*
    c:\users\satish guptha sir\appdata\local\programs\python\python312\scripts\f2py.exe
    c:\users\satish guptha sir\appdata\local\programs\python\python312\scripts\numpy-config.exe
Proceed (Y/n)? y
  Successfully uninstalled numpy-2.2.6

C:\Users\Satish Guptha Sir>
```

Listing installed packages

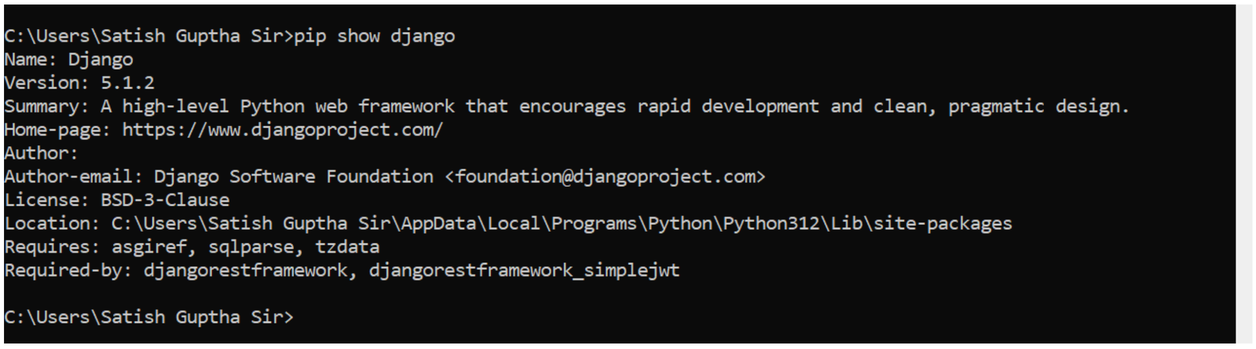
pip list



```
C:\Users\Satish Guptha Sir>pip list
Package                       Version
----------------------------- --------------
anyio                         4.6.0
argon2-cffi                   23.1.0
argon2-cffi-bindings          21.2.0
arrow                         1.3.0
asgiref                       3.8.1
asttokens                     2.4.1
async-lru                     2.0.4
attrs                         24.2.0
babel                         2.16.0
beautifulsoup4                4.12.3
bleach                        6.1.0
blinker                       1.9.0
certifi                       2024.8.30
cffi                          1.17.1
charset-normalizer            3.3.2
click                         8.1.7
colorama                      0.4.6
comm                          0.2.2
```

Displaying information about package

Pip show package-name



```
C:\Users\Satish Guptha Sir>pip show django
Name: Django
Version: 5.1.2
Summary: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
Home-page: https://www.djangoproject.com/
Author:
Author-email: Django Software Foundation <foundation@djangoproject.com>
License: BSD-3-Clause
Location: C:\Users\Satish Guptha Sir\AppData\Local\Programs\Python\Python312\Lib\site-packages
Requires: asgiref, sqlparse, tzdata
Required-by: djangorestframework, djangorestframework_simplejwt

C:\Users\Satish Guptha Sir>
```

Object Oriented Programming (OOP)

The main objective learning Object Oriented Programming is developing user defined data types (OR) building custom classes or data types.