How import modules at package level?

__init__.py

1. To access module level functions using package name, by avoiding package name
2. To create package level variables, functions and classes
3. Package initialization is done using __init__.py

__init__.py

```
from .m1 import *
```

test1.py

```
import package1

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

__init__.py

```
from .m1 import *

p=100 # package level variable
```

m1.py

```
import package1
def fun1():
    print("fun1 of m1 inside package1")
    print(package1.p)
def fun2():
    print("fun2 of m1 inside package1")
def fun3():
    print("fun3 of m1 inside package1")
```

m2.py

```
import package1
def fun2():
    print(package1.p)
```

How manage predefined packages?

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

PIP stands for **Preferred Installer Program.** It is an installer used for installing predefined packages.

All python libraries are exists in www.pypi.org
pypi stands python package index, it is a repository of python packages or libraries.

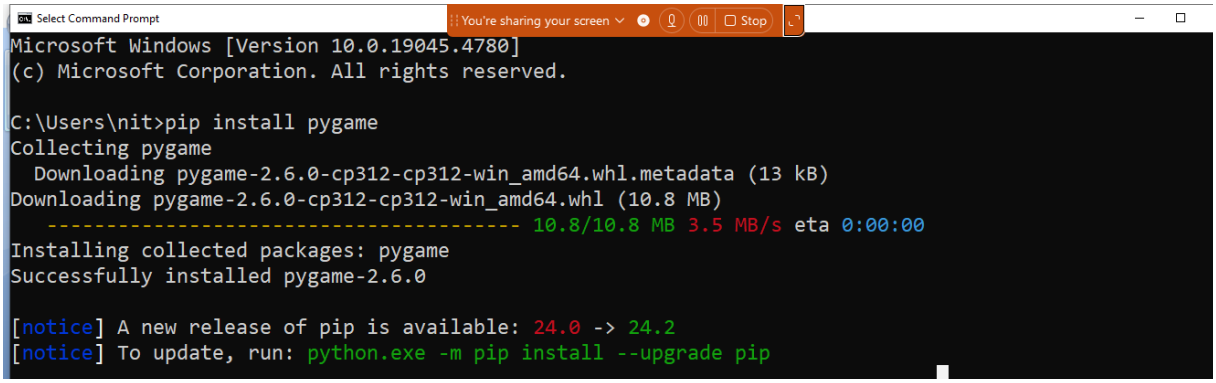
Pip provides the following commands for managing packages.

1. **Install** : this command is for installing package

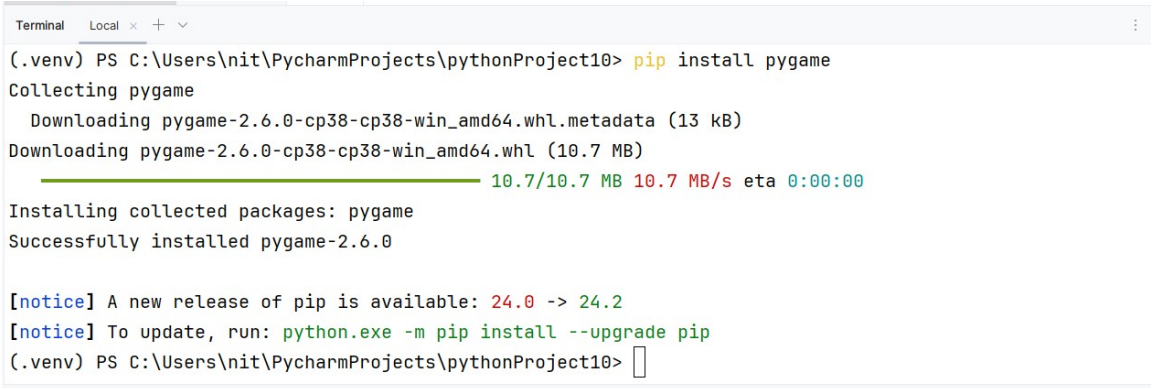
Syntax: pip install <package-name>

This command is executed from command prompt (OR) in pycharm terminal and execute command

Command Prompt



PyCharm



2. **Uninstall** : This command for uninstalling or removing existing packages

Syntax: pip uninstall <package-name>

```
C:\Users\nit>pip uninstall chess
Found existing installation: chess 1.10.0
Uninstalling chess-1.10.0:
  Would remove:
    c:\users\nit\appdata\local\programs\python\python312\lib\site-packages\chess-1.10.0.dist-info\*
    c:\users\nit\appdata\local\programs\python\python312\lib\site-packages\chess\*
Proceed (Y/n)? y
  Successfully uninstalled chess-1.10.0

C:\Users\nit>pip uninstall flask
Found existing installation: Flask 3.0.0
Uninstalling Flask-3.0.0:
  Would remove:
    c:\users\nit\appdata\local\programs\python\python312\lib\site-packages\flask-3.0.0.dist-info\*
    c:\users\nit\appdata\local\programs\python\python312\lib\site-packages\flask\*
    c:\users\nit\appdata\local\programs\python\python312\scripts\flask.exe
Proceed (Y/n)? y
  Successfully uninstalled Flask-3.0.0
```

3. **list** : This is used for listing all the installed packages

```
C:\Users\nit>pip list
Package                       Version
----------------------------- -----------
asgiref                       3.8.1
attrs                         23.2.0
beautifulsoup4                4.12.2
blinker                       1.6.3
```

4. **show** : This command display information about package

```
C:\Users\nit>pip show pandas
Name: pandas
Version: 2.2.2
Summary: Powerful data structures for data analysis, time series, and statistics
Home-page: https://pandas.pydata.org
Author:
Author-email: The Pandas Development Team <pandas-dev@python.org>
License: BSD 3-Clause License

Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc. and PyData Development Te

All rights reserved.
```

5. **freeze** : freeze read the all installed libraries. These libraries can redirected to one file

```
C:\Users\nit>pip freeze>repo

C:\Users\nit>
```

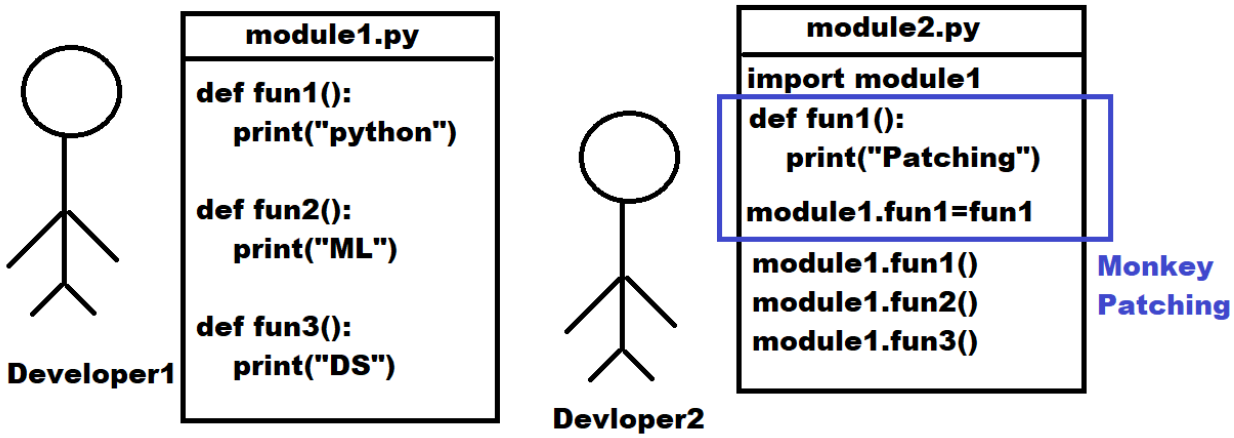
6. **download** : this command download package but not install (zip file/wheel file)

```
C:\Users\nit>pip download pygame
Collecting pygame
  Using cached pygame-2.6.0-cp312-cp312-win_amd64.whl.metadata (13 kB)
Using cached pygame-2.6.0-cp312-cp312-win_amd64.whl (10.8 MB)
Saved c:\users\nit\pygame-2.6.0-cp312-cp312-win_amd64.whl
Successfully downloaded pygame

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\nit>pip show pygame
WARNING: Package(s) not found: pygame

C:\Users\nit>pip install pygame
Collecting pygame
  Using cached pygame-2.6.0-cp312-cp312-win_amd64.whl.metadata (13 kB)
Using cached pygame-2.6.0-cp312-cp312-win_amd64.whl (10.8 MB)
Installing collected packages: pygame
```

Monkey Patching

What is monkey patching?

By using Monkey Patching in Python, you can make these modifications at runtime without altering the source code. Monkey patching refers to the dynamic (run-time) modification of a class or module




```
>>> import math
>>> math.pi
3.141592653589793
>>> math.pi=3.147 # Monkey Patch
>>> math.pi
3.147
>>>
========== RESTART: Shell ==========
>>> import math
>>> math.pi
3.141592653589793
```

Module1.py	Test.py
`x=100 # global variable`	`import module1`

```
y=200 # global variable
def fun1():
    print("inside fun1 of module1")

def fun2():
    print("inside fun2 of module1")

def fun3():
    print("inside fun3 of module1")
```

```
def monkey_fun1():
    print("Patch Function")

module1.fun1=monkey_fun1

module1.fun1()
module1.fun2()
module1.fun3()
```

Object Oriented Programming (OOP)