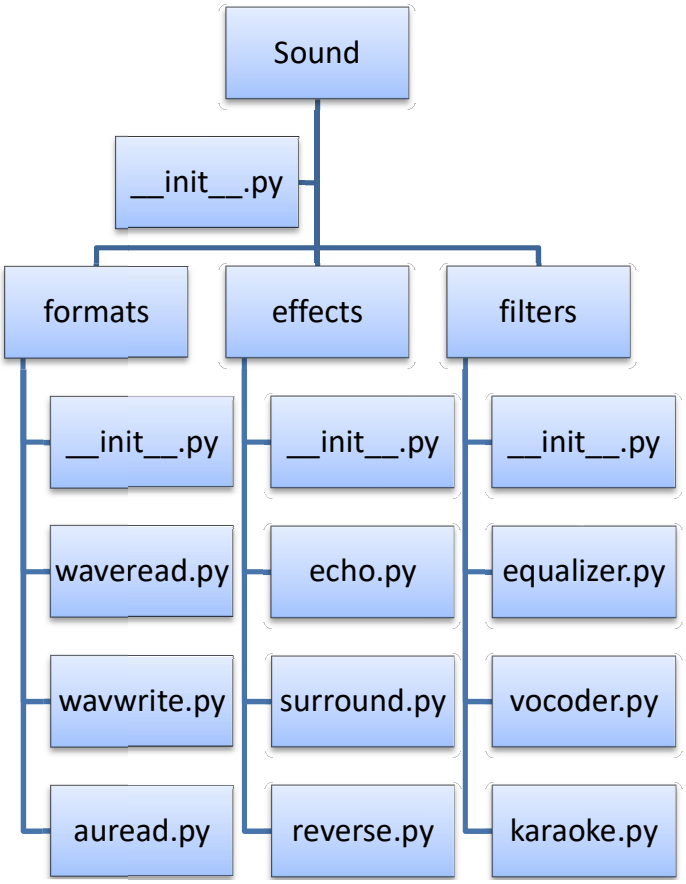
Chapter - 8: Scope and Modules

Packages

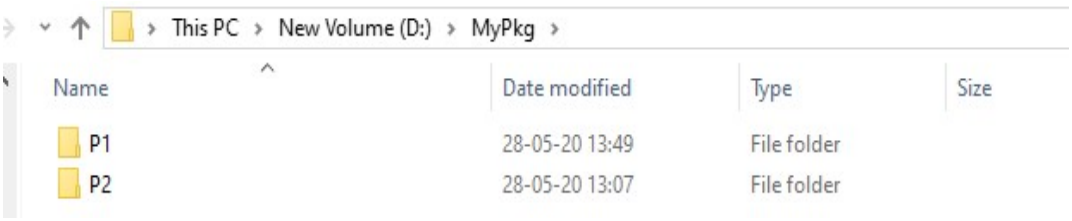
- A package is a well-organized hierarchy of sub-directories and files, for easier access of modules.
- The directory is a package [or sub package] and files are modules.
- Package is a place for similar modules, and help in grouping and organizing them.
- Package directory or sub-directory must contain a file "**__init__.py**" used as initialization code. It can be an empty file. However, it new version of python, this is not compulsory.
- Packages help avoid collisions between module names and modules help avoid collision between variable names.
- Packages are helpful to manage the large projects having lots of mudules.



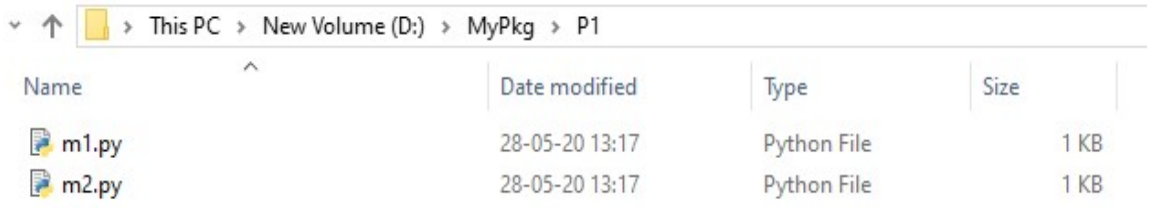
Create Package

To create a Package – MyPkg perform the following steps –

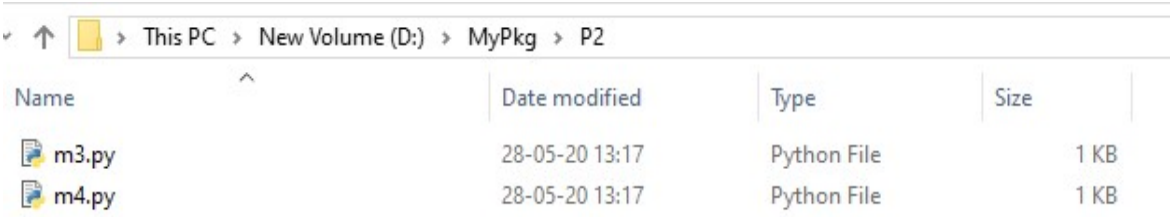
1) Create a folder on “D:\MyPkg”
2) Create a sub folder in D:\MyPkg – P1 & P2



3) Create a module **m1.py** , **m2.py** in D:\MyPkg\P1 sub folder.



4) Create a module m3.py and m4.py in D:\MyPkg\P1 sub folder.



5) Content of the Module are as follows

Module M1.py

```
Employee={ "name":"Ajay M1", "age":27, "salary":30000}
def show(name):
    print("Name :", name)
```

Module M2.py

```
Employee={ "name":"Vijay M2", "age":27, "salary":30000}
def sum(x,y):
   return(x+y)
```

Module M3.py

```
def show(name,age):
      print("Name :", name)
      print("Age :", age)

Employee={ "name":"Ajay M3", "age":27, "salary":30000}
```

Module M4.py

```
def mul(x,y):
   print("sum : ", x*y)

Employee={ "name":"Vijay M4", "age":27, "salary":30000}
```

Import Package

- Import Statement help to provide the Module contents to the caller program code.
- Import Package creates a separate namespace for each sub package, which contains all the objects define in the module.
- We can provide the path for package using **sys.path.append**("Package Path") in the code or modify the PYTHONPATH environment variable.
- Package content like identifiers, functions, class or objects defined in the module can be accessed using the **dot notation** like <PackageName>.<ModuleName>**.**<FunctionName>. eg. MyPkg.my_abc**.**show( )

Syntax

```
import  Package_Name.Module_Name
```

Example

To import the package "D:**Mykg\P1**" for using in the "**m1.py**" and "**m2.py**" module file.

```
import sys
sys.path.append("D:\\")

import MyPkg.P1.m1
import MyPkg.P1.m2

print("Objects in P1 - m1 module ")
print( dir(MyPkg.P1.m1))

print("Objects in P1 - m2 module ")
print( dir(MyPkg.P1.m2))

print("Accessing P1 - m1 module ")
MyPkg.P1.m1.show("Ajay")

print("Accessing P1 - m1 module ")
print( MyPkg.P1.m1.Employee["name"])

print("Accessing P1 - m2 module ")
print( "Sum : ", MyPkg.P1.m2.sum(20,20))

print("Accessing P1 - m2 module ")
print( MyPkg.P1.m2.Employee["name"])
```

Output

```
Objects in P1 - m1 module
['Employee', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'show']

Objects in P1 - m2 module
['Employee', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'sum']

Accessing P1 - m1 module
Name : Ajay

Accessing P1 - m1 module
Ajay M1

Accessing P1 - m2 module
Sum : 40

Accessing P1 - m2 module
Vijay M2
```

In this output , “Employee” & “show” exists in Module-m1 and
“Employee” & “sum” exists in Module –m2.
Employee object exists in both Module(m1,m2) , but it is not colliding with each other , due to separate namespace.

Renaming Module

To import the package “D:**Mykg\P1\m1**” and rename it as“**M1**” for using in the program file.

```
import sys
sys.path.append("D:\\")

import MyPkg.P1.m1 as M1
import MyPkg.P1.m2 as M2

print("Accessing P1 - m1 module ")
M1.show("Ajay")

print("Accessing P1 - m1 module ")
print( M1.Employee["name"])

print("Accessing P1 - m2 module ")
print( "Sum : ", M2.sum(20,20))

print("Accessing P1 - m2 module ")
print( M2.Employee["name"])
```

Output

```
Accessing P1 - m1 module
Name : Ajay

Accessing P1 - m1 module
Ajay M1

Accessing P1 - m2 module
Sum : 40

Accessing P1 - m2 module
Vijay M2
```

Using the dir( ) function

This function is used to list all the functions or variables defined in the Package imported.

```
import sys
sys.path.append("D:\\")
import MyPkg.P1.m1 as M1
import MyPkg.P1.m2 as M2

print("Objects in P1 - m1 module ")
print( dir(M1))
print( M1.__file__)
print( M1.__name__)

print("Objects in P1 - m2 module ")
print( dir(M2))
print( M2.__file__)
print( M2.__name__)
```

Output

```
Objects in P1 - m1 module

['Employee', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'show']

D:\MyPkg\P1\m1.py

MyPkg.P1.m1


Objects in P1 - m2 module

['Employee', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'sum']

D:\MyPkg\P1\m2.py

MyPkg.P1.m2
```

In this, **Employee** and **show** is defined in the "**m1"** module and
Employee and **sum** is defined in the "**m2"** module and
Remaining is the system defined.