Monday, November 21, 2022 8:26 PM

Modules and Packages

Modules:- A group of functions, variables and classes saved a file, which is nothing but a module and it's in build.

ex:- math, os, sys, functools, unittest etc.

Every:- Python file show as a module.

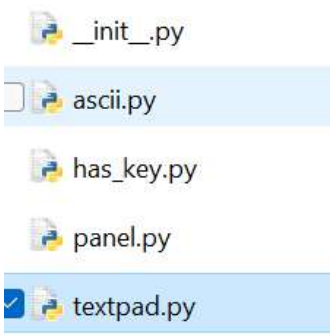
```
class Template:
    """A string class for supporting $-substitutio
    delimiter = '$'

    idpattern = r'(?a:[_a-z][_a-z0-9]*)'
    braceidpattern = None
    flags = _re.IGNORECASE

    def __init_subclass__(cls):
        super().__init_subclass__()
        if 'pattern' in cls.__dict__:
            pattern = cls.pattern
        else:
```

Packages :- It is an encapsulation mechanism to group related modules in a single unit, it's a package of module. It's look like a folder and it’s contain several python files(.py) and __init__.py(imp)

Ex:- pandas, numpy, pytest, tenserflow etc



Using **import** keyword we call python module

```
import math
import opcode
import os
import datetime
import sys
import zipfile
import json
import xml

print(dir(math))
```

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
>>> math.pi
3.141592653589793
>>> math.pow(3,2)
9.0
```

How to create user define module.:-

Create a python file, and create class, functions and variables

modules_1.py modules_2.py

```python

a = 70
def add(b,c):
    add = b+c+a
    return add

def product(b,c):
    prod = b*c
    return prod

```

Note:- If we want call this python file as a module, create another python file and using import keyword, import the module .

Ex:- 1st file :- modules_1.py
2nd file:- module_2.py

Import modules_1.py

modules_1.py modules_2.py

```python
import modules_1
print(dir(modules_1))
print(modules_1.add(40,30))
print(modules_1.product(7,8))
```

Note:- Whenever we are using a module in our program . For that module complied file will be generated

and stored in the disk permanetly .

Class-2

Renaming a module at the of import(Module aliasing)

Using as keyword we will alias module name.

```
import math as m
print(dir(m))
```

Here math is a original module name and m is a alias name
We can access member(methods) by using alias name m.

from…..import

We can import particular member of module by using from…import.
The main advantage of this is we can access member directly without using module name, and also we reduce the storage.

```
from math import factorial,pow,pi
print(factorial(5))
print(pow(2,3))
print(pi)
```

Note:- We can import all member of module using *.

```
from math import *
print(factorial(12))
```

Various Possibility of import.

Import module name
Import module1,module2,module3
Import module **as** m
Import module1 **as** m1,module2 **as** m2,module3 **as** m3
From module **import** member
From module **import** member1,member2,member3
From module **import** *

Important Basic Modules.

```
import math
import opcode
import os
import datetime
import sys
import zipfile
import json
import xml
```

```
print(dir(math))
```

Math , os , sys, datetime (Very very important)

Math :-https://www.w3schools.com/python/module_math.asp

Note:- Last we will discuss Json

The Special Variable (__name__)

Fro every Python program a special variable __name__ will be added internally. This Variable stores information regarding whether the program is executed as an individual program as a module.

If the program executed as an individual program then the value a variable is __main__.

```
>>> import datetime
>>> dir(datetime)
['MAXYEAR', 'MINYEAR', '__all__', '__builtins__', '__cached__', '__doc__', '__fi
le__', '__loader__', '__name__', '__package__', '__spec__', 'date', 'datetime',
'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']
>>>
```

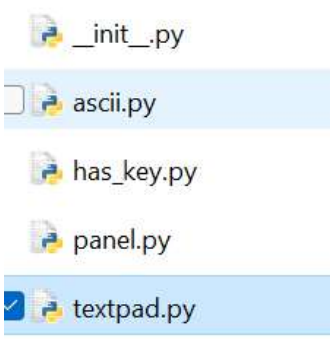
```
def f1():
    if __name__ == 'main':
        print('The code excutaed as a program')
    else:
        print('The code executed as a module from some other program')

print(f1())
```

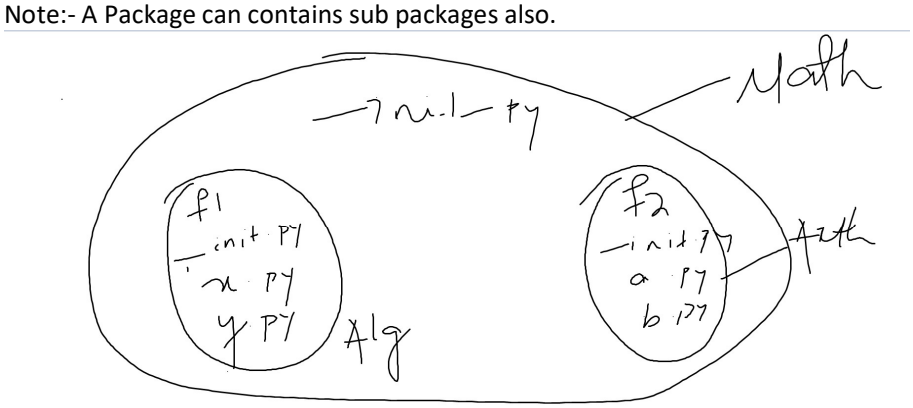
Packages

It is an encapsulation mechanism to group related modules in a single unit, it's a package of module. It's look like a folder and it’s contain several python files(.py) and __init__.py(imp)

Any folder or directory contains __init__.py file , is considered as a python package. This file can be empty .



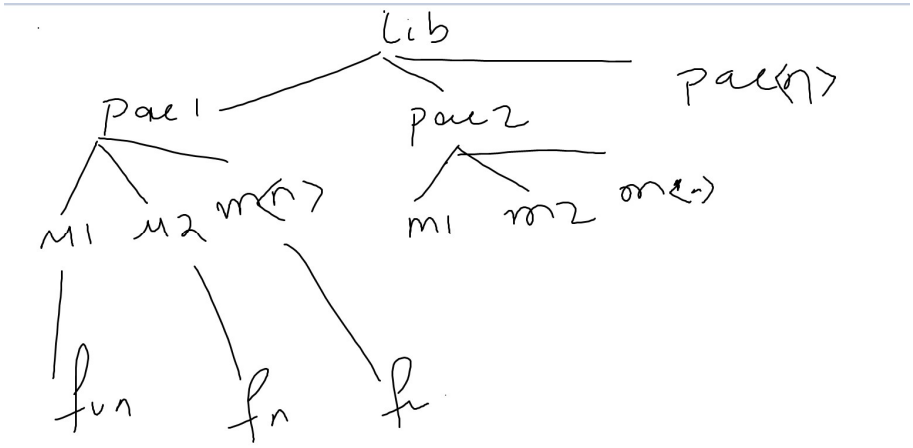
Note:- A Package can contains sub packages also.



The main advantage of package statement are.

1.We can resolve naming conflicts.(Camel Casing)
2.We can identify our components uniquely.

3.It improves modularity of the applications.



How install Lib or Packages in our system

Using **pip** we will install outside lib or package in our system.

Pip:- Python enchantment proposal , preferred installer program, python installer program

```
PS C:\Users\chand> pip install selenium
Collecting selenium
  Downloading selenium-4.7.2-py3-none-any.whl (6.3 MB)
     ---------------------------------------- 6.3/6.3 MB 17.5 MB/s eta 0:00:00
Requirement already satisfied: trio-websocket~=0.9 in c:\python3.10.2\lib\site-packages (from selenium) (0.9.2)
Requirement already satisfied: urllib3[socks]~=1.26 in c:\python3.10.2\lib\site-packages (from selenium) (1.26.9)
Requirement already satisfied: certifi>=2021.10.8 in c:\python3.10.2\lib\site-packages (from selenium) (2022.5.18.1)
Requirement already satisfied: trio~=0.17 in c:\python3.10.2\lib\site-packages (from selenium) (0.20.0)
Requirement already satisfied: async-generator>=1.9 in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (1.10)
Requirement already satisfied: attrs>=19.2.0 in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (21.4.0)
Requirement already satisfied: sortedcontainers in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: sniffio in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (1.2.0)
Requirement already satisfied: idna in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (3.3)
Requirement already satisfied: outcome in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (1.1.0)
Requirement already satisfied: cffi>=1.14 in c:\python3.10.2\lib\site-packages (from trio~=0.17->selenium) (1.15.0)
Requirement already satisfied: wsproto>=0.14 in c:\python3.10.2\lib\site-packages (from trio-websocket~=0.9->selenium) (1.1.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\python3.10.2\lib\site-packages (from urllib3[socks]~=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\python3.10.2\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\python3.10.2\lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.13.0)
Installing collected packages: selenium
Successfully installed selenium-4.7.2
WARNING: You are using pip version 22.0.3; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Python3.10.2\python.exe -m pip install --upgrade pip' command.
PS C:\Users\chand>
```

When we have no selenium module below error give.

```
Type "help", "copyright", "credits" or "license()
import selenium
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import selenium
ModuleNotFoundError: No module named 'selenium'
```

With the help of pip, install selenium the it's working fine and we will use selenium module

```
import selenium
dir(selenium)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '_
_package__', '__path__', '__spec__', '__version__']
```