```
                        Module's
                        -------
a python file is a python module,it contains executable
statements,variables,functions and classes.

our python file name is a module name.

        ex: sammple.py          --> sample is module name

in python,every file act as a one module,by default that module is called main
module.

if we want to access the data from one module to another module,first we need to
import that module.

whenever we are importing a module,that imported module act as a sub module.

modules provides reusability of code from outside the program.
modules provides better modularity of our applications.

how to importing a modules?
---------------------------
we can importing a modules in 3-ways,they are

        1).Normal importing a module

        2).from importing a module

        3).from import with '*'

Normal importing a module:
---------------------------
        import modulename

ex:
--
        arithmeticcalculations.py
        -------------------------
print("hai")
print(__name__) # module name
a=10
b=20
def add(x,y):
    z=x+y
    print(z)
add(4,5)
class test:
    def sub(self,x,y):
        self.x=x
        self.y=y
        print(self.x-self.y)
t1=test()
t1.sub(3,2)

        output
        ------
hai
__main__
9
1

        sample.py
        ---------
import arithmeticcalculations
print(__name__)
print(arithmeticcalculations.a)
print(arithmeticcalculations.b)
arithmeticcalculations.add(2,3)
t2=arithmeticcalculations.test()
t2.sub(2,3)

        output
        ------
hai
arithmeticcalculations
9
1
__main__
10
20
5
-1
```

what is the purpose/meaning of if __name__=='__main__' condition?

to check wheather that module is main module or not,wheather it is main module then only that condition is True otherwise that condition is False.

ex2:

```
        arithmeticcalculations.py
        -------------------------

print("hai")
print(__name__) # module name
a=10
b=20
def add(x,y):
    z=x+y
    print(z)
```

```
if __name__=='__main__':
    add(4,5)
class test:
    def sub(self,x,y):
        self.x=x
        self.y=y
        print(self.x-self.y)
if __name__=='__main__':
    t1=test()
    t1.sub(3,2)

        output
        ------
hai
__main__
9
1

        sample.py
        --------
import arithmeticcalculations
print(__name__)
print(arithmeticcalculations.a)
print(arithmeticcalculations.b)
arithmeticcalculations.add(2,3)
t2=arithmeticcalculations.test()
t2.sub(2,3)

        output
        ------
hai
arithmeticcalculations
__main__
10
20
5
-1
```

Renameing a module
----
we can rename or aliasing a module name by using 'as' keyword.

once we can rename a module afterthat we can access the properties from by using rename only.

ex:

```
        arithmeticcalculations.py
        -------------------------
```

```
print("hai")
print(__name__) # module name
a=10
b=20
def add(x,y):
    z=x+y
    print(z)
if __name__=='__main__':
    add(4,5)
class test:
    def sub(self,x,y):
        self.x=x
        self.y=y
        print(self.x-self.y)
if __name__=='__main__':
    t1=test()
    t1.sub(3,2)

        output
        ------
hai
__main__
9
1

        sample.py
        --------
import arithmeticcalculations as ac
print(__name__)
print(ac.a)
print(ac.b)
ac.add(2,3)
t2=ac.test()
t2.sub(2,3)

        output
        -----
hai
arithmeticcalculations
__main__
10
20
5
-1

from importing a module
-----------------------
        from modulename import p_1,p_2,....,p_n
```

```
ex:
---
        arithmeticcalculations.py
        -------------------------

print("hai")
print(__name__) # module name
a=10
b=20
def add(x,y):
    z=x+y
    print(z)
if __name__=='__main__':
    add(4,5)
class test:
    def sub(self,x,y):
        self.x=x
        self.y=y
        print(self.x-self.y)
if __name__=='__main__':
    t1=test()
    t1.sub(3,2)

        output
        ------
hai
__main__
9
1

        sample.py
        --------
from arithmeticcalculations import a,b,add,test
print(__name__)
print(a)
print(b)
add(2,3)
t2=test()
t2.sub(2,3)

        output
        -------
hai
arithmeticcalculations
__main__
10
20
5
-1
```

```
from importing a module with '*' option
----------------------------------------
        from modulename import *

ex:
---
        arithmeticcalculations.py
        -------------------------

print("hai")
print(__name__) # module name
a=10
b=20
def add(x,y):
    z=x+y
    print(z)
if __name__=='__main__':
    add(4,5)
class test:
    def sub(self,x,y):
        self.x=x
        self.y=y
        print(self.x-self.y)
if __name__=='__main__':
    t1=test()
    t1.sub(3,2)

        output
        ------
hai
__main__
9
1

        sample.py
        --------
from arithmeticcalculations import *
print(__name__)
print(a)
print(b)
add(2,3)
t2=test()
t2.sub(2,3)

        output
        ------
hai
arithmeticcalculations
__main__
10
```

```
20
5
-1
```

module search path:
----
whenever we are importing a module internally our python interpreter searching for that module in the following ways,

        1).main module location/current working directory

        2).python path/default python installation location

our imported module is not available in the above specified locations in that case to raise ModuleNotFoundError Exception.

ex1:

```
        C:\\python310\\siva\\wishes.py
        ----------------------------
print("hai")


        C:\\python310\\siva\\msg.py
        ---------------------------
import wishes
print("siva")
print("good morning")

        output
        -------
hai
siva
good morning
```

ex2:

```
        C:\\python310\\wishes.py
        ------------------------
print("hai")


        C:\\python310\\siva\\msg.py
        ---------------------------
import wishes
print("siva")
print("good morning")

        output
        -------
```

```
hai
siva
good morning

ex3:
---
        g:\\wishes.py
        -------------
print("hai")


        C:\\python310\\siva\\msg.py
        ---------------------------
import wishes
print("siva")
print("good morning")

        output
        -------
ModuleNotFoundError: No module named 'wishes'

reloading a module:
-------------------
by default each and every module is loaded into the memory location at only once.

ex:
---
        C:\\python310\\siva\\wishes.py
        ----------------------------
print("hai")


        C:\\python310\\siva\\msg.py
        ---------------------------
import wishes
print("siva")
print("good morning")
import wishes
print("krishna")
print("good evening")

        output
        ------
hai
siva
good morning
krishna
good evening

if we want to loading a module once again,in that case we are using reload() of
```

```
importlib module.

ex2:
---
        C:\\python310\\siva\\wishes.py
        ----------------------------
print("hai")


        C:\\python310\\siva\\msg.py
        ---------------------------
import wishes
import importlib
print("siva")
print("good morning")
importlib.reload(wishes)
print("krishna")
print("good evening")

        output
        ------
hai
siva
good morning
hai
krishna
good evening
```