IV. MODULES

A module allows you to logically organize the python code. Grouping related code into a module makes the code easy to use. A module is a file consisting python code.

- A module can define functions, classes and variables.
- A module can also include runnable code.

Ex

The Python code for a module support normally resides in a file named support.py.

```
def print_func (par):
print ("Hello : ", par)
return
```

We can invoke a module by two statements

- import statement
- from…import statement

i) The import Statement

- You can use any Python source file as a module by executing an import statement in some other Python source file

Syntax :

```
Import math
```

Example:

```
import math                             # Import module support
print(" the value of pi is ",math.pi)  # Now we can call the
                                        function in that module
as
```

Output:

```
The value of pi is 3.141592653589793
```

(ii) The from..import statement

Python from statement lets you import specific attributes from a module into the current namespace.

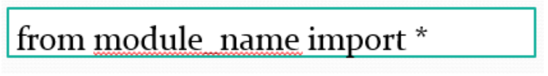
Syntax

```
from module_name import function_name
```

(iii) The from…import * statement

It is also possible to import all names from a module

Syntax



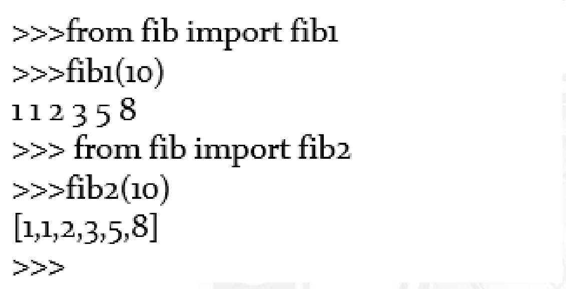
```
from module_name import *
```

Example

To import the function fib1 and /or fib2 from the module fib

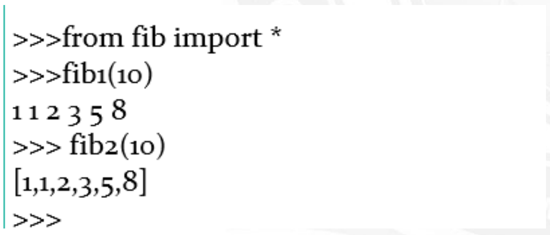
from fib import fib1

Let us import the each functions from the program defined fib.py



```
>>>from fib import fib1
>>>fib1(10)
1 1 2 3 5 8
>>> from fib import fib2
>>>fib2(10)
[1,1,2,3,5,8]
>>>
```

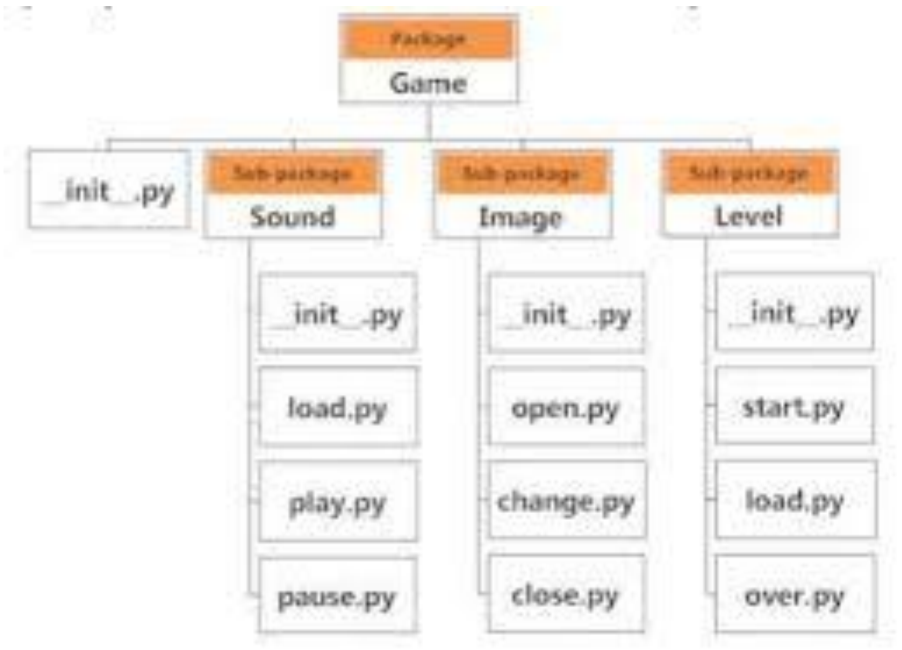
Let us use the from…import * statement



```
>>>from fib import *
>>>fib1(10)
1 1 2 3 5 8
>>> fib2(10)
[1,1,2,3,5,8]
>>>
```

V.PACKAGES

- A **package** is a collection of modules. A Python package can have sub-packages and modules.
- A directory must contain a file named __init__.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.
- Here is an example. Suppose we are developing a game, one possible organization of packages and modules could be as shown in the figure below.



Importing module from a package

We can import modules from packages using the dot (.) operator.

- For example, if want to import the start module in the above example, it is done as follows. *Import Game.Level.start*
- Now if this module contains a function named *select_difficulty(),* we must use the full name to reference it.

  Game.Level.start.select_difficulty(2)
- If this construct seems lengthy, we can import the module without the package prefix as follows.

  from Game.Level import start
- We can now call the function simply as follows.

  start.select_difficulty(2)
- Yet another way of importing just the required function (or class or variable) form a module within a package would be as follows.

  from Game.Level.start import select_difficulty
- Now we can directly call this function.

  select_difficulty(2)
- Although easier, this method is not recommended. Using the full namespace avoids confusion and prevents two same identifier names from colliding.

- While importing packages, Python looks in the list of directories defined in sys.path, similar as for module search path.