**Programming and Problem Solving through Python Language**
**O Level / A Level**

# Chapter - 8: Scope and Modules

## Modular Programming

- It is a process of breaking large programs into smaller and manageable programs.
- Modularity in python can be implemented through using functions, modules and packages.
- These modules can be clubbed together to created a larger application.

## Advantage of Modular Programming

1. **Simplicity** – Instead of taking the entire problem, we take a small portion of the problem. It can make the code much easier to develop and maintain. eg. Function or Module.

2. **Maintainability** – Function or Module applies the logical boundaries between portions of problem. It reduces the coupling or interdependency among the modules. This helps the team of programmer to work simultaneously on the large projects.

3. **Reusability** – It allows reusing the same code in different part of the large application, without duplicating them. Import utilities used in the python to reuse the functionality.

4. **Scoping** – Modules or Packages creates a namespace which helps to remove the ambiguity for names defined in the programs.

## Modules

- A module helps to break the large program to manageable and organized code. It allows the reusability of code.
- Module is a Python file containing a set of Python Statements and Functions.
- Module files has the ".py" extension. e.g. my_abc.py

## Create Module

To create a module , save a file with extension ".py"

File **my_abc.py**

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

Employee= { "name":"Ajay", "age":30, "Salary":20000 }
```

## Import Module

- Import Statement help to provide the Module contents to the caller program code.
- Import module created a separate namespace, which contains all the objects define in the module.
- Module content like identifiers , functions , class or objects defined in the module can be accessed using the **dot notation** like <ModuleName>**.**<FunctionName>. eg. my_abc**.**show( )

**Syntax**

```
import  Module_Name
```

**Example**

To import the module "**my_abc.py**" for using in the "**m1.py**" program file.

```
import my_abc

print("\n Accessing function from  Module\n")
my_abc.show("vikas",27)

print("\n Accessing Variable from  Module\n")
print(my_abc.Employee["name"])
print(my_abc.Employee["age"])
print(my_abc.Employee["salary"])
```

**Output**

```
Accessing function from  Module
Name :  vikas
Age :  27

 Accessing Variable from  Module
Ajay
30
20000
```

## Renaming Module

To import the module "**my_abc.py**" and rename it "**m**" for using in the "**m1.py**" program file.

```
import my_abc as m

print("\n Accessing function from  Module\n")
m.show("vikas",27)

print("\n Accessing Variable from  Module\n")
print(m.Employee["name"])
print(m.Employee["age"])
print(m.Employee["salary"])
```

**Output**

```
Accessing function from  Module
Name :  vikas
Age :  27

 Accessing Variable from  Module
Ajay
30
20000
```

## Using the dir( ) function

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

```
import my_abc
x=dir( my_abc )
print( x )

print( __file__ )
print( __name__ )
```

**Output**

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

C:/Python38-32/m2.py

__main__
```

In this, **Employee** and **show** is defined in the "**my_abc"** module and remaining are the system defined.

## Using the from …import

- This is used to import the part of the module as required.
- If the module is imported using the "**from..import**" syntax, then module name is not used before the function or variable imported.
- For example if show( ) is imported from my_abc module, we can't use my_abc.show( ), instead use show( ) only.

**Example-1**

```
from my_abc import show

print("\n Accessing function from  Module\n")
show("vikas", 27)
```

```
print("\n Accessing Variable from  Module\n")
print(Employee["name"])             # This is not accessible,  as not imported.
```

**Output**

```
Accessing function from  Module
Name :  vikas
Age :  27

 Accessing Variable from  Module

Traceback (most recent call last):
  File "C:/Python38-32/m1.py", line 7, in <module>
    print(Employee["name"])
NameError: name 'Employee' is not defined
```

**Example-2**

```
from my_abc import Employee

print("\n Accessing Variable from  Module\n")
print(Employee["name"])
print(Employee["age"])
print(Employee["salary"])

print("\n Accessing function from  Module\n")
show("vikas",27)
```

**Output**

```
Accessing Variable from  Module

Ajay
30
20000

 Accessing function from  Module

Traceback (most recent call last):
  File "C:/Python38-32/m1.py", line 10, in <module>
    show("vikas",27)
NameError: name 'show' is not defined
```

## Importing all the Modules using from…import

- It can be used to import the objects from the modules.
- If the module is imported using the "**from..import**" syntax, then module name is not used before the function or variable imported.

**Example**

```
from my_abc import *

print("\n Accessing Variable from  Module\n")
print(Employee["name"])
print(Employee["age"])
print(Employee["salary"])


print("\n Accessing function from  Module\n")
show("vikas",27)
```

**Output**

```
Accessing Variable from  Module

Ajay
30
20000

 Accessing function from  Module

Name :  vikas
Age :  27
```