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

Chapter – 8 : Scope and Module

Scope of Object and Names

- The scope refers to the region in the program code where variables and names are accessible.
- It also determines the visibility and lifetime of a variable in the program code.
- Assigning a value, defining a function or class (using def or class) or importing module operation creates a name and its place in the code defines its scope.
- The names or objects which are accessible are called **in-scope**.
- The names or objects which are not accessible are called **out-of-scope**.
- The Python scope concept follows the **LEGB** (Local, Enclosing, Global and built-in) rule.
- The scope concept helps to avoid the name collision and use of global names across the programs.

Names and Objects in Python

- Python is a dynamically types language, so when we assign the value to the variable for the first time, it creates the variable. Others names or objects come in existence as given :

Operation	**Statement**
Assignment	X=value
Import Operation	import module
Function definitions	def fun( )
Arguments in the function	def fun(x1, x2 , x3)
Class definition	class abc :

- Python uses the location of the name assignment or definition to associate it with a scope.
- If a variable assigned value in the function, it has a local scope.
- If a variable assigned value at the top level and outside all the functions, it has a global scope.

Python Scope Vs Namespace

- The scope of a variables and objects are related to the concept of the namespace.
- The scope determines the visibility of the names and objects in the program.
- A namespace is a collection of names.
- In python, scopes are implemented as dictionaries that maps names to objects. These dictionaries are called namespaces.
- The dictionaries have the names as key and the objects as value.
- A strings, lists, functions, etc everything in Python is an object.

Searching a name in the Namespace

- Whenever we use a name, such as a variable or a function name, Python searches through different scope levels (or namespaces) to determine whether the name exists or not.
- If the name exists, then we always get the first occurrence of it.
- Otherwise, we get an error.

Types of namespace

1.	Built-in Namespace	It includes functions and exception names that are built-in in the python.
2.	Global Namespace	It includes the names from the modules that are imported in the code. It lasts till the end of program.
3	Local Namespace	It includes the names defined in the function. It is created when the function is called and ends when the value returned.

LEGB Rule for Python Scope

- The LEGB stands for Local, Enclosing, Global and Built-in.
- Python resolves names using the LEGB rules.
- The LEGB rule is a kind of name lookup procedure, which determines the order in which Python looks up names.
- For example, if we access a name, then Python will look that name up sequentially in the local, enclosing, global, and built-in scope.

Local (or function) scope

- It is the code block or body of any Python function or lambda expression.
- This Python scope contains the names that you define inside the function.
- These names will only be visible from the code of the function.
- It's created at function call, not at function definition, so we have as many different local scopes as function calls.
- It is applicable if we call the same function multiple times, or recursively.
- Each call will result in a new local scope being created.

Enclosing (or nonlocal) scope

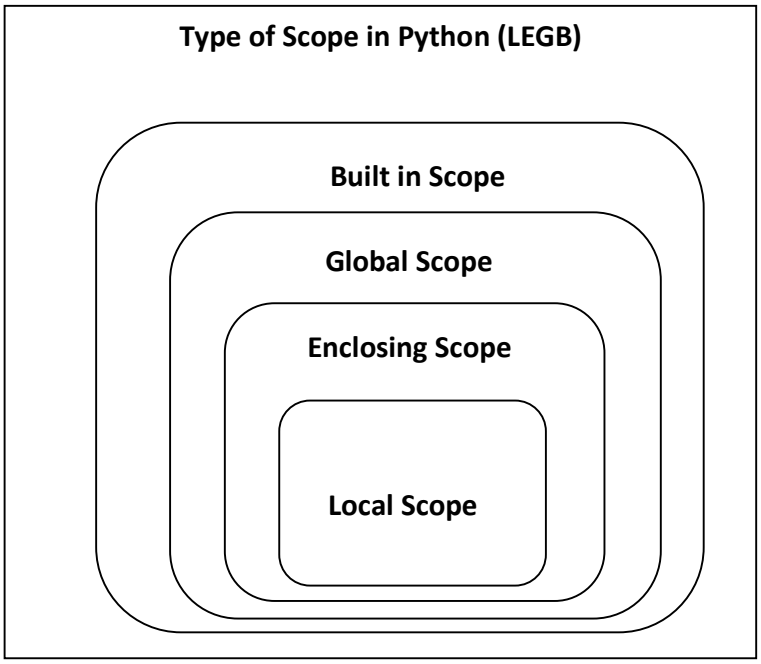
- It is a special scope that only exists for nested functions.
- If the local scope is an inner or nested function, then the enclosing scope is the scope of the outer or enclosing function.
- This scope contains the names that you define in the enclosing function.
- The names in the enclosing scope are visible from the code of the inner and enclosing functions.

Global (or module) scope

- It is the top-most scope in a Python program, script, or module.
- This scope contains all of the names that are defined at the top level of a program or a module.
- Names in this Python scope are visible from everywhere in your code.

Built-in scope

- It is a special scope which is created or loaded at script run or opens an interactive session.
- This scope contains names such as keywords, functions, exceptions, and other attributes that are built into Python.
- Names in this scope are also available from everywhere in your code.



Example

```
#var1 is in the global namespace
var1 = 5
def ABC( ):

  # var2 is in the local namespace
   var2 = 6

   def  XYZ( ):

       # var3 is in the nested local
       # namespace
       var3 = 7
```

- In this example, the **var1** is declared in the global namespace because it is not enclosed inside any function. So it is accessible everywhere in the script.
- **var2** is inside the ABC( ). So, it can be accessed only inside the ABC( ) and outside the function, it no longer exists.
- **var3** also has a local scope, the function is nested and we can use **var3** only inside XYZ( ).