



## Learning Objectives

After the completion of this chapter, the student will be able to

- Understand what is Scoping
- Able to implement the LEGB rule
- Understand what is module
- Understand the implementation of access control in programming language

### 3.1 Introduction

Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program. In other words, which parts of your program can see or use it. Normally, every variable defined in a program has global scope. Once defined, every part of your program can access that variable. But it is a good practice to limit a variable's scope to a single definition. This way, changes inside the function can't affect the variable on the outside of the function in unexpected ways.

### 3.2 Variable Scope

To understand the scope of variables in a programming language, it is important to learn about what variables really are. Essentially, they're addresses (references, or pointers), to an object in memory. When

you assign a variable with `:=` to an instance (object), you're binding (or mapping) the variable to that instance. Multiple variables can be mapped to the same instance.



#### Note

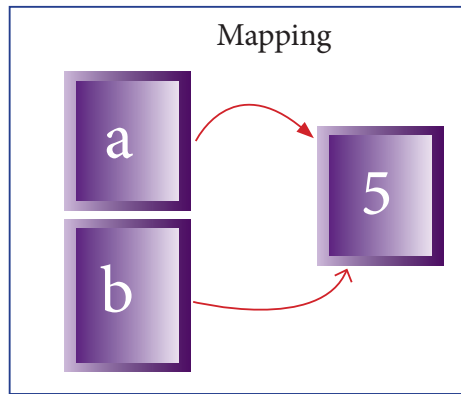
The process of binding a variable name with an object is called mapping. `=` (equal to sign) is used in programming languages to map the variable and object.

Programming languages keep track of all these mappings with namespaces. ***Namespaces are containers for mapping names of variables to objects.*** You can think of them as dictionaries, containing a list of words and their meanings. The words are mapped with their meaning in dictionaries, whereas names are mapped with objects (`name := object`) in programming language. This allows access to objects by names you choose to assign to them.

In the following example, ***a*** is first mapped to the integer ***5***. In this case, ***a*** is the variable name, while the integer value ***5*** is the object.

Then, ***b*** is set equal to ***a***. This actually means that ***b*** is now bound to the same integer value as ***a***, which is ***5***.

1. `a:=5`
2. `b:=a`

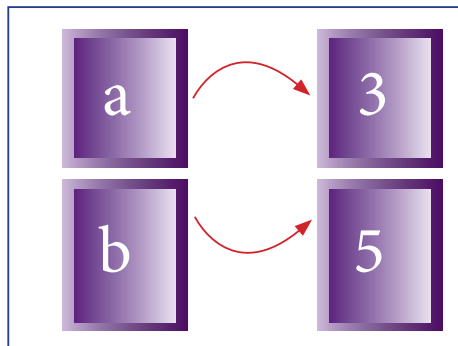


Now consider the following extension of the above statements

1. `a:=5`
2. `b:=a`
3. `a:=3`

If you then change *a* to be equal to 3, a budding programmer might expect *b* also be equal to 3, but that is not the case. *b* is still mapped (or pointing) to the integer value of 5. The only thing that changed is *a*, which is now mapped to the integer value 3.

Mapping after changing the value of *a*



**The scope of a variable is that part of the code where it is visible.** Actually, to refer to it, you don't need to use any prefixes then. Let's take an example,

1. `Disp():`
2. `a:=7`

When you try to display the value of *a* outside the procedure the program flags

the error “*name ‘a’ is not defined*”. This is because the lifetime of the variable is only till the end of the procedure. Also, *the duration for which a variable is alive is called its ‘life time’*.

### 3.3 LEGB rule ↩

Scope also defines the order in which variables have to be mapped to the object in order to obtain the value. Let us take a simple example as shown below:

1. `x:= 'outer x variable'`
2. `display():`
3. `x:= 'inner x variable'`
4. `print x`
5. `print x`
6. `display()`

When the above statements are executed the statement (4) and (5) display the result as

#### Output

outer x variable

inner x variable

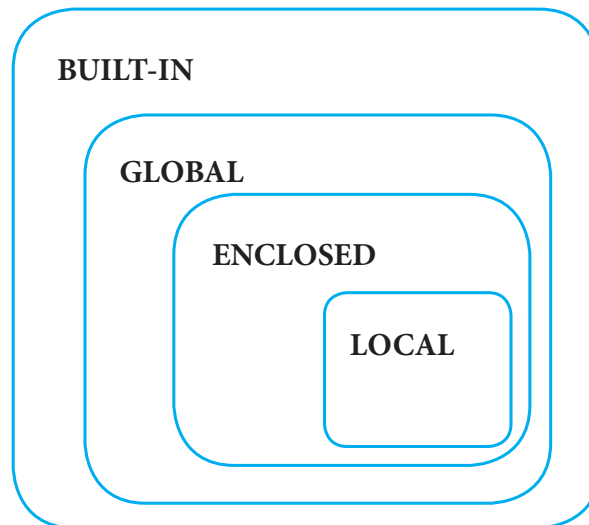
Above statements give different outputs because the same variable name *x* resides in different scopes, one inside the function `display()` and the other in the upper level. The value ‘*outer x variable*’ is printed when *x* is referenced outside the function definition. Whereas when `display()` gets executed, ‘*inner x variable*’ is printed which is the *x* value inside the function definition. From the above example, we can guess that there is a rule followed, in order to decide from which scope a variable has to be picked.

The **LEGB** rule is used to decide the order in which the scopes are to be searched for scope resolution. The scopes are listed



below in terms of hierarchy (highest to lowest).

Local(L)	Defined inside function/class
Enclosed(E)	Defined inside enclosing functions (Nested function concept)
Global(G)	Defined at the uppermost level
Built-in (B)	Reserved names in built-in functions (modules)



### 3.4 Types of Variable Scope

There are 4 types of Variable Scope, let's discuss them one by one:

#### 3.4.1 Local Scope

Local scope refers to variables defined in current function. Always, a function will first look up for a variable name in its local scope. Only if it does not find it there, the outer scopes are checked.

Look at this example

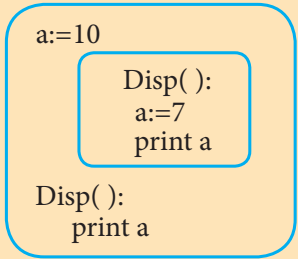
1. <b>Disp():</b>	Entire program	Output of the Program
2.   a:=7		7
3.   print a		
4. Disp()		



On execution of the above code the variable **a** displays the value 7, because it is defined and available in the local scope.

### 3.4.2 Global Scope

A variable which is declared outside of all the functions in a program is known as global variable. This means, global variable can be accessed inside or outside of all the functions in a program. Consider the following example

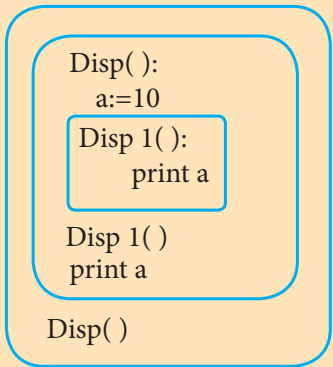
	Entire program	Output of the Program
1. a:=10		7 10
2. <b>Disp()</b> :		
3. a:=7		
4. print a		
5. Disp()		
6. print a		

On execution of the above code the variable **a** which is defined inside the function displays the value 7 for the function call Disp() and then it displays 10, because **a** is defined in global scope.

### 3.4.3 Enclosed Scope

All programming languages permit functions to be nested. A function (method) with in another function is called nested function. **A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.**

When a compiler or interpreter search for a variable in a program, it first search Local, and then search Enclosing scopes. Consider the following example

	Entire program	Output of the Program
1. Disp():		10 10
2. a:=10		
3. Disp1():		
4. print a		
5. Disp1()		
6. print a		
7. Disp()		

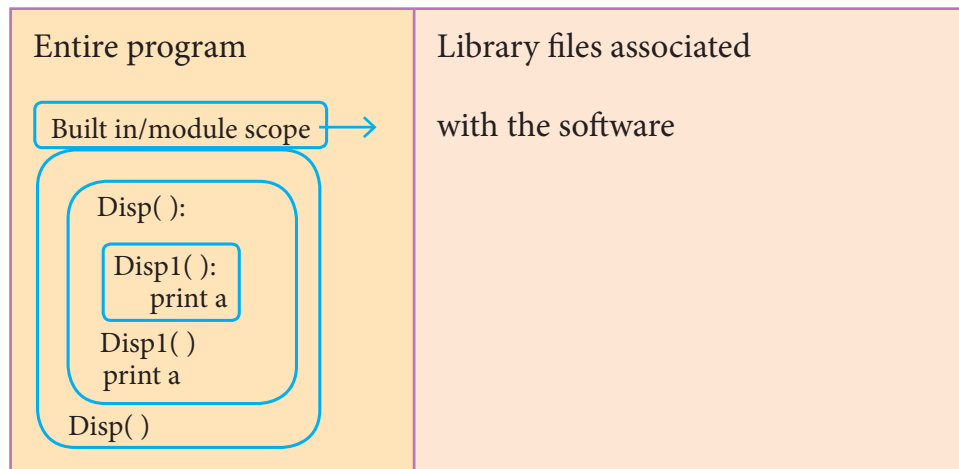




In the above example `Disp1()` is defined with in `Disp()`. The variable 'a' defined in `Disp()` can be even used by `Disp1()` because it is also a member of `Disp()`.

### 3.4.4 Built-in Scope

Finally, we discuss about the widest scope. The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter. Any variable or function which is defined in the modules of a programming language has Built-in or module scope. They are loaded as soon as the library files are imported to the program.



Normally only Functions or modules come along with the software, as packages. Therefore they will come under Built in scope.

## 3.5 Module

A module is a part of a program. Programs are composed of one or more independently developed modules. A single module can contain one or several statements closely related each other. Modules work perfectly on individual level and can be integrated with other modules. A software program can be divided into modules to ease the job of programming and debugging as well. A program can be divided into small functional modules that work together to get the output. The process of subdividing a computer program into separate sub-programs is called Modular programming. Modular programming enables programmers to divide up the work and debug pieces of the program

independently. The examples of modules are procedures, subroutines, and functions.

### 3.5.1 Characteristics of Modules

The following are the desirable characteristics of a module.

1. Modules contain instructions, processing logic, and data.
2. Modules can be separately compiled and stored in a library.
3. Modules can be included in a program.
4. Module segments can be used by invoking a name and some parameters.
5. Module segments can be used by other modules.

### 3.5.2 The benefits of using modular programming include

- Less code to be written.
- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand.
- Errors can easily be identified, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

### 3.5.3 Access Control

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the object. In other words access control is a selective restriction of access to data. IN Object oriented programming languages it is implemented through access modifiers. Classical object-

oriented languages, such as C++ and Java, control the access to class members by public, private and protected keywords. Private members of a class are denied access from the outside the class. They can be handled only from within the class.

Public members (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method. ***This arrangement of private instance variables and public methods ensures the principle of data encapsulation.***

Protected members of a class are accessible from within the class and are also available to its sub-classes. No other process is permitted access to it. This enables specific resources of the parent class to be inherited by the child class.

Python doesn't have any mechanism that effectively restricts access to any instance variable or method. Python prescribes a convention of prefixing the name of the variable or method with single or double underscore to emulate the behaviour of protected and private access specifiers.

All members in a Python class are public by default, whereas by default in C++ and java they are private. Any member can be accessed from outside the class environment in Python which is not possible in C++ and java.



### **Points to remember:**

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- The process of binding a variable name with an object is called mapping. `:` `=` (colon and equal to sign) is used in programming languages to map the variable and object.
- Namespaces are containers for mapping names of variables to objects.
- The scope of a variable is that part of the code where it is visible.
- The LEGB rule is used to decide the order in which the scopes are to be searched for scope resolution.
- Local scope refers to variables defined in current function.
- A variable which is declared outside of all the functions in a program is known as global variable.
- A function (method) within another function is called nested function.
- A variable which is declared inside a function which contains another function definition within it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.
- Built-in scope has all the names that are pre-loaded into program scope when we start the compiler or interpreter.
- A module is a part of a program. Programs are composed of one or more independently developed modules.
- The process of subdividing a computer program into separate sub-programs is called Modular programming.
- Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the object.
- Public members (generally methods declared in a class) are accessible from outside the class.
- Protected members of a class are accessible from within the class and are also available to its sub-classes.
- Private members of a class are denied access from the outside the class. They can be handled only from within the class.
- Python prescribes a convention of prefixing the name of the variable/method with single or double underscore to emulate the behaviour of protected and private access specifiers.
- C++ and Java, control the access to class members by public, private and protected keywords.
- All members in a Python class are public by default whereas by default in C++ and Java all members are private.

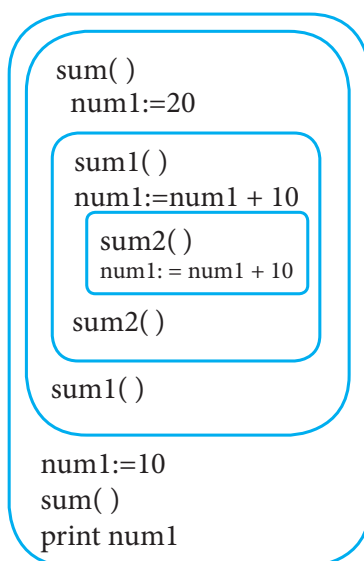






## Hands on Practice

1. Observe the following diagram and Write the pseudo code for the following



## Evaluation

### Part - I

Choose the best answer

(1 Mark)

1. Which of the following refers to the visibility of variables in one part of a program to another part of the same program.  
(A) Scope (B) Memory (C) Address (D) Accessibility
2. The process of binding a variable name with an object is called  
(A) Scope (B) Mapping (C) late binding (D) early binding
3. Which of the following is used in programming languages to map the variable and object?  
(A) :: (B) := (C) = (D) ==
4. Containers for mapping names of variables to objects is called  
(A) Scope (B) Mapping (C) Binding (D) Namespaces
5. Which scope refers to variables defined in current function?  
(A) Local Scope (B) Global scope  
(C) Module scope (D) Function Scope





6. The process of subdividing a computer program into separate sub-programs is called
  - (A) Procedural Programming
  - (B) Modular programming
  - (C) Event Driven Programming
  - (D) Object oriented Programming
7. Which of the following security technique that regulates who can use resources in a computing environment?
  - (A) Password
  - (B) Authentication
  - (C) Access control
  - (D) Certification
8. Which of the following members of a class can be handled only from within the class?
  - (A) Public members
  - (B) Protected members
  - (C) Secured members
  - (D) Private members
9. Which members are accessible from outside the class?
  - (A) Public members
  - (B) Protected members
  - (C) Secured members
  - (D) Private members
10. The members that are accessible from within the class and are also available to its sub-classes is called
  - (A) Public members
  - (B) Protected members
  - (C) Secured members
  - (D) Private members

## Part - II

Answer the following questions

(2 Marks)

1. What is a scope?
2. Why scope should be used for variable. State the reason.
3. What is Mapping?
4. What do you mean by Namespaces?
5. How Python represents the private and protected Access specifiers?

## Part - III

Answer the following questions

(3 Marks)

1. Define Local scope with an example.
2. Define Global scope with an example.
3. Define Enclosed scope with an example.



4. Why access control is required?
5. Identify the scope of the variables in the following pseudo code and write its output  
color:= 'Red'  
mycolor():  
    b:='Blue'  
    myfavcolor():  
        g:='Green'  
        print color, b, g  
    myfavcolor()  
    print color, b  
mycolor()  
print color

#### Part - IV

Answer the following questions

(5Marks)

1. Explain the types of scopes for variable or LEGB rule with example.
2. Write any Five Characteristics of Modules.
3. Write any five benefits in using modular programming.

#### REFERENCES

1. *Data Structures and Algorithms in Python* By Michael T. Goodrich, Roberto Tamassia and Michael H. Goldwasser.
2. *Data Structure and Algorithmic Thinking in Python* By Narasimha Karumanchi
3. <https://www.python.org>

