

3.SCOPING

Section – A

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

Section-B

Answer the following questions

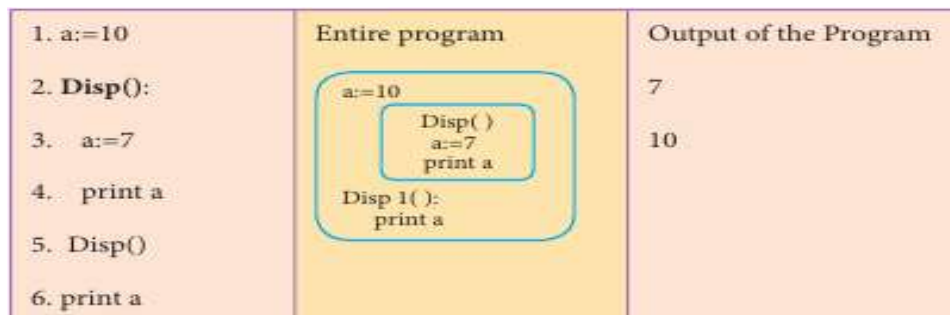
(2 Marks)

1. What is a scope?

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.

2. Why scope should be used for variable. State the reason.

- The scope should be used for variables because; it limits a variable's scope to a single definition.
- That is the variables are visible only to that part of the code.
- Example:



3. What is Mapping?

- 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.

4. What do you mean by Namespaces?

- Namespaces are containers for mapping names of variables to objects (name := object).

• Example: a:=5

- Here the variable 'a' is mapped to the value '5'.

5. How Python represents the private and protected Access specifiers?

- Python prescribes a convention of adding a prefix **__**(double underscore) results in a variable name or method becoming **private**.
- Example: `self.__n2=n2`
- Adding a prefix **_** (single underscore) to a variable name or method makes it **protected**.
- Example: `self._sal = sal`

Section-C

Answer the following questions

(3 Marks)

1. Define Local scope with an example.

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

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

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

2. Define Global scope with an example.

- A variable which is declared outside of all the functions in a program is known as global variable.
- Global variable can be accessed inside or outside of all the functions in a program.
- Example:**

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

- 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. Define Enclosed scope with an example.

- 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 searches for a variable in a program, it first search Local, and then search Enclosing scopes.

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

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

4. Why access control is required?

- 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 OOPS Access control is implemented through access modifiers.

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

```

OUTPUT:

Red Blue Green

Red Blue

Red

Scope of Variables:

Variables	Scope
Color:=Red	Global
b:=Blue	Enclosed
G:=Green	Local

Section - D

Answer the following questions:

(5 Marks)

1. Explain the types of scopes for variable or LEGB rule with example.

SCOPE:

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.

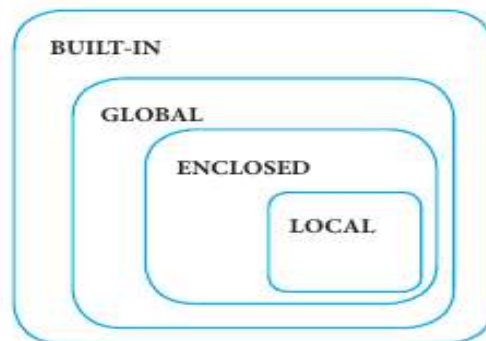
TYPES OF VARIABLE SCOPE:

- Local Scope
- Enclosed Scope
- Global Scope
- Built-in Scope

LEGB RULE:

- 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)



i) LOCAL SCOPE:

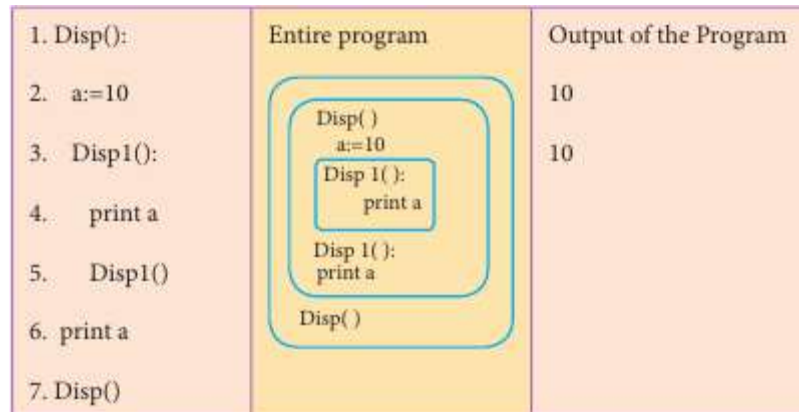
- Local scope refers to variables defined in current function.
- A function will always look up for a variable name in its local scope.
- Only if it does not find it there, the outer scopes are checked.
- **Example:**

1. Disp(): 2. a:=7 3. print a 4. Disp()	Entire program <pre>graph TD Entire[Entire program] -- contains --> DispCall[Disp()] DispCall -- contains --> DispDef["Disp(): a:=7 print a"]</pre>	Output of the Program 7
--	--	--------------------------------

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

ii) ENCLOSED SCOPE:

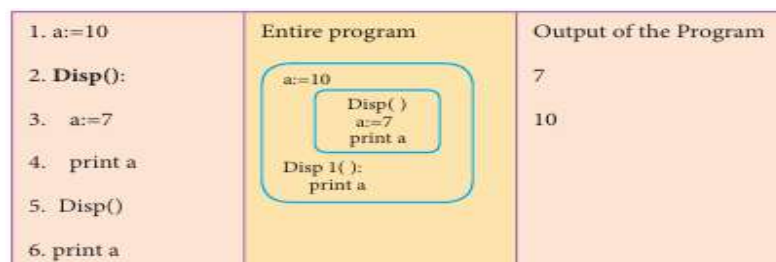
- A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.
- When a compiler or interpreter searches for a variable in a program, it first search Local, and then search Enclosing scopes.



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

iii) GLOBAL SCOPE:

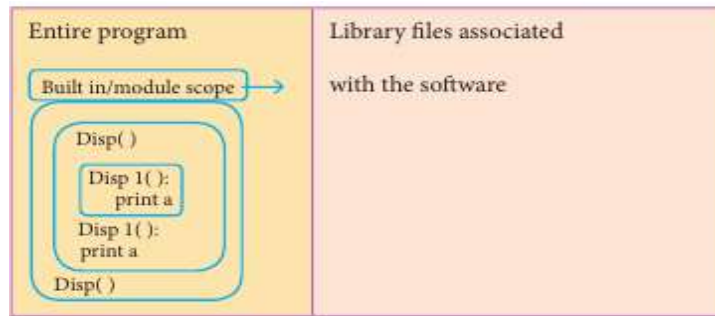
- A variable which is declared outside of all the functions in a program is known as global variable.
- Global variable can be accessed inside or outside of all the functions in a program.
- Example:**



- 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.

iv) BUILT-IN-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 module which is defined in the library functions of a programming language has Built-in or module scope.



2. Write any Five 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. Write any five benefits in using modular programming.

- 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 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.

PREPARED BY

J. BASKARAN M.Sc., B.Ed. (C.S)
jbaskaran89@gmail.com

J. ILAKKIA M.Sc., M.Phil., B.Ed. (C.S)
jilakkia@gmail.com