1. Why are functions advantageous to have in your programs?

Ans🡪 Functions enhance code organization, reusability, modularity, maintainability, readability, and scalability, making programs more efficient, manageable, and adaptable. They are a fundamental building block of software development that promotes good coding practices and enables the creation of complex programs through the composition of smaller, well-defined functions.

2. When does the code in a function run: when it's specified or when it's called?

Ans🡪 The code in a function runs when the function is called, not when it is specified. When you define a function, you are essentially creating a reusable block of code with a specific name and a defined set of instructions. To execute the code within a function, you need to call the function by using its name followed by parentheses ().

3. What statement creates a function?

Ans🡪 The def statement is used to create a function. The def statement is followed by the function name, a pair of parentheses (), and a colon :

def function\_name(parameters):

# Function body (code block)

# Optional return statement (if required)

# return value

4. What is the difference between a function and a function call?

Ans🡪 A function is a named block of reusable code that performs a specific task or a set of instructions. A function call, also known as a function invocation, is the act of executing a function. It is performed by using the function's name followed by parentheses (). When a function is called, the program flow jumps to the function's definition, executes the code inside the function's block.

5. How many global scopes are there in a Python program? How many local scopes?

Ans🡪 Global Scope: There is one global scope in a Python program, accessible throughout the entire program. Variables defined in the global scope are global variables.

Local Scopes: The number of local scopes depends on the number of functions or code blocks that define their own local scopes. Each function call creates a new local scope, and variables defined within those scopes are local variables specific to that scope.

6. What happens to variables in a local scope when the function call returns?

Ans🡪 Variables in a local scope are temporary and limited in scope to the specific function call. Once the function call returns, the local variables are no longer accessible, and the memory allocated for those variables is released. It's important to ensure that any data needed outside the function is returned appropriately or stored in a different scope (e.g., global scope or passed as a parameter).

7. What is the concept of a return value? Is it possible to have a return value in an expression?

Ans🡪 Yes, it is possible to have a return value in an expression. The return value can be used just like any other value within expressions, assignments, or any other appropriate context in the program.

def square(num):

return num \*\* 2

result = square(5) # Assigning the return value to a variable

print(result) # Output: 25

total = square(3) + square(4) # Using return values in an expression

print(total) # Output: 25

8. If a function does not have a return statement, what is the return value of a call to that function?

Ans🡪 If a function does not have a return statement, the return value of a call to that function is None. None is a special Python object that represents the absence of a value.

9. How do you make a function variable refer to the global variable?

Ans🡪 To make a function variable refer to a global variable, you can use the global keyword within the function. By using the global keyword, you can indicate that a variable inside the function should refer to the global variable with the same name.

Code-

x = 10 # Global variable

def modify\_global():

global x # Declare 'x' as a global variable within the function

x = 20 # Modify the global variable

print("Before function call:", x)

modify\_global()

print("After function call:", x)

Output-

Before function call: 10

After function call: 20

10. What is the data type of None?

Ans🡪 The data type of None in Python is NoneType. None is a special constant that represents the absence of a value or the lack of a specific object.

11. What does the sentence import areallyourpetsnamederic do?

Ans🡪 When you try to execute the statement import areallyourpetsnamederic, it will raise a ModuleNotFoundError because Python cannot find a module with that name.

12. If you had a bacon() feature in a spam module, what would you call it after importing spam?

Ans🡪

import spam

spam.bacon()

13. What can you do to save a programme from crashing if it encounters an error?

Ans🡪 To save a program from crashing when encountering an error, you can implement error handling techniques using exception handling. Exception handling allows you to catch and handle errors or exceptions that may occur during the execution of your program.

You can use a try-except block to handle exceptions. The code within the try block is monitored for any exceptions, and if an exception occurs, it is caught by the corresponding except block where you can specify how to handle the exception.

14. What is the purpose of the try clause? What is the purpose of the except clause?

Ans🡪 The purpose of the try clause in Python is to define a block of code where potential exceptions or errors might occur. The code within the try block is monitored for any exceptions, and if an exception occurs, the execution of the try block is immediately halted, and the control is transferred to the corresponding except block.

The purpose of the except clause is to define a block of code that handles the exceptions raised within the try block. When an exception occurs in the try block, the program flow is diverted to the corresponding except block.