1. **­­­Why are functions advantageous to have in your programs?**
2. Function is used to call a set of codes in a program to perform certain tasks wich will be used in the program again and again, to reduce the number of lines or reuse the same code again and again Functions are used,
3. Advantages are:
   1. **Modularity:**

* It means functions allow you to break down a program into smaller, reusable parts.
* We can define a function once and call it multiple times from different parts of our program.
  1. **Code Reusability:**
* Once function is defined and call we can reuse the code which we passed in that function clause we can reuse it by simply calling that functions
  1. **Readability and Maintainability**:
* Functions help in organizing and structuring the code and making it easier to read, understand, and maintain.
* By dividing the code into functions, we can focus on individual tasks or Functionalities.
  1. **Code Debugging**:
* Functions help in isolating and debugging issues since you can test and debug them separately from the rest of the program

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

* Here when we create function by using def keyword it starts accepting that below is the code which contains set of instruction/condition need to perform and it will not execute the code at the time of function creation, once we created a function and we need to call that function when it required in the project/program then only it will execute the specified conditions in between that function clause what we provide

1. **What statement creates a function?**

* in Python, the statement used to create a function is the def statement.
* The def keyword is followed by the function name, parentheses for specifying parameters (if any), and a colon.
* The code block that defines the function is indented below the def statement.
* Here's an example of a function creation in Python:
* **def greet():**
* **print("Hello, World!")**

1. **What is the difference between a function and a function call?**

* **Function**: it is used to create a function for reusable purpose but to use this function in our program we need to call it.
  1. Ex:

def greet():

print("Hello, World!")

* **Function call**: it is use a created function we need to use this call function
  1. Ex: great()

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

global scope is the outermost scope in a Python program. It exists throughout the entire program and is accessible from any part of the code. Variables defined in the global scope are considered global variables. Global variables can be accessed and modified by any function or code block within the program.

Local Scopes: Local scopes are created when a function or a code block is executed. Each function call or code block creates its own local scope. Local scopes are isolated from each other and from the global scope. Variables defined within a local scope are called local variables and are only accessible within that specific scope.

When a variable is accessed in Python, the interpreter first searches for it in the local scope. If the variable is not found in the local scope, it then looks for it in the enclosing scopes, moving from inner scopes to outer scopes, until it reaches the global scope. If the variable is not found in any of the scopes, a NameError is raised.

So, the number of global scopes in a Python program is one, and the number of local scopes depends on the number of function calls and code blocks executed during the program's execution. Each function call or code block creates its own local scope, and when that function call or code block completes, its local scope is destroyed.

1. **What happens to variables in a local scope when the function call returns?hGlobal Scope: The**
   * When a function call returns and completes its execution, the local scope associated with that function is destroyed. This means that the variables defined within the local scope are no longer accessible or usable once the function call returns.
   * Once the function call is completed, the local variables cease to exist, and any references to them will result in a NameError if accessed outside the function. The memory allocated for the local variables is freed up, and their values are no longer available for use.
2. **What is the concept of a return value? Is it possible to have a return value in an expression?**
   * The concept of a return value is the value that a function sends back to the caller. It allows a function to produce a result that can be used in expressions or assigned to variables. Yes, it is possible to have a return value in an expression, allowing you to directly use the result of a function call within an expression without explicitly assigning it to a variable.
3. **If a function does not have a return statement, what is the return value of a call to that function?**
   * If a function does not have a return statement, the return value of a call to that function is None.
   * Example:
   * def my\_function():
   * print("Function without return statement")
   * result = my\_function()
   * print(result) # Output: None
4. **How do you make a function variable refer to the global variable?**
   * To make a function variable refer to the global variable, you can use the global keyword followed by the variable name inside the function. Here's an example:

Ex:

x = 10 # Global variable

def my\_function():

global x # Declare x as a global variable

x = 20 # Modify the global variable

my\_function()

print(x) # Output: 20

1. **What is the data type of None?**
   * The data type of None in Python is NoneType. It represents the absence of a value or the lack of a return value.

Ex: result = None

print(type(result)) # Output: <class 'NoneType'>

1. **What does the sentence import areallyourpetsnamederic do?**
   * The sentence import areallyourpetsnamederic attempts to import a module named "areallyourpetsnamederic".
   * If such a module exists and is accessible, it will be imported, allowing access to its functionality.
   * If the module does not exist or is not accessible, a ModuleNotFoundError will be raised.
2. **If you had a bacon() feature in a spam module, what would you call it after importing spam?**

* If you had a bacon() function in a spam module, you would call it using the syntax spam.bacon() after importing the spam module.

Example:

import spam

spam.bacon()

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

* To save a program from crashing if it encounters an error, you can use exception handling. Place the code that may raise an error inside a try-except block. If an error occurs, it will be caught in the except block, allowing you to handle the error gracefully without the program terminating abruptly.
* Example:

try:

# Code that may raise an error

# ...

except Exception as e:

# Error handling code

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

* The purpose of the try clause is to enclose the code that might raise an exception, allowing you to catch and handle potential errors gracefully.
* The purpose of the except clause is to specify the code that should be executed if an exception occurs within the try block, providing an opportunity to handle the exception and prevent the program from crashing.
* By combining the try and except clauses, you can implement robust error handling in your code, ensuring that your program can handle unexpected situations and continue its execution without abrupt termination.