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

**Answer:**

* Functions reduce the need for duplicate code. This makes programs shorter, easier to read, and easier to update.

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

The code in a function executes when the function is called, not when the function is defined.

For example, consider the following code:

def greet():

print("Hello, World!")

print("Before function call")

greet()

print("After function call")

**Output**

Before function call

Hello, World!

After function call

*The code inside the function is executed only when the function is called, not when it is defined.*

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

Python, the statement used to create a function is called a "function definition" or "function declaration". In Python, the def statement is used to define a function. **def** statement is used to create a function by defining its name, parameters, and the code that defines its behavior.

def add\_numbers(a, b):

sum = a + b

print("The sum is:", sum)

In this example, the add\_numbers function takes two parameters, a and b. It calculates their sum and prints the result.

*Once the function is defined, it can be called or invoked later in the program by using its name followed by parentheses and passing appropriate arguments:*

add\_numbers(3, 5)

**Output:**

The sum is: 8

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

Answer: A function consists of the def statement and the code in its def clause. A function call is what moves the program execution into the function, and the function call evaluates to the function's return value.

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

In a Python program, there is only one global scope, and it is accessible throughout the entire program. The global scope refers to the outermost level of a program where variables, functions, and classes defined outside of any function or class are considered global and can be accessed from anywhere within the program.

On the other hand, local scopes are created whenever a function is called or a block of code (such as a loop or conditional statement) is entered. Each function call or code block creates a new local scope that is separate and isolated from other local scopes. Variables defined within a local scope are only accessible within that particular scope and its nested scopes.

global\_var = 10

def my\_function():

local\_var = 20

print(global\_var)

print(local\_var)

my\_function()

print(global\_var)

Output:

10

20

10

To summarize, a Python program has one global scope accessible throughout the program, and multiple local scopes are created whenever a function is called or a code block is entered.

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

When a function returns, the local scope is destroyed, and all the variables in it are forgotten.

def my\_function():

local\_var = 10

print(local\_var)

my\_function()

print(local\_var

If we try to print local\_var outside the function then it will throw the error

NameError: name 'local\_var' is not defined

**This error occurs because local\_var is a local variable that exists only within the local scope of the function and is not accessible in the global scope**

**7. 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 refers to the value that a function can send back to the caller once it has finished executing. When a function is called, it may perform certain operations, manipulate data, or calculate a result. The return value allows the function to communicate the result of its operation back to the caller

In Python, a function can use the return statement to specify the value that it wants to return. The return statement can be followed by an expression or a variable that represents the value to be returned

def add\_numbers(a, b):

sum = a + b

return sum

result = add\_numbers(3, 5)

print(result

Output = 0

So, the concept of a return value allows functions to produce a result and provide it to the caller, enabling the caller to use or manipulate that result further.

it is not possible to have a return value directly in an expression. The return statement is used to explicitly specify the value to be returned from a function, and it is used as a standalone statement within the function's body. However, you can assign the return value to a variable and use that variable in expressions as needed.

**8. 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, or if the return statement is not explicitly used in the function's code, the function will automatically return a special value called None.

***None*** is a built-in constant in Python that represents the absence of a value. It is often used to indicate that a function does not have a meaningful result or does not need to return anything specific.

def greet():

print("Hello, World!")

result = greet()

print(result)

In this example, the greet function does not have a return statement. When the function is called, it prints "Hello, World!" but does not return any value explicitly.

Output:

Hello, World!

None

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

To make a function variable refer to a global variable in Python, you can use the global keyword. When you define a variable as global within a function, it indicates that the variable should refer to the global variable with the same name, rather than creating a new local variable.

global\_var = 10

def my\_function():

global global\_var # Declare the variable as global

global\_var = 20 # Update the value of the global variable

print(global\_var) # Print the updated global variable

my\_function()

print(global\_var)

#Output

20

20

**10. What is the data type of None?**

The data type of None is actually a unique data type called NoneType. It is the only instance of this data type and can be compared to itself using the is operator.

result = None

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

print(result is None) # Output: True

**11. What does the sentence import areallyourpetsnamederic do?**

That import statement imports a module named areallyourpetsnamederic.

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

This function can be called with spam.bacon().

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

Place the line of code that might cause an error in a try clause.

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

Answer: In Python, the try-except statement is used to handle exceptions or errors that may occur during the execution of a program. The try block contains the code that might raise an exception, while the except block defines the actions to be taken if a specific exception occurs.

**try:**

**# Code that might raise an exception**

**result = 10 / 0 # Division by zero raises ZeroDivisionError**

**print("This line won't be executed")**

**except ZeroDivisionError:**

**# Handling the ZeroDivisionError exception**

**print("Cannot divide by zero")**