1. Why are functions advantageous to have in your programs

Ans1)Functions are advantageous in programs for several reasons:

Modularity: Functions allow you to break down a program into smaller, manageable pieces, each responsible for a specific task. This modular approach makes the code easier to understand, maintain, and debug.

Reuse: Once defined, functions can be reused multiple times within a program or across different programs. This promotes code reusability and reduces duplication, saving time and effort in development.

Abstraction: Functions provide a level of abstraction, allowing you to focus on what a piece of code does rather than how it does it. This makes code more readable and helps in understanding its functionality without needing to delve into implementation details.

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

Ans2)The code within a function runs when the function is called, not when it's specified.

When you define a function in your code, you are essentially defining a set of instructions that will be executed only when the function is invoked (called) elsewhere in the program. Until the function is called, its code is not executed.

Once the function is called, the program execution jumps to the body of the function, executes the code within it, and then returns to the point in the program where the function was called from.

3. What statement creates a function?

Ans3)In Python, for example, the def statement is used to define a function.

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

Ans4)Function:

* A function is a block of code that is defined to perform a specific task or set of tasks.
* It is essentially a reusable piece of code that encapsulates a particular functionality.
* Functions are defined using a declaration statement specific to the programming language being used (e.g., def in Python, function in JavaScript, etc.).
* When a function is defined, it establishes a blueprint or template for what actions should be performed when the function is called.
* Functions typically have a name, parameters (optional), and a body containing the code to be executed.

Function Call:

* A function call is an instruction that tells the program to execute the code contained within a particular function.
* It involves using the name of the function followed by parentheses () to signal to the program to execute the code defined within that function.
* Function calls provide a way to utilize the functionality defined in a function at specific points in a program.
* When a function is called, the program jumps to the function's definition, executes its code, and then returns to the point in the program where the function was called from.

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

Ans5)In Python, there is typically one global scope in a program. This global scope includes variables, functions, and classes defined at the outermost level of the program or module. Variables and other entities defined in the global scope are accessible from anywhere within the program.

Local scopes, on the other hand, are created whenever a function is called. Each function call creates its own local scope. Variables defined within a function are local to that function's scope and are only accessible from within that function unless explicitly passed or returned.

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

Ans6)When a function call returns, the local scope associated with that function is destroyed. Any variables defined within that local scope cease to exist, and their values are no longer accessible. This process is known as "variable cleanup" or "variable deallocation."

Here's what happens to variables in a local scope when a function call returns:

Variable Destruction: All variables defined within the local scope of the function are destroyed. This means that their memory is deallocated, and they are no longer accessible.

Scope Exit: The program flow exits the local scope of the function and returns to the caller's scope (which could be the global scope or another function's scope).

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

Ans7)The concept of a return value in programming refers to the value that a function sends back to the caller when the function is executed. When a function is called, it may perform some operations and then optionally return a value to the caller. This return value can be used by the caller for further computation, assignment to a variable, or any other purpose.

Example:-

def add(a, b):

return a + b

result = add(3, 5)

print(result) # Output: 8

In this example, the add() function takes two parameters a and b, adds them together, and then returns the result using the return statement. When the function is called with arguments 3 and 5, the return value 8 is assigned to the variable result, which is then printed.

Example:-

total = add(2, 3) + add(4, 5)

print(total) # Output: 14

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

Ans8)If a function does not have a return statement, Python implicitly returns None when the function is called.

For example:

def greet():

print("Hello")

result = greet()

print(result) # Output: None

In this example, the greet() function does not have a return statement. When the function is called, it prints "Hello" but does not explicitly return any value. Therefore, the value assigned to result is None. None is a special built-in constant in Python that represents the absence of a value.

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

Ans9)In Python, if you want to access or modify a global variable from within a function, you can use the global keyword to declare that the variable being referenced or modified is a global variable. This tells Python to look for the variable in the global scope rather than creating a new local variable with the same name.

10. What is the data type of None?

Ans10)In Python, None is a special constant representing the absence of a value or a null value. It is often used to indicate that a variable or expression does not have a meaningful value.

The data type of None is called NoneType.

Example:-

print(type(None)) # Output: <class 'NoneType'>

11. What does the sentence import areallyourpetsnamederic do?

Ans11)The sentence import areallyourpetsnamederic is a syntactically valid Python statement, but it doesn't have any predefined meaning in Python's standard library or common programming practice.

In Python, the import statement is used to import modules or packages, which are files containing Python code that can be reused in other Python programs. When you use import, Python looks for a module with the specified name and makes its functions, classes, and variables available for use in your current Python script or interactive session.

However, the sentence import areallyourpetsnamederic suggests importing a module named areallyourpetsnamederic, which is not a standard or commonly used module in Python. It seems more like a humorous or whimsical statement rather than a practical programming instruction.

If there were a module named areallyourpetsnamederic, its content would be imported, and you could use any functions, classes, or variables defined within that module in your Python script. But without such a module existing, this statement doesn't have any functional effect.

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

Ans12)

If you have imported the spam module in your Python script, and the bacon() function is a part of that module, you would call it using the dot notation like this:

Example:-

import spam

spam.bacon()

In this example, spam is the name of the module, and bacon() is a function defined within the spam module. By using the dot notation (spam.bacon()), you specify that you want to call the bacon() function from the spam module.

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

Ans13)To prevent a program from crashing when it encounters an error, you can implement error handling techniques such as using try-except blocks. Here's how you can do it in Python:

try:

# Code that might cause an error

# For example:

result = 10 / 0 # Division by zero will raise an error

except Exception as e:

# Handle the error gracefully

print("An error occurred:", e)

# You can choose to perform alternative actions, log the error, or ignore it

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

Ans14)The try clause in Python is used to enclose the code that might raise an exception, allowing you to handle potential errors gracefully. Its purpose is to execute the code block within it and monitor for any exceptions that might occur during execution.

The primary purposes of the try clause are:

Error Handling: The try clause provides a mechanism to catch and handle exceptions that might occur during the execution of the enclosed code block.

Preventing Crashes: By enclosing potentially error-prone code within a block, you prevent the program from crashing if an exception occurs.

The primary purposes of the except clause are:

Exception Handling: The except clause catches and handles exceptions raised within the associated try block.

Error Recovery: It allows you to implement error recovery strategies or provide alternative actions to be taken when an exception occurs.

global\_var =glo

access\_global() # Output: Inside function: 10