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

**Code Reusability:** Functions allow you to write a piece of code once and use it multiple times throughout your program. This means less repetition and more efficient code.

**Simplify Complex Problems:** By breaking down complex problems into smaller, manageable functions, you can tackle each piece individually, making your code easier to understand and maintain.

**Maintainability:** When you need to update or fix a part of your code, having functionality encapsulated in functions means you only need to make changes in one place, rather than scattered throughout your program.

**Readability:** Functions can make your code more readable by giving descriptive names to blocks of code. It’s easier to understand what’s happening when you see calculateTotal() rather than a long, mysterious block of code.

**Modularization:** Functions promote modular programming, allowing you to separate different parts of your program into distinct functions. This helps in organizing your code better and makes it easier to collaborate with others.

**Testing and Debugging:** Smaller, isolated functions are easier to test and debug than large, monolithic code blocks. You can test individual functions to ensure they work correctly before integrating them into the larger program.

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

The code in a function runs when it is called. Just defining or specifying a function only tells the program what it should do when the function is called, but it doesn't actually execute the code inside the function.

1. What statement creates a function?

The statement that creates a function is called a **function definition**. In most programming languages, you define a function by using a specific keyword followed by the function name, parentheses, and a code block enclosed in curly braces or indented under the function name.

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

The function is the blueprint or the plan.

The function call is the execution of that plan.

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

In a Python program, there is exactly **one global scope**. This is where variables that are defined at the top level of the script or module reside. The global scope exists from the beginning of the program until it terminates, and variables defined here are accessible from any part of the program.

Regarding local scopes, there can be **many**. Each time you call a function, a new local scope is created. This local scope exists only during the function’s execution and is destroyed once the function returns. Variables defined within this function are accessible only within that local scope.

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

When a function call returns, the local scope associated with that function call is cleaned up. This means that all variables in the local scope are destroyed and their memory is freed. To put it simply, local variables are temporary and only exist during the execution of the function. Once the function completes its task and exits, those variables are no longer accessible or needed, so they're removed from memory. It's a bit like a to-do list for a single errand: once the errand is done, the list is thrown away.

1. 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 in programming refers to the value that a function gives back to the caller when it completes its execution. Essentially, it's the result produced by the function. When you call a function, you may want it to process some data and provide you with an output. The return statement is used within the function to specify this output.

1. 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, what is the return value of a call to that function. When a function does not have a return statement, it implicitly returns None in Python. This means that even if you don't explicitly specify a return value, the function call will still provide a result, which is None.

1. 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. This keyword tells the function that the variable being used is the one defined in the global scope, not a new local variable.

1. What is the data type of None?

In Python, the data type of None is NoneType. It's a special type that represents the absence of a value or a null value. None

11. What does the sentence import a really our pets name deric do?

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

The import spam statement brings the entire spam module into your script.

You then access the bacon() function within the spam module using the dot notation spam.bacon().

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

One effective way to prevent a program from crashing when it encounters an error is by using **exception handling**. In Python, you can handle exceptions using the try, except, and optionally, finally blocks. This allows your program to catch and manage errors gracefully without terminating unexpectedly.

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

The try clause is used to wrap the code that you want to monitor for potential exceptions (errors). You place the code that might raise an exception within this block. If an exception occurs, the normal flow of execution stops and control is transferred to the corresponding except block.