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

Answer-

* **Reusability:** Functions allow us to define a block of code that is going to be executed multiple times from different parts of the program, reducing code duplication and promoting modularity.
* **Organization**: Functions help in structuring code by breaking it into smaller, more manageable pieces. This makes the program easier to read, understand, and maintain.
* **Abstraction:** Functions abstract away complex logic by encapsulating it into a single entity. This simplifies the usage and understanding of the code.
* **Code maintenance:** Functions make it easier to update or modify code since changes made within a function will affect all instances where the function is called.

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

The code inside a function runs when it is called. Function definitions serve as a blueprint or template for the code to be executed when the function is invoked or called. The function is executed and control jumps to the function's block of code when the function is called during program execution.

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

The def statement is used to create a function in Python. It is followed by the function name, parentheses, and a colon. The function block is indented and contains the code that will be executed when the function is called.

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

A function is a block of reusable code that performs a specific task when called. It is a defined procedure or action. On the other hand, a function call is the act of invoking or executing a function, triggering the code within the function to run and produce a result. In simpler terms, a function is a predefined action, while a function call is the actual usage of that action in the program.

**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. It is the outermost scope, accessible throughout the program. It contains global variables and functions that can be accessed from any part of the program. On the other hand, local scopes are created whenever a function is called. Each function call creates its own local scope, which is separate from the global scope. Local scopes contain local variables that are accessible only within the specific function or block in which they are defined.

**6. 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 is destroyed. Any variables defined within the local scope cease to exist, and their values are no longer accessible. Only the return value (if any) is passed back to the calling code.

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

The return value is the value that a function gives back or returns after its execution. It is the result of the function's computation, which can be used in further operations or assigned to a variable. Yes, it is possible to use a return value in an expression by directly using the function call within the expression or assigning the return value to a variable, which can then be used in the expression

**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, the return value of a call to that function is None. None is a special Python object that represents the absence of a value. It is commonly used to indicate that a function does not explicitly return a value.

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

To make a function variable refer to a global variable, you can use the global keyword before the variable assignment inside the function. This informs Python that the variable is intended to refer to a global variable rather than creating a new local variable

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

The data type of None is NoneType. It represents the absence of a value or the absence of a specific data type

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

The statement import areallyourpetsnamederic is syntactically valid but doesn't refer to any standard Python library or module. It appears to be a whimsical statement that could be used for entertainment or as an example of a playful import statement. It won't have any actual effect unless there is a module named "areallyourpetsnamederic" that you have created or installed.

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

import spam

spam.bacon()

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

Using try-except blocks: Surround the potentially problematic code with a try block and provide one or more except blocks to handle specific types of exceptions. This allows you to catch and handle exceptions gracefully without the program abruptly terminating.

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

The purpose of the try clause in Python is to define a block of code where exceptions might occur. It is followed by one or more except clauses, which specify how to handle specific exceptions that may be raised within the try block.

If an exception occurs within the try block, the execution jumps to the corresponding except block that matches the raised exception's type.

The except block handles the exception by executing the specified code, which may involve logging the error, displaying a user-friendly message, or taking appropriate corrective action.

By using try and except together, you can gracefully handle exceptions and prevent the program from crashing due to unhandled errors.