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

a. Reusability: Functions can be defined once and used multiple times in a program, reducing the amount of code that needs to be written.
b. Modularity: Functions can be used to break down a program into smaller, more manageable pieces, making it easier to develop and maintain.
c. Abstraction: Functions can hide complex implementation details from the user, providing a simpler interface for the user to work with.
d. Debugging: Functions can be tested and debugged independently of the rest of the program, making it easier to identify and fix errors.
e. Readability: Functions can make the code more readable and easier to understand by providing descriptive names for specific operations.

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

Ans: The code in a function runs when it is called, not when it is specified.

When a function is defined, the code inside the function is not executed. Instead, the code is only executed when the function is called.

When the function is called, the program jumps to the location of the function's definition and starts executing the code inside the function. Once the function finishes executing, the program returns to the point where the function was called and continues executing the rest of the program.

3. What statement creates a function?

Ans: The "def" statement creates a function in Python.

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

Ans: A function is a block of code that performs a specific task when it is executed, while a function call is the act of using or invoking the function to execute its code and return a value, if specified.

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

Ans: In a Python program, there is only one global scope, which is created when the program starts running. All the variables defined outside of functions are in this global scope.

On the other hand, there can be multiple local scopes in a Python program, which are created whenever a function is called. Each function has its own local scope, which contains the variables defined within that function and are accessible only within that function. Once the function is completed, the local scope is destroyed.

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

Ans: When a function call returns, the local scope of that function is destroyed, and all the variables defined within that scope are also destroyed. Any attempt to access those variables outside of the function will result in a NameError, as those variables are no longer defined. Therefore, variables defined in a local scope are temporary and exist only as long as the function is running.

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

Ans: The value that a function returns after it has finished running is referred to as a return value. The "return" statement in Python is used to provide the return value, which can be any object or data type.

A return value can absolutely exist in an expression. A function call can be a component of an expression in Python, and the result can be used as a value. For instance, the code below increases an integer variable's value by the function's returned value:

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

Ans: When a function is called without a return statement, None is returned from the call. In Python, a function implicitly returns None at the end of execution if the return statement is not used to explicitly return a value. As a result, if a function call is utilised in an expression or assigned to a variable, the outcome will be None.

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

Ans: Use the global keyword inside the function to declare the variable as a global variable to make a function variable refer to the global variable of the same name. By using the global variable instead of a new local variable with the same name, Python is instructed to do so.

10. What is the data type of None?
Ans: NoneType is the data type for None. The special value None in Python is used to denote a null value or the lack of a value. For functions that don't have a specific value to return, it is frequently used as the default return value. Python provides a built-in data type called NoneType that only has the word "None" as a value.

11. What does the sentence import areallyourpetsnamederic do?

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

spam.bacon()

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

```
    # code that might raise an exception

    x = 10 / 0

except:
    # code to handle the exception

    print("An error occurred.")
```

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

Ans: Python employs the try and except clauses to handle exceptions.

The try clause's main function is to contain code that might cause an exception. If an exception is raised while the code inside the try block is being run, control is then transferred to the except block.

The except clause's job is to deal with the exception that the try block raised. Only if an exception is raised in the associated try block is the code contained in the except block performed. The code that has to run in order to handle the exception is found in the unless block.