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

Functions provide several advantages:

- **Code Reusability**: Functions allow you to reuse code without rewriting it, which reduces redundancy and saves time.
- **Modularity**: Functions enable you to break down complex programs into smaller, manageable parts, making the code easier to understand and maintain.
- **Abstraction**: Functions allow you to hide implementation details, exposing only what is necessary. This helps in focusing on higher-level logic without getting bogged down by specifics.
- **Testing and Debugging**: Functions can be tested individually, making it easier to identify and fix bugs.

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

The code in a function runs when the function is **called**, not when it is defined (specified).

3. What statement creates a function?

A function is created using the **def** statement followed by the function name and parentheses. For

3. def my_function():

   # function code here

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

4.Function: A function is a block of code that performs a specific task and is defined using the def statement.

- **Function Call**: A function call is the execution of that function. It is made by using the function's name followed by parentheses, which may include arguments if required.

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

5.Global Scopes: There is **one global scope** in a Python program, which exists for the duration of the program's execution.

- **Local Scopes**: There can be **multiple local scopes**, as each function call creates a new local scope.

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

6.When a function call returns, the variables in the local scope are **destroyed** and no longer exist. Their values cannot be accessed outside the function.

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

7.A **return value** is the value that a function sends back to the caller when it completes its execution. Yes, it is possible to have a return value in an expression.

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

8.If a function does not have a return statement, it implicitly returns **None**.

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

9.You can use the **global** keyword inside the function to indicate that you want to refer to the global variable.

10. What is the data type of None?

The data type of None is **None Type**. It is a special constant in Python that represents the absence of a value.

11. What does the sentence import areallyourpetsnamederic do?

The statement import areallyourpetsnamederic attempts to import a module named areallyourpetsnamederic. If the module exists and is available in the Python path, it will load the module and make its functions and variables accessible for use. If the module does not exist, it will raise an Import Error.

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

12.You would call the bacon() function using dot notation like this:

```
import spam

spam.bacon()  # Calls the bacon() function from the spam module
```

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

To save a program from crashing due to an error, you can use **exception handling** with try and except blocks. This allows you to catch and handle exceptions gracefully.

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

- **try Clause**: The try clause is used to wrap code that might raise an exception. It allows you to specify a block of code to be tested for errors.
- **except Clause**: The except clause is used to define how to handle the exception if it occurs. It allows you to specify actions to take when an error is encountered in the try block.