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

ANS :- Functions are advantageous to have in programs for several reasons:

1. Modularity and Reusability: Functions allow you to break down complex tasks into smaller, modular units of code. This promotes code organization, readability, and maintainability. Additionally, functions can be reused in multiple parts of a program or in different programs altogether, reducing code duplication and promoting efficient development.

2. Abstraction and Encapsulation: Functions enable you to abstract away implementation details. By providing a well-defined interface (function signature) and hiding the internal workings, functions allow other parts of the program to use them without needing to understand the underlying complexity. This promotes encapsulation and information hiding, making programs easier to understand, test, and modify.

3. Code Reusability and Maintainability: Functions make it easier to reuse code. Instead of rewriting the same logic multiple times, you can define a function once and call it whenever needed. If a bug is found or a change is required, you can update the function in a single place, and the changes will propagate to all the places where the function is used. This enhances maintainability and reduces the likelihood of introducing errors during code modifications.

4. Readability and Understandability: Functions can improve the readability of code by providing well-named blocks of code that convey their purpose. Well-designed functions with descriptive names can make the program logic easier to understand, enhancing collaboration among developers and making it easier to troubleshoot and debug.

5. Testing and Debugging: Functions facilitate unit testing. Since functions encapsulate specific functionality, you can write focused test cases to verify their behavior. This enables easier testing and debugging of smaller units of code, leading to more reliable and robust software.

6. Code Organization and Scalability: Functions help in organizing code into logical blocks, making it easier to manage and navigate larger codebases. They enable you to structure programs in a modular manner, promoting scalability as new features or enhancements can be added as additional functions without overwhelming the main program.

Overall, functions contribute to improved code quality, maintainability, reusability, and readability, making programs more efficient, robust, and easier to develop and maintain over time.

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

ANS :- The code within a function runs when the function is called, not when it is specified. The function definition serves as a blueprint or template for the behavior of the function, but the actual execution of the code within the function occurs when the function is invoked or called

3. What statement creates a function?

ANS :- n Python, the def statement is used to create a function. The def statement is followed by the function name, parentheses for optional parameters, a colon (:), and an indented block of code that defines the function's behavior.

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

ANS :- Function: A function is a block of reusable code that performs a specific task. It is defined using the def statement and has a unique name, optional parameters, and a code block that specifies its behavior. Functions are created to encapsulate a specific set of instructions that can be executed whenever needed. They can accept input parameters and may return a value. 2. Function Call: A function call is the act of executing or invoking a function to perform its defined task. When a function call is made, the program transfers control to the function, and the code within the function is executed. Function calls are made using the function name followed by parentheses, optionally passing arguments as inputs to the function.

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

ANS :- In a Python program, there can be one global scope and multiple local scopes.

1. Global Scope:

- The global scope refers to the outermost level of the program, outside of any function or class.

- Variables defined in the global scope are accessible throughout the entire program, including inside functions and classes.

- There is only one global scope in a Python program.

2. Local Scopes:

- Local scopes are created whenever a function is called or when a class method is invoked.

- Each function or method call creates its own local scope, independent of other function or method calls.

- Variables defined within a local scope are only accessible within that specific scope.

- When the function or method execution completes, the local scope is destroyed, and the variables within it are no longer accessible.

- Local scopes are also created within comprehensions (such as list comprehensions or dictionary comprehensions) and generator expressions.

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

ANS :- When a function call returns in Python, the local scope associated with that function is destroyed, and the variables within that local scope cease to exist. This process is known as "scope cleanup" or "garbage collection". Here's what happens to variables in a local scope when the function call returns:

1. Variable Cleanup: All the variables defined within the local scope of the function are cleaned up. They are no longer accessible or usable once the function execution completes.

2. Memory Deallocation: The memory allocated for the local variables is deallocated, freeing up the resources that were used to store those variables. This memory can then be reused for other purposes within the program.

3. Name Reusability: Since the local variables are destroyed after the function call returns, the names of those variables become reusable. This means you can use the same variable names within other functions or in the global scope without any conflict.

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

ANS :- A return value is the value that a function can send back to the caller. It allows functions to produce output or results that can be utilized in the program's logic. Return values can be assigned to variables or used directly within expressions for further computation

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 reaches the end of its code block without encountering a return statement, it automatically returns None by default. This implies that the function call does not produce any meaningful result or output.

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

ANS :- In Python, if we want to make a function variable refer to a global variable, we can use the global keyword within the function. This allows us to explicitly specify that a variable inside the function should refer to the global variable with the same name.

10. What is the data type of None?

ANS :- The data type of None in Python is called NoneType. It represents the absence of a value or the lack of a specific object.

11. What does the sentence import areallyourpetsnamederic do?

ANS :- The sentence "import areallyourpetsnamederic" is not a valid Python import statement and would result in a ModuleNotFoundError in Python.

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

ANS:- If we have a bacon() function within a module named spam, we can call it after importing the spam module by using the module name followed by the function name. Here's an example of how we would call the bacon() function after importing the spam module:

import spam

spam.bacon()

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

ANS :- To prevent a program from crashing when it encounters an error, we can use error handling techniques to catch and handle exceptions. In Python, this can be achieved using a try-except block. The try-except block allows us to specify a section of code where we anticipate potential errors, and if any of those errors occur, we can gracefully handle them without the program terminating abruptly.

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

ANS :- The purpose of the `try` clause is to enclose code that may potentially raise an error or exception. It allows us to attempt the execution of this code while being prepared to handle any exceptions that might occur.The purpose of the `except` clause is to specify the type of exception we want to catch and handle. When an exception of the specified type occurs within the `try` block, the corresponding `except` block is executed, allowing us to handle the exception in a controlled manner.