Answers

1. Functions are advantageous in programming for several reasons:
   Modularity: Functions allow you to break down your code into smaller, manageable chunks, making it easier to understand and maintain.
   Reusability: Once defined, functions can be called multiple times from different parts of the program, promoting code reuse and reducing redundancy.
   Abstraction: Functions enable you to abstract away complex operations behind a simple interface. Users of the function don't need to know the internal implementation details; they only need to know how to use it.
   Readability: Using descriptive function names and modularizing code improves the readability of the program, making it easier for other developers to understand and collaborate.
   Debugging: Functions help in debugging by isolating errors to specific parts of the code. When a function is not working correctly, you can focus on fixing that specific function without affecting other parts of the program.

2. The code within a function runs when the function is called, not when it's specified. Defining a function merely creates a blueprint for its behavior, outlining the operations it will perform when invoked. The function's execution occurs only when it is called explicitly within the program.

3. The `def` statement is used to create a function in Python. It is followed by the function name and its parameters, if any, and ends with a colon (:). The function's code block is then indented and defines the actions the function will perform when called

4. A function is a block of reusable code that performs a specific task when called. It consists of a name, parameters (if any), and a block of code defining its behavior.

   A function call, on the other hand, is the act of invoking or executing a function to perform its defined task. It involves using the function's name followed by parentheses, optionally passing arguments (values) to the function if it expects parameters. When a function is called, the program execution temporarily jumps to the function's code block, executes it, and then returns to the point in the program where the function was called.

5. In a Python program, there is one global scope, which includes variables defined outside of any function.

Local scopes are created whenever a function is called, and they exist only during the execution of that function. Each function call creates its own local scope. Therefore, the number of local scopes depends on the number of function calls made during the program's execution. Once the function call ends, its local scope is destroyed, and the variables defined within it are no longer accessible.

6. 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. This means that variables defined within the function are no longer accessible outside the function once the function call returns.
7. The concept of a return value in Python refers to the value that a function sends back to the caller after its execution. It allows functions to compute results and pass them back to the part of the program that called the function. Yes, it is possible to have a return value in an expression. The return value of a function can be used as part of an expression, allowing it to be assigned to variables, used in calculations, or passed as arguments to other functions.
8. If a function does not have a return statement, the return value of a call to that function will be **`None`**
9. To make a function variable refer to a global variable, you can use the **`global`** keyword followed by the variable name within the function. This indicates that the variable being referred to is the one defined in the global scope.
10. The data type of `None` in Python is `NoneType`. It represents the absence of a value or a null value.
11. The sentence `import areallyourpetsnamederic` in Python does not have any inherent meaning or functionality. It will raise an `ImportError` because there is no module named `areallyourpetsnamederic` in the Python standard library or any known external libraries. It's often used humorously or as a placeholder in examples to demonstrate the syntax of the `import` statement.
12. If you had a **`bacon()`** function in a **`spam`** module, you would call it using dot notation after importing the **`spam`** module.
13. To prevent a program from crashing when it encounters an error, you can use exception handling techniques. In Python, you can use try-except blocks to catch and handle exceptions gracefully. By wrapping potentially error-prone code within a try block and specifying how to handle exceptions in the except block, you can ensure that the program continues running even if errors occur.
14. The try clause in Python is used to enclose code that may potentially raise an exception or an error during execution. It allows you to monitor and catch exceptions that might occur while executing the code within the try block.

The except clause, on the other hand, is used to handle exceptions that occur within the try block. If an exception occurs, Python looks for an appropriate except clause that matches the type of exception raised. If a matching except clause is found, the code within that except block is executed to handle the exception gracefully.