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

Ans :

Functions offer several advantages:

Code Reusability: You can write code once and use it multiple times in different parts of your program, avoiding redundancy and saving time.

Modularity: Functions break down complex programs into smaller, manageable units, making them easier to understand, maintain, and debug.

Organization: Functions help organize code logically, making programs easier to read and follow.

Encapsulation: Functions can encapsulate related code and data, creating well-defined modules that improve code maintainability and reduce coupling between program parts.

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

Ans : The code within a function doesn't run when it's defined. It only executes when the function is called. This allows you to define functions at the beginning of your program and call them later when needed.

3. What statement creates a function?

Ans : The def statement is used to create a function. Here's the basic syntax:

```
def function_name(parameters):

 # Function body (code to be executed)
```

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

Ans : **Function:** A block of code with a name that defines functionality and can be reused.

Function Call: The act of executing the code defined within a function. You provide the function with its arguments (if needed), and it performs the specified task.

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

Ans : **Global Scope:** There's only one global scope in a Python program. It encompasses all code outside any function definition.

Local Scope: Each function has its own local scope, only accessible within that function.

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

Ans : When a function call returns, local variables within that function are no longer accessible and are garbage collected (removed from memory) unless they are assigned to a global variable using the global keyword (not recommended).

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

Ans : **Return Value:** The data a function optionally sends back to the calling code after it finishes execution.

Return Value in Expressions: No, return values cannot directly be used within expressions. The function call itself evaluates to the return value, which can then be used in expressions.

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

Ans : If a function doesn't have a return statement, it implicitly returns None by default.

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

Ans : **Accessing Global Variables:** To make a function variable refer to a global variable with the same name, use the global keyword before assigning a value to the variable within the function. This is generally discouraged due to potential side effects and reduced code readability.

10. What is the data type of None?

Ans : The data type of None is NoneType, a special datatype indicating the absence of a value. It's often used as a return value to signal that no meaningful value is returned.

11. What does the sentence import areallyourpetsnamederic do?

Ans : **import areallyourpetsnamederic:**

This line imports a module named areallyourpetsnamederic. However, such a module likely doesn't exist in standard libraries or common repositories. It's essential to use modules that are available and relevant to your program's needs.

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

Ans : Calling bacon() after importing spam:

Import spam():

spam.bacon()

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

Ans : To prevent program crashes from errors, use try-except blocks. The try block contains the code you suspect might encounter errors. If an error occurs, the code within the except block is executed to handle the error gracefully, potentially providing informative messages or recovering from the error.

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

Ans : **try clause:** Contains code that might raise an exception.

except clause: Captures the exception and provides code to handle it.

Example :

```
try:
    # Try to convert user input to an integer
    num = int(input("Enter a number: "))
    # Perform some operation with the number
    result = 10 / num
    print("Result:", result)
except ValueError:
    # Handle the case where the user doesn't enter a valid integer
    print("Please enter a valid integer.")
except ZeroDivisionError:
    # Handle the case where the user enters 0
    print("Cannot divide by zero.")
```