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

Functions are advantageous to have in your programs for several reasons:

Modularity: Functions allow you to break down complex tasks into smaller, manageable pieces of code. This modularity makes the code easier to read, understand, and maintain.

Reusability: Once you define a function, you can use it multiple times throughout your program or even in other programs. This saves you from duplicating code.

Abstraction: Functions can hide the implementation details, providing a higher-level interface to the rest of the program. This makes it easier to work with and reason about the code.

Readability: Functions can give meaningful names to blocks of code, improving the readability and self-documentation of your program.

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

The code inside a function runs when the function is called, not when it is specified. When you define a function, you're essentially defining a set of instructions that will be executed only when you call that function by its name.

3. What statement creates a function?

def statement is used to creates a function for example : def func\_assignment()

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

A function is a block of code that performs a specific task and is defined using the def statement. It is like a blueprint that describes how the task should be done.

A function call, on the other hand, is when you invoke or execute the function to perform the task it defines. To call a function, you simply use the function's name followed by parentheses containing any required arguments.

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

In a Python program, there is only one global scope. It is the outermost scope and contains variables defined outside of any function or class.

As for local scopes, there can be multiple local scopes. Each function call creates a new local scope, and any variables defined within the function belong to that specific local scope.

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

When a function call returns, the local scope, along with all the variables defined within that function, is destroyed. Any data stored in those local variables is no longer accessible or retained.

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

The concept of a return value refers to the value that a function gives back (returns) after its execution. When a function reaches a return statement, it immediately stops executing, and the value specified after return is passed back to the caller. This allows the caller to receive and use the result of the function's operation.

Yes, it is possible to have a return value in an expression. For example, you can use a function call with a return value as part of an arithmetic expression, as an argument to another function, or for assignment to a variable.

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

it automatically returns None. None is a special built-in constant in Python that represents the absence of a value.

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

To make a function variable refer to the global variable (instead of creating a new local variable), you can use the global keyword within the function. This tells Python that the variable should be treated as a global variable.

10. What is the data type of None?

The data type of None is "NoneType." It is a special data type in Python that represents the absence of a value.

11. What does the sentence import areallyourpetsnamederic do?

That import statement imports a module named areallyourpetsnamederic.

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

If you had a bacon() function in a spam module, you would call it after importing spam using the dot notation:  
import spam  
spam.bacon()

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

To save a program from crashing when it encounters an error, you can use error handling techniques. In Python, this is typically done using the try, except, and optionally finally blocks. The idea is to wrap the potentially error-prone code inside the try block and specify how to handle specific types of errors in the except block.

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

The try clause in Python is used to enclose the code that might raise an exception. It allows you to test a block of code for errors.

The except clause is used to handle exceptions that occur within the try block. If an exception of the specified type occurs, the code inside the except block is executed, providing a way to handle the error gracefully.

try:

# Code that might raise an exception

result = 10 / 0 # This will raise a ZeroDivisionError

except ZeroDivisionError:

# Code to handle the ZeroDivisionError

print("Error: Cannot divide by zero.")