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

**Ans.** Functions are advantageous in programs because they allow for the modularization and organization of code, making it easier to understand, debug, and maintain. They also promote code reusability by allowing the same code to be used multiple times within a program or even in other programs. Additionally, functions can improve the performance of a program by allowing for the computation of a specific task to be performed only when necessary, rather than every time the code is executed.

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

**Ans.** The code in a function run when it’s called.

1. What statement creates a function?

**Ans.** The **‘def’** statement creates a function.

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

**Ans.** A function is a block of code that is defined by its name and is designed to perform a specific task. A function call, on the other hand, is the actual act of executing the function by passing arguments (if any) to it and obtaining a result. Simply put, a function is the blueprint, and the function call is when you use the blueprint to build something.

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

**Ans.** In a Python program, there is only one global scope. The global scope is the outermost scope and contains all the global variables and functions that are defined in the program.

However, there can be multiple local scopes within the global scope. A local scope is created when a function is called and contains all the variables and functions defined within the function. When the function returns, the local scope is destroyed. Each time a function is called, a new local scope is created. This allows for the creation of variables and functions with the same name in different scopes without any naming conflicts.

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

**Ans.** When a function returns, the local scope is destroyed, and all the variables in it are forgotten.

1. 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 or a method returns after it has finished executing. The concept of a return value allows for functions or methods to return a result to the caller, which can then be used for further processing or stored for later use.

No, it is not possible to have a return value in an expression. An expression is a piece of code that evaluates to a value, but it does not return a value like a function or a method does. An expression is used as part of a larger statement or expression, but it does not return a result on its own.

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

**Ans.** If a function does not have a return statement, the return value of a call to that function is ‘None’ in Python. This means that the function does not return any value, and the value of the expression that calls the function is None.

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

**Ans.** To make a function variable refer to a global variable, you can use the "global" keyword in the function. For example, if you have a global variable named "x" and you want to use it inside a function, you can declare "global x" at the beginning of the function and then use the variable "x" throughout the function, which will refer to the global variable.

1. What is the data type of None?

**Ans.** The data type of None in Python is None type.

1. What does the sentence import areallyourpetsnamederic do?

**Ans.** That import statement import a module named by areallyourpetsnamederic.

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

**Ans.** After importing spam I would call it as **spam.bacon().**

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

**Ans.** There are several things that can be done to save a program from crashing when it encounters an error:

1. Implement error handling techniques such as try-catch blocks to handle exceptions.
2. Debug the code and fix any bugs or issues.
3. Regularly save progress and keep backup copies of the code.
4. Use robust libraries and frameworks that are known to be reliable.
5. Conduct testing and simulations to identify potential problems before deployment.
6. Monitor the program's performance and address any issues promptly.

By taking these steps, it may be possible to prevent or mitigate the effects of errors and keep the program

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

**Ans.** The try clause is used to enclose a block of code that could raise an exception. If an exception is raised within the try block, the program execution will immediately jump to the beginning of the corresponding except clause.

The purpose of the except clause is to provide a way to handle exceptions that are raised in the try block. The code in the except clause will be executed if an exception is raised, allowing you to catch and handle the exception in a controlled manner, rather than letting the program terminate with an error. The except clause can also specify the type of exception that it is prepared to handle, allowing you to write separate handlers for different types of exceptions.