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

Functions in python provide several advantages:

Reusability: Functions can be called multiple times, allowing you to write code once and use it multiple times, making your code more organized and efficient.

Abstraction: Functions allow you to encapsulate a block of code, hiding the details from the user, making it easier to read and maintain the code.

Readability: Functions help break down large and complex programs into smaller and manageable blocks, improving the readability and understandability of the code.

Debugging: Functions can be individually tested and debugged, making it easier to find and fix bugs.

Modularity: Functions can be written, tested, and debugged independently, allowing you to develop and maintain complex programs more easily.

In short, functions help you write clean, maintainable, and efficient code, making it easier to develop and maintain complex programs.

2.When does the code in a function run: when it&#39;s specified or when it&#39;s called?

The code in a function in Python runs when the function is called. The function definition specifies the code that will be executed, but it is not executed until the function is invoked by calling its name followed by a set of parentheses. When the function is called, the code inside the function runs from top to bottom and then returns control back to the point where the function was called.

3. What statement creates a function?

The def statement is used to create a function in Python. The basic syntax for creating a function is:

def function\_name(parameters):

# function body

# statements

return value

Here, def is a keyword that indicates the start of a function definition, function\_name is the name of the function, parameters are the inputs to the function (which can be optional), and the indented code block after the : is the body of the function that specifies the actions that the function will perform. The return statement is used to return a value from the function (which can also be optional).

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

In Python, a function and a function call are two different things.

A function is a block of reusable code that performs a specific task. It is defined using the def keyword and has a name, a set of parameters (optional), and a body of code that specifies what the function does. The function definition is executed only once when the program is run, and the code inside the function is executed only when the function is called.

A function call, on the other hand, is an action that invokes a function. A function call is made by writing the function name followed by a set of parentheses that may contain arguments, if the function takes any. The function call triggers the code inside the function to be executed, and the returned value is stored in a variable or displayed on the screen, if the function has a return statement.

In short, a function is a piece of code that performs a specific task, while a function call is an action that invokes the function and runs the code inside it.

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

There is only one global scope in a Python program, which contains variables and functions that are defined outside of any function or class definition. Global variables and functions are accessible from anywhere in the program, including within functions.

There can be multiple local scopes in a Python program, one for each function or class definition. A local scope is created whenever a function or class definition is encountered, and contains variables and functions that are defined within that particular function or class definition. Local variables and functions are only accessible within the scope in which they are defined, and are not accessible from outside the scope.

It's important to note that each function call creates its own local scope, which is separate from the local scopes of other function calls. This is known as nested scoping. The innermost local scope takes precedence over outer scopes when resolving names, meaning that if a variable with the same name is defined in both an inner and an outer scope, the inner definition will be used.

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

When a function call in Python returns, the local scope that was created for that function call is destroyed, and any variables defined within the local scope are no longer accessible. This is known as scope termination or garbage collection, and is a feature of Python's memory management system.

The process of scope termination ensures that the memory used by local variables is freed up for other uses, avoiding memory leaks and improving the performance of the program. This is especially important in large programs, where many function calls and local scopes may be created and destroyed throughout the lifetime of the program.

It's important to note that local variables are distinct from global variables and variables in other local scopes. Global variables persist for the entire lifetime of the program, and variables in other local scopes persist for the lifetime of their respective scopes.

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

In Python, a return value is a value that is returned from a function to the code that called the function. The return value is specified using the return statement in the body of the function, and it can be any valid Python expression, including numbers, strings, lists, dictionaries, and more.

The return value can be used to pass information from the function back to the code that called it. This allows functions to perform a specific task and return a result, which can then be used by the calling code. The calling code can assign the returned value to a variable, use it in an expression, or simply ignore it, depending on the requirements of the program.

Yes, it is possible to have a return value in an expression. The return value of a function can be used in any expression, just like any other value. For example, you can use the return value of a function in a mathematical expression, in a conditional statement, or as an argument to another function call. This makes functions very flexible and powerful, allowing you to create complex programs by combining the results of multiple functions in different ways.

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

If a function in Python does not have a return statement, it is said to have a None return value. This means that when the function is called, it will not return any value back to the calling code.

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

To make a function variable refer to a global variable in Python, you need to use the global keyword in the function. The global keyword allows you to access and modify global variables from within a function.

10. What is the data type of None?

In Python, None is a special constant that represents the absence of a value. It is not considered to be a data type in the traditional sense, but it can be used as a value just like any other data type. For example, you can assign None to a variable, pass it as an argument to a function, or return it from a function.

In Python, the data type of None is NoneType.

11. What does the sentence import areallyourpetsnamederic do?

The sentence import areallyourpetsnamederic in Python attempts to import a module with the name areallyourpetsnamederic. If there is no module with that name, Python will raise an ImportError:

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 in Python, you would call it after importing the spam module

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

Use try and except statements: The try and except statements allow you to catch exceptions (errors) in your code and handle them gracefully. This way, you can prevent your program from crashing and provide a more user-friendly error message or alternative solution

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 define a block of code that you want to test for errors (also known as exceptions). If an error occurs while executing the code in the try block, the code in the except clause will be executed instead.

The purpose of the except clause is to handle the exception that was raised in the try block and provide a way to handle the error in a graceful manner. The except clause should contain code that either resolves the error, or provides a user-friendly message about what went wrong.