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

**Ans:**

Functions are advantageous in programs because they promote code reusability, modularity, and readability. They allow us to break down complex tasks into smaller, manageable units, making code easier to understand, maintain, and debug. Functions also enable efficient testing and facilitate collaboration among developers.

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

**Ans:**

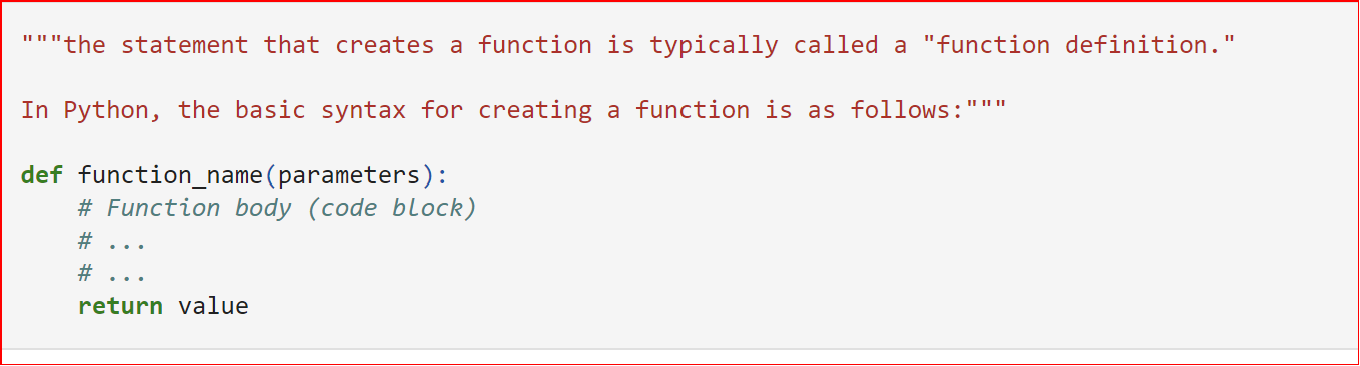
The code within a function runs when the function is called. When you define a function, you are essentially creating a reusable block of code with a specific name and a set of instructions. However, these instructions are not executed until the function is called or invoked in your program.

When you call a function by its name followed by parentheses, the program transfers control to that function, and the code inside the function's body is executed. The function may take input parameters, which can be used within the function's code to perform certain operations or calculations. Once the function has finished executing its code, it may return a value or perform some action, and the program resumes from the point where the function was called.

In summary, the code in a function runs when it is called or invoked during the program's execution, not when it is specified during the function definition.

**3. What statement creates a function?**

**Ans:**



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

**Ans:**

The difference between a function and a function call lies in their respective roles and behaviors within a program.

Function: A function is a block of reusable code that performs a specific task or a set of instructions. It encapsulates a series of statements and can accept input parameters (arguments) and return a result (output). Functions are typically defined with a name, a set of parameters (optional), and a body of code. They are created to be called or invoked at a later time when their functionality is needed.

Function Call: A function call is an instruction or statement that executes a specific function at a particular point in the program. It invokes the function and provides any necessary arguments or inputs. When a function call is encountered during program execution, the control flow jumps to the corresponding function definition, executes the code within the function body, and returns back to the point of the call once the function completes its execution.

In summary, a function is a defined set of instructions that performs a specific task, while a function call is the act of invoking or executing that function at a particular point in the program. Function calls allow you to use the functionality provided by a function when and where it is needed.

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

**Ans:**

Python programs have one global scope, which represents the outermost level of the program. It encompasses the entire program and is accessible from anywhere within the program. In addition to the global scope, local scopes are created whenever a function or a code block (such as a loop or conditional statement) is executed. Each function or code block has its own local scope, which is separate from the global scope. Local scopes are temporary and exist only during the execution of the corresponding function or code block.

In summary, there is one global scope and multiple local scopes in a Python program.

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

**Ans:**

When a function call returns in Python, the local scope associated with that function is destroyed. This means that the variables defined within the local scope are no longer accessible or available for use outside of the function. The variables in the global scope or other higher-level scopes are not affected by the destruction of local variables. They remain in their respective scopes and can still be accessed and modified.

When a function call returns, the variables in the local scope of the function are destroyed, and their values are no longer accessible. The program resumes execution in the calling scope, and any returned values can be utilized if specified.

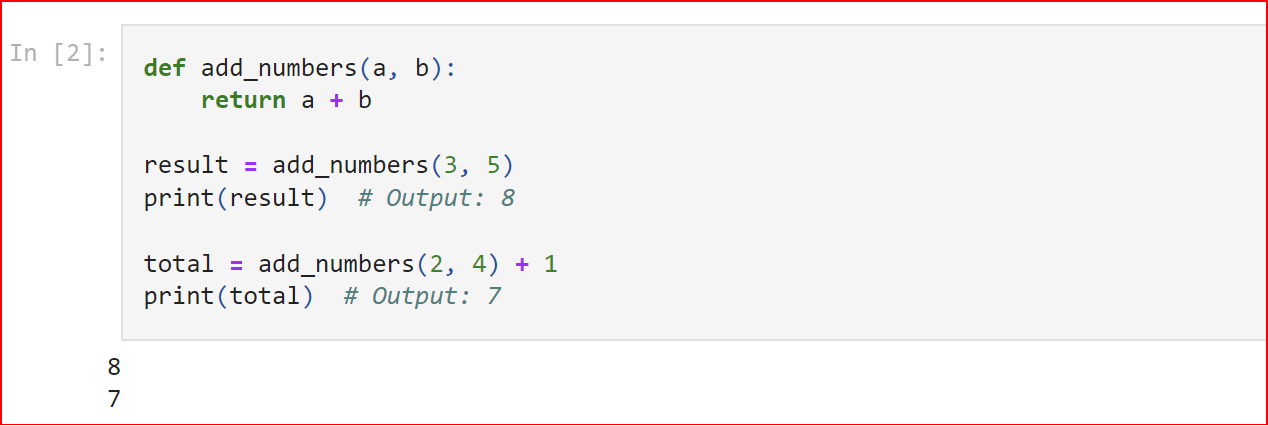
**7. What is the concept of a return value? Is it possible to have a return value in an expression?**

**Ans**:

The concept of a return value in programming refers to the value that a function can send back to the caller. When a function is executed and reaches a return statement, it can optionally specify a value to be returned. This return value provides a way for the function to communicate information or a result back to the code that called it.

A return value can be of any data type, including integers, strings, booleans, lists, or even more complex data structures. It allows functions to perform a computation or operation and then pass the result back to the caller, enabling the caller to use or further process that result as needed.

Yes, it is possible to have a return value in an expression. In Python, you can assign the return value of a function directly to a variable or use it as part of an expression. Here's an example



**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 does not have a return statement, or if the return statement is not reached during the execution of the function, the function call will still produce a return value. In Python, the return value in such cases is None.

None is a special object in Python that represents the absence of a value. It is often used to indicate that a function has no specific result to return or that a variable doesn't have a value assigned.

Here's an example:

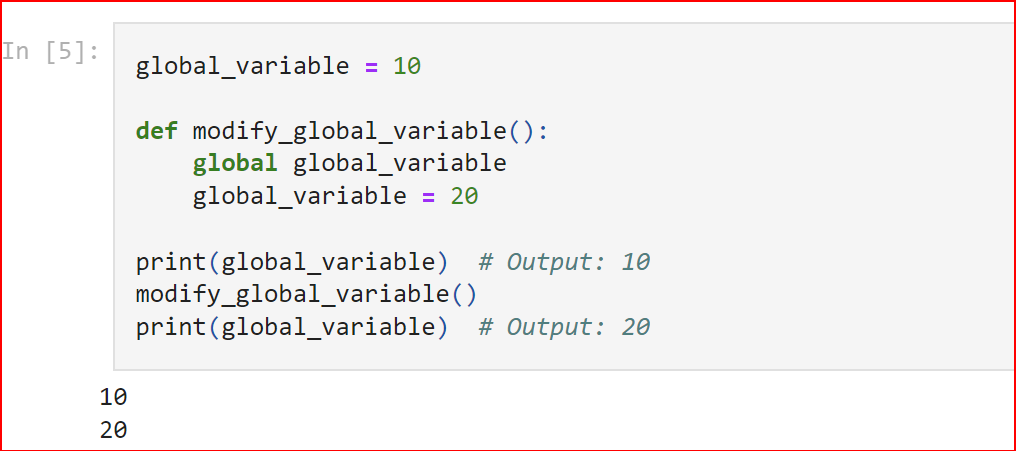


**9. How do you make a function variable refer to the global variable?**

**Ans:**

To make a function variable refer to the global variable of the same name, we can use the global keyword in Python. The global keyword allows you to indicate that a variable inside a function should refer to the same variable in the global scope.

Here's an example to demonstrate how to make a function variable refer to the global variable:



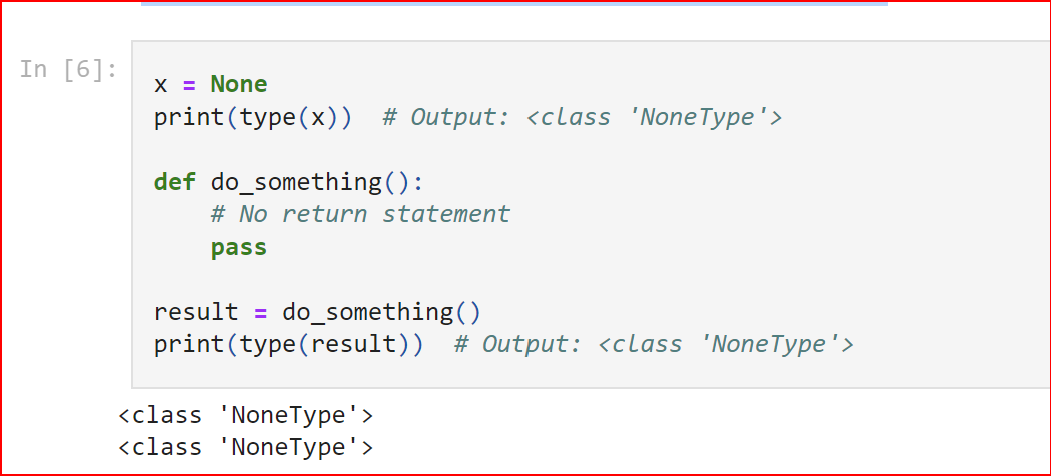
**10. What is the data type of None?**

**Ans:**

The data type of None is a special type called NoneType. It represents the absence of a value or the lack of any specific data.

The None value is often used to indicate that a variable or function doesn't have a value or doesn't return a meaningful result. It is commonly used as a placeholder or default value.

Here's an example that demonstrates the data type of None:



**11. What does the sentence import areallyourpetsnamederic do?**

**Ans:** 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?**

**Ans:**

After importing the spam module, you can access the bacon() feature by using the following syntax:

import spam

spam.bacon()

spam is the module that contains the bacon() feature. By prefixing bacon() with spam., you can call the bacon() function and execute the code within it.

Importing the spam module makes all the functions, classes, or variables defined in the module accessible using the module name as a prefix. This allows you to organize and access related functionalities within a module without conflicting with other names in your program.

Remember to replace bacon() with the actual name of the feature or function provided by the spam module that you wish to call.

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

**Ans:**

To save a program from crashing if it encounters an error, you can use error handling techniques such as try-except blocks to catch and handle exceptions. By encapsulating potentially problematic code within a try block and providing appropriate error handling in the except block, we can prevent the program from abruptly terminating and gracefully manage errors.

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

ANS:

The purpose of the try clause in Python is to enclose a block of code where exceptions might occur. It allows you to monitor and execute the code normally. If an exception is raised within the try block, the program flow is transferred to the except clause.

The purpose of the except clause is to define the actions to be taken when a specific exception occurs. It specifies the exception type to handle, and if an exception of that type is raised in the try block, the corresponding except block is executed to handle the exception. The except clause provides an opportunity to catch and handle exceptions, preventing program crashes and allowing for error-specific actions