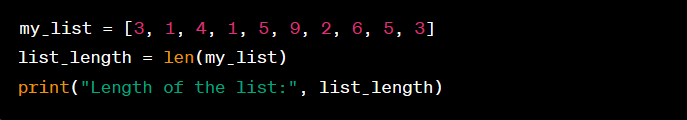
1. **In Python, what is the difference between a built-in function and a user-defined function? Provide an example of each.**

Ans: *In Python, the primary difference between a built-in function and a user-defined function is their origin and purpose:*

*1. Built-in Function:*

* *Origin: Built-in functions are functions that are included in Python's standard library, and they are available for use without the need for additional coding. These functions are part of the Python language itself.*
* *Purpose: Built-in functions are general-purpose functions that provide essential functionality, such as mathematical operations, string manipulation, list manipulation, input/output, and more.*
* *Examples:*
* *`len()`: Returns the number of items in a container (e.g., a list, string).*
* *`print()`: Outputs content to the console.*
* *`max()`: Returns the maximum value from a sequence.*
* *`str()`: Converts an object into a string representation.*

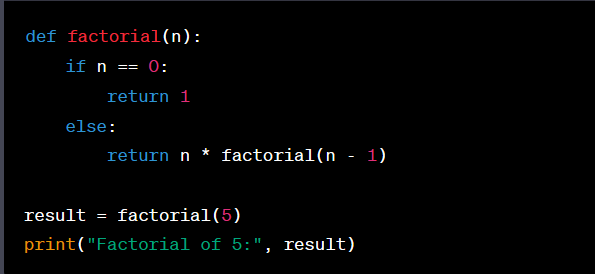
*Example of using a built-in function:*

**

*2. User-Defined Function:*

* *Origin: User-defined functions are functions created by a programmer to perform a specific task or set of tasks. These functions are defined by the user within their Python code.*
* *Purpose: User-defined functions allow programmers to encapsulate a block of code into a reusable function, promoting code modularity, reusability, and organization. They can be tailored to specific needs and provide custom functionality.*
* *Examples:*

*Here's an example of a user-defined function that calculates the factorial of a number:*

**

*In this example, `factorial()` is a user-defined function that calculates the factorial of a number. You can call it with different values to compute the factorial.*

1. **How can you pass arguments to a function in Python? Explain the difference between positional arguments and keyword arguments**.

Ans:

|  |  |
| --- | --- |
| *Positional Arguments:*   * *Positional arguments are the most common way of passing arguments to a function.* * *You pass arguments in the order in which the function expects them, and the function assigns the values to its parameters based on their position.* * *The order and number of arguments must match the function's parameter list.* * *Example of using positional arguments:*     *In this example, "Alice" is passed as the first argument and "Hello" as the second argument.* | *Keyword Arguments:*   * *Keyword arguments allow you to pass arguments by specifying the parameter names, along with their values.* * *This method provides more clarity and flexibility as the order of arguments doesn't matter.* * *You can skip some parameters or provide them out of order if you specify their names.* * *Example of using keyword arguments:*     *In this example, we use keyword arguments to specify the parameter names when calling the greet function. The order of the arguments doesn't matter in this case.* |

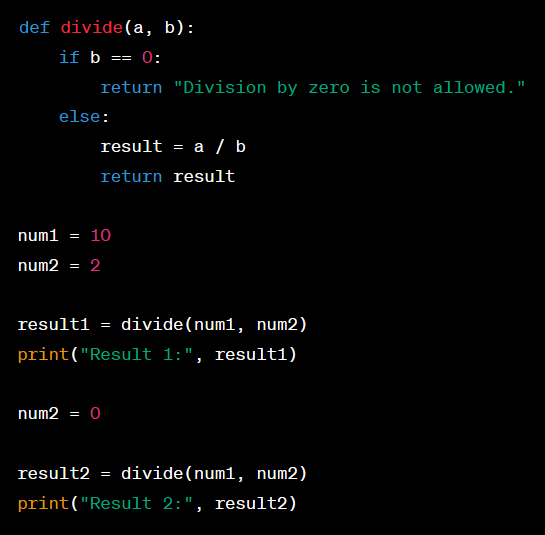
1. **What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.**

Ans: *The `return` statement in a function serves the purpose of specifying the value(s) that the function should return when it's called. It allows a function to produce a result that can be used or manipulated in the code that called the function. The primary purposes of the `return` statement are as follows:*

* *Returning Values: The `return` statement specifies the value or values that the function should return to the caller. These values can be of any data type, including numbers, strings, lists, or other objects*
* *Exiting the Function: When a `return` statement is encountered in a function, it immediately exits the function, and the control flow returns to the point in the code from where the function was called.*
* *Passing Data Back: It allows you to pass data back to the calling code. The calling code can then store or use the returned value(s)*

*Yes, a function can have multiple `return` statements. The first `return` statement that is executed in the function will cause the function to exit and return the specified value. Subsequent `return` statements will not be executed.*

*Here's an example of a function with multiple `return` statements:*



1. **What are lambda functions in Python? How are they different from regular functions? Provide an example where a lambda function can be useful.**

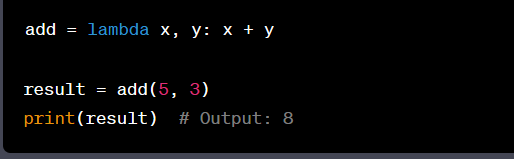
**Ans**: *Lambda functions, also known as anonymous functions or lambda expressions, are a concise way to create small, unnamed functions in Python. They are defined using the `lambda` keyword, followed by a list of arguments, a colon `:`, and an expression. Lambda functions are typically used for simple operations that can be expressed in a single line of code. The general syntax of a lambda function is:*



Key differences between lambda functions and regular (or named) functions:

* Anonymous Nature: Lambda functions are anonymous, meaning they don't have a name like regular functions. They are often used for short, one-time tasks and are not reusable across the codebase.
* Simplicity: Lambda functions are concise and designed for simple operations, making them ideal for functions that don't require extensive code.
* No Statements: Lambda functions can only contain a single expression, not multiple statements. Regular functions can have multiple statements and more complex logic.

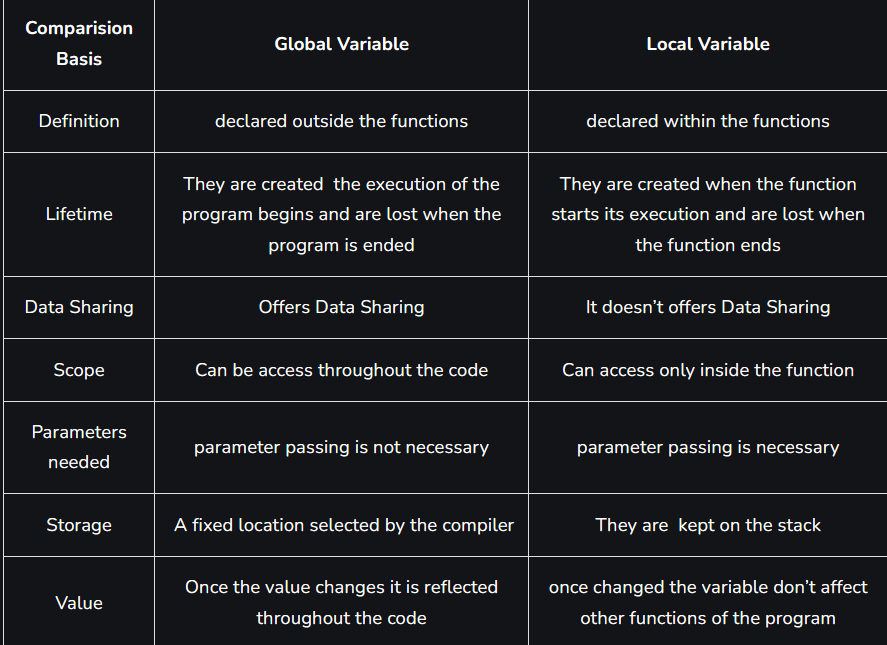
Here's an example of a lambda function that adds two numbers:



1. **How does the concept of "scope" apply to functions in Python? Explain the difference between local scope and global scope**.

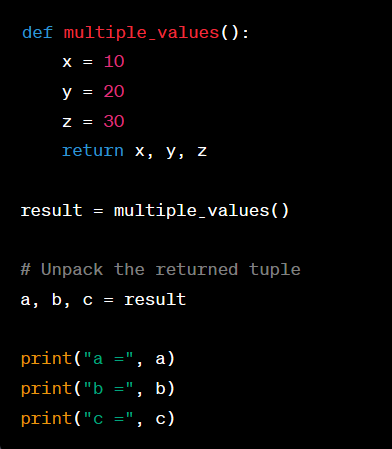
Ans: Scope: *The location where we can find a variable and also access it if required is called the scope of a variable.*

*In Python, "scope" refers to the region or context in which a variable is accessible and can be used. Python has several types of scopes, but the most common ones are local scope and global scope, which apply to functions*



1. **How can you use the "return" statement in a Python function to return multiple values?**

**Ans**: *In Python, you can use the return statement in a function to return multiple values as a tuple. This allows you to return multiple pieces of data from a function, and you can then access and unpack these values when the function is called. Here's how you can do it:*



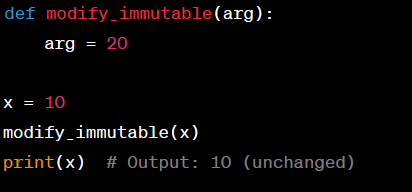
1. **What is the difference between the "pass by value" and "pass by reference" concepts when it comes to function arguments in Python?**

Ans: *In Python, it's important to understand that function arguments are neither strictly "pass by value" nor "pass by reference." Instead, Python uses a mechanism called "pass by object reference" or "call by sharing." This means that when you pass an argument to a function, you are passing a reference to the object in memory, and changes made to the object within the function can affect the original object outside of the function. However, the behavior may differ based on the mutable or immutable nature of the object.*

Here's a breakdown of how this works:

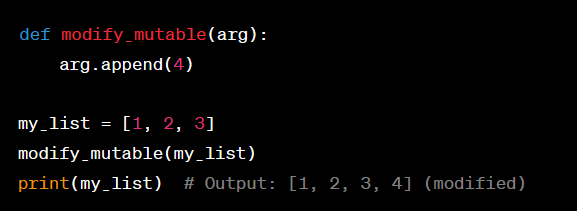
1. Immutable Objects (Pass by Value-Like Behaviour):

* Immutable objects, such as numbers, strings, and tuples, cannot be changed after they are created.
* When you pass an immutable object as an argument to a function, you cannot modify the original object. Any changes made within the function create new objects and do not affect the original.

Example: 

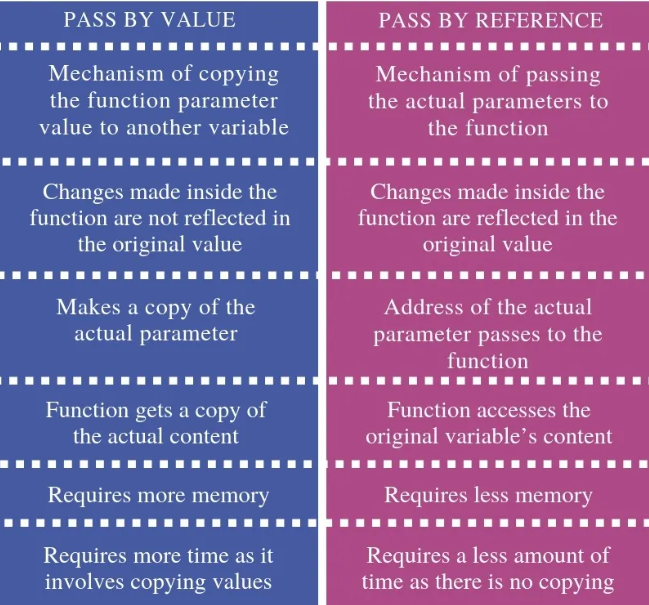
2. Mutable Objects (Pass by Reference-Like Behaviour):

* + Mutable objects, such as lists and dictionaries, can be changed after they are created.
  + When you pass a mutable object as an argument to a function and modify it within the function, those changes are reflected in the original object outside of the function because you are working with the same object reference.

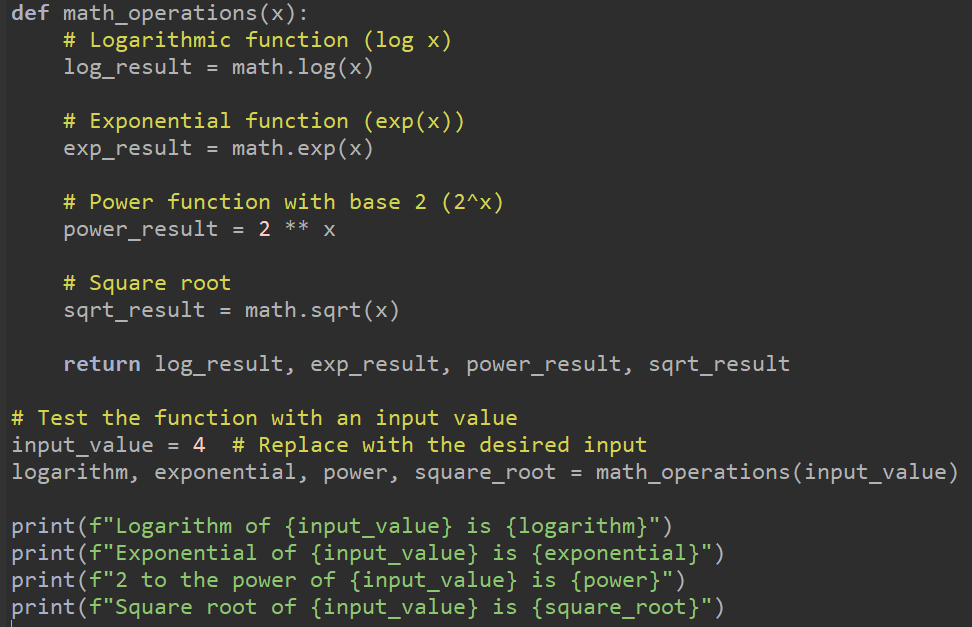
Example: 

*So, in Python, whether it appears as "pass by value" or "pass by reference" depends on the mutability of the object being passed. Immutable objects exhibit "pass by value"-like behaviour, while mutable objects exhibit "pass by reference"-like behaviour.*

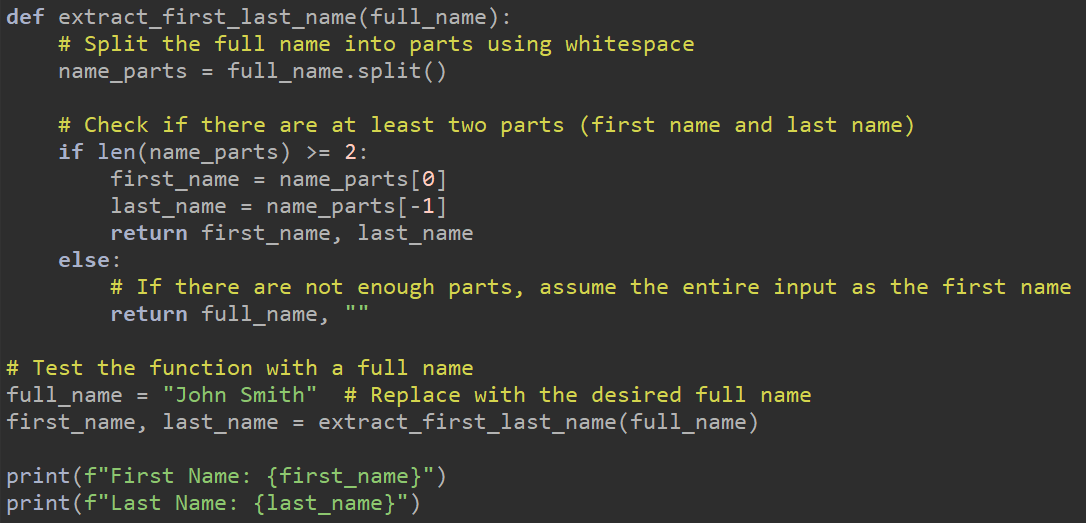
*It's important to understand this behaviour because it can affect the behaviour of your functions and how you manipulate data. If you want to avoid modifying an object passed as an argument, consider creating a new object within the function. If you want to modify it, work directly with the object reference.*



1. **Create a function that can intake integer or decimal value and do following operations:**
   1. **Logarithmic function (log x)**
   2. **Exponential function (exp(x))**
   3. **Power function with base 2 (2x)**
   4. **Square root**

**Ans:** 

1. **Create a function that takes a full name as an argument and returns first name and last name.**

**Ans:** ****