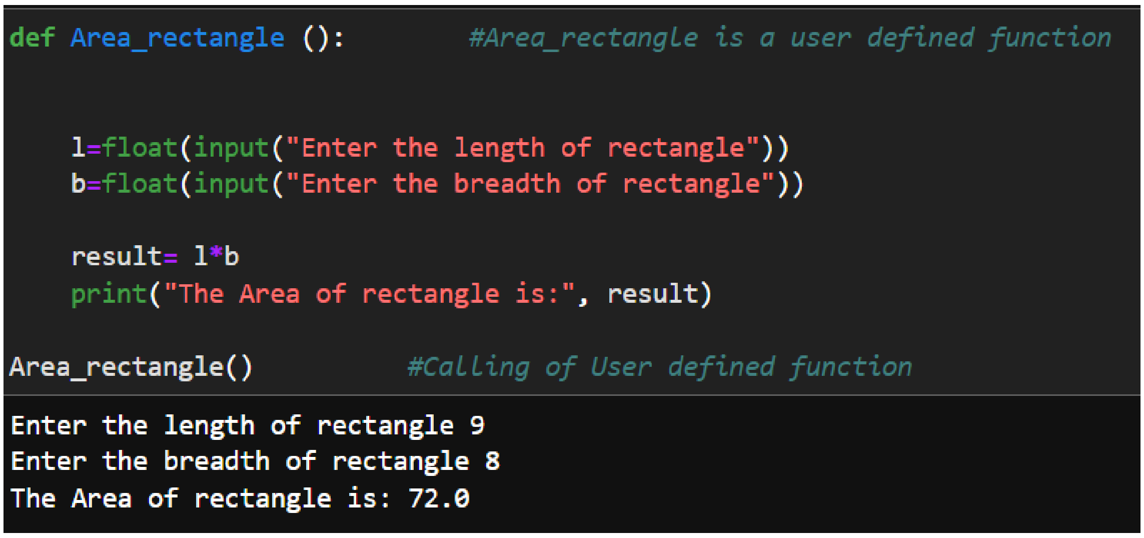
Python Assignment 8

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

   Ans: A programmer creates a function according to the requirement of a program, which is called a user-defined function.

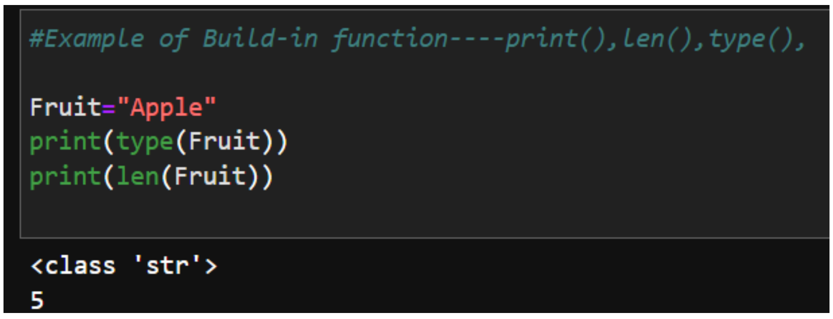
   A user- defined function is required to write the complete code before using the function in a program.



```
def Area_rectangle ():        #Area_rectangle is a user defined function


    l=float(input("Enter the length of rectangle"))
    b=float(input("Enter the breadth of rectangle"))

    result= l*b
    print("The Area of rectangle is:", result)

Area_rectangle()           #Calling of User defined function
```

```
Enter the length of rectangle 9
Enter the breadth of rectangle 8
The Area of rectangle is: 72.0
```

The predefined functions in python which can be used anytime are known as built-in functions. Ex: len(), type()

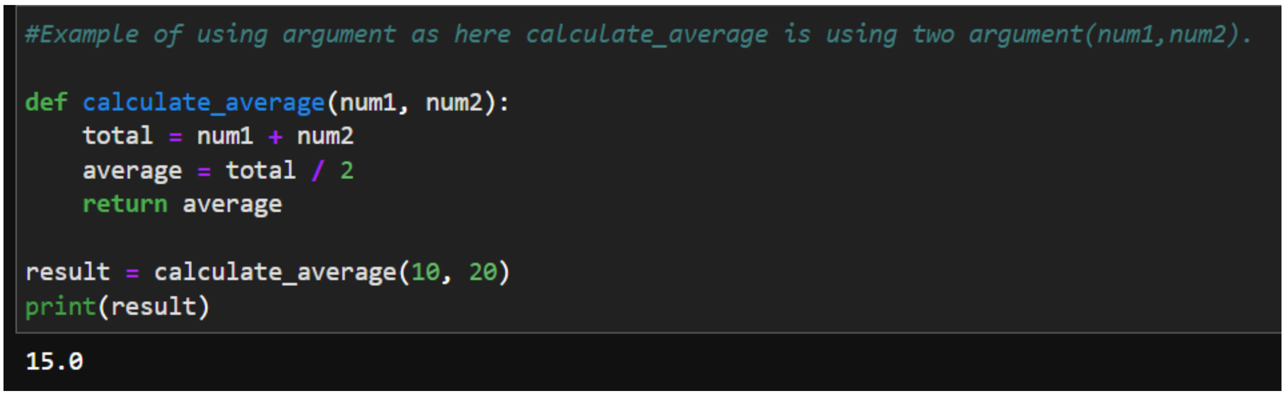


```
#Example of Build-in function----print(),len(),type(),

Fruit="Apple"
print(type(Fruit))
print(len(Fruit))
```

```
<class 'str'>
5
```

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

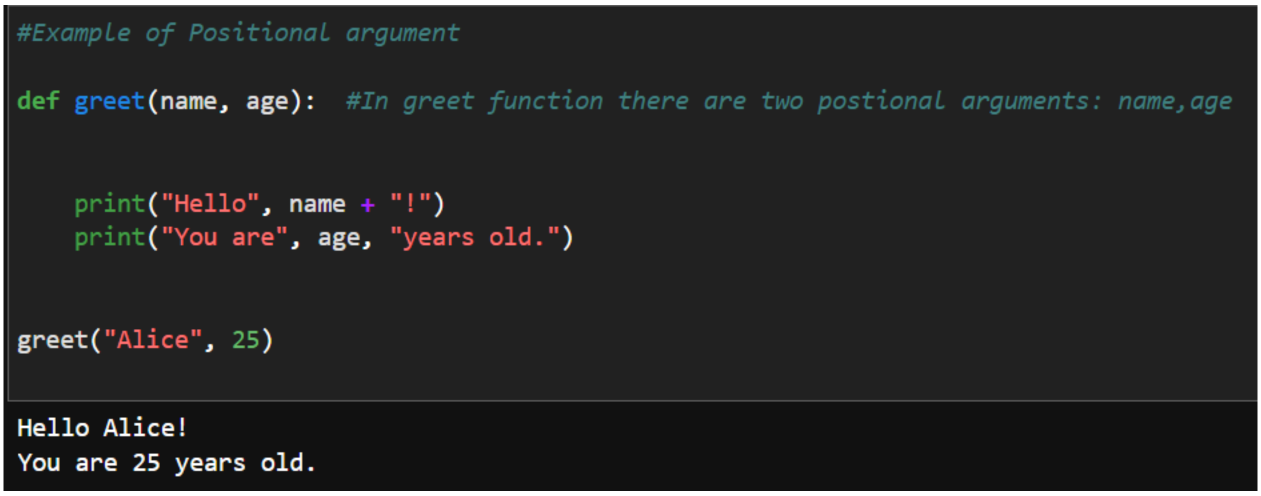
Ans: Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.



```
#Example of using argument as here calculate_average is using two argument(num1,num2).

def calculate_average(num1, num2):
    total = num1 + num2
    average = total / 2
    return average

result = calculate_average(10, 20)
print(result)
```

```
15.0
```

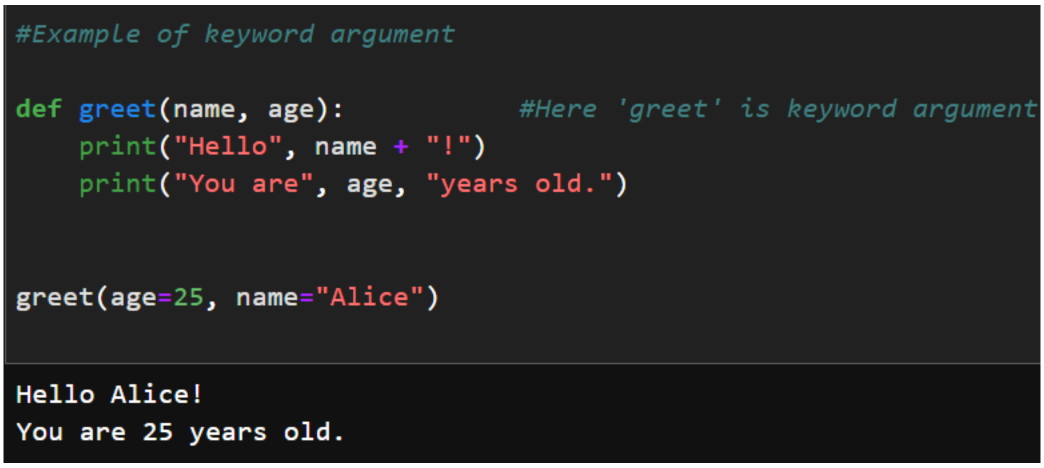
Position arguments are passed to a function based on their position in the function call. The order in which the argument is passed is important because it determines which argument corresponds to which parameter in the function definition. A positional argument is a name that is not followed by an equal sign (=) and default value.



```
#Example of Positional argument

def greet(name, age):  #In greet function there are two postional arguments: name,age


    print("Hello", name + "!")
    print("You are", age, "years old.")


greet("Alice", 25)
```

```
Hello Alice!
You are 25 years old.
```

Keyword arguments are passed to a function using the name of the parameter as a key-value pair. The order of the arguments doesn't matter, as long as they are correctly associated with their corresponding parameters. A keyword argument is followed by an equal sign and an expression that gives its default value.



```
#Example of keyword argument

def greet(name, age):           #Here 'greet' is keyword argument
    print("Hello", name + "!")
    print("You are", age, "years old.")


greet(age=25, name="Alice")
```

```
Hello Alice!
You are 25 years old.
```

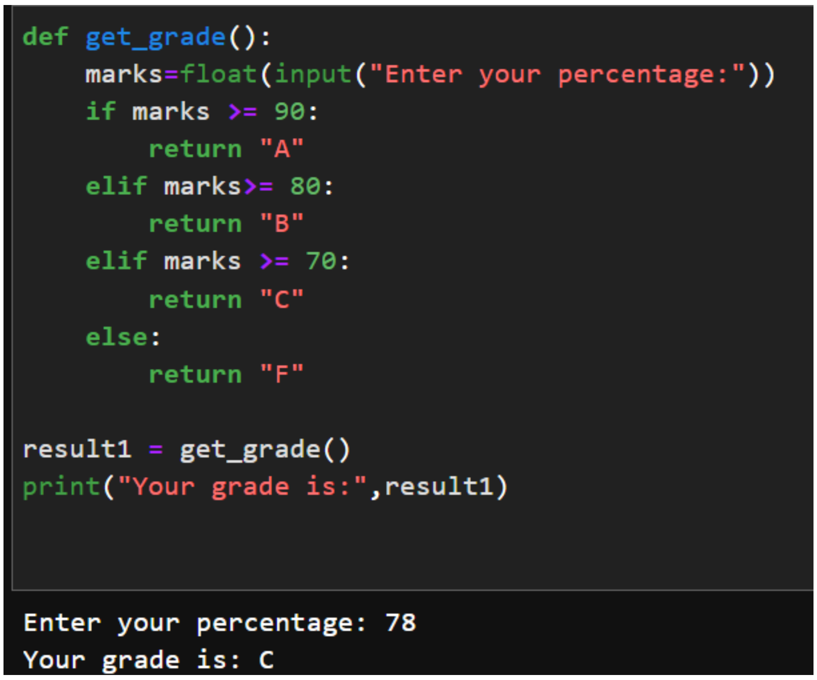
3. **What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.**

Ans: The **return** statement in a function is used to specify the value that the function should return when it is called. It allows the function to produce a result or output that can be used in the program's logic or stored in a variable for further use.

Two main purposes of **'return'** statement are:

1. It terminates the execution of the function and immediately returns control back to the caller.
2. It specifies the value or values that the function should return to the caller.

A function can have multiple '**return'** statements, but only one **return** statement is executed during the function call. Once a '**return'** statement is encountered, the function exits, and the specified value is returned.



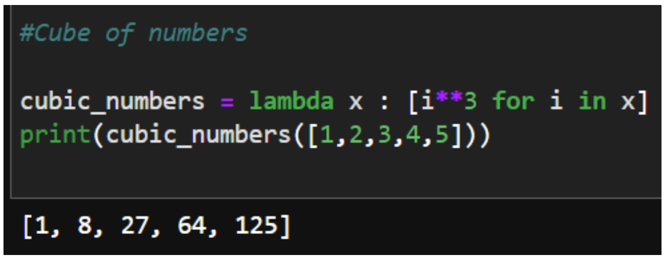
```python
def get_grade():
    marks=float(input("Enter your percentage:"))
    if marks >= 90:
        return "A"
    elif marks>= 80:
        return "B"
    elif marks >= 70:
        return "C"
    else:
        return "F"

result1 = get_grade()
print("Your grade is:",result1)
```

```
Enter your percentage: 78
Your grade is: C
```

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

Ans:

In Python, lambda functions, also known as anonymous functions, are small, inline functions that can be defined without a formal name. They are created using the ‘lambda’ keyword and are commonly used for simple, one-line operation.



```python
#Cube of numbers

cubic_numbers = lambda x : [i**3 for i in x]
print(cubic_numbers([1,2,3,4,5]))
```

```
[1, 8, 27, 64, 125]
```

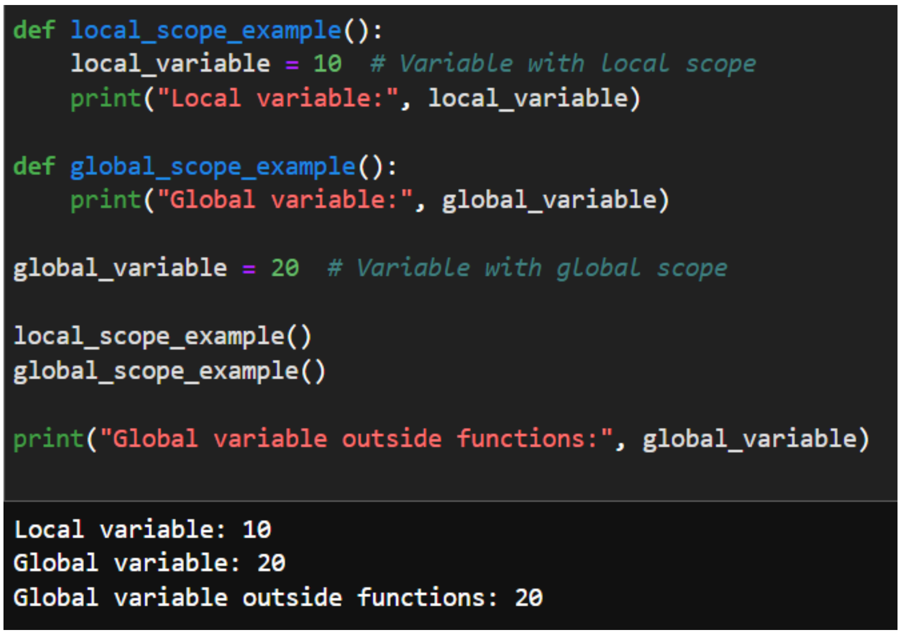
5. **How does the concept of "scope" apply to functions in Python? Explain the difference between local scope and global scope.**

Ans:

In Python, the concept of “scope” refers to the region in a program where a variable or a name is accessible. It determines the visibility and lifetime of variables, i.e., where they can be referenced and modified.

When a variable a defined within a function, it is said to have “local scope”. This means that the variable is only accessible within that function and cannot be accessed from outside the function or in other functions unless it is explicitly passed as an argument or returned as a result.

On the other hand, variables defined outside of any function have “global scope”. Global variables are accessible from any part of the program, including all functions.

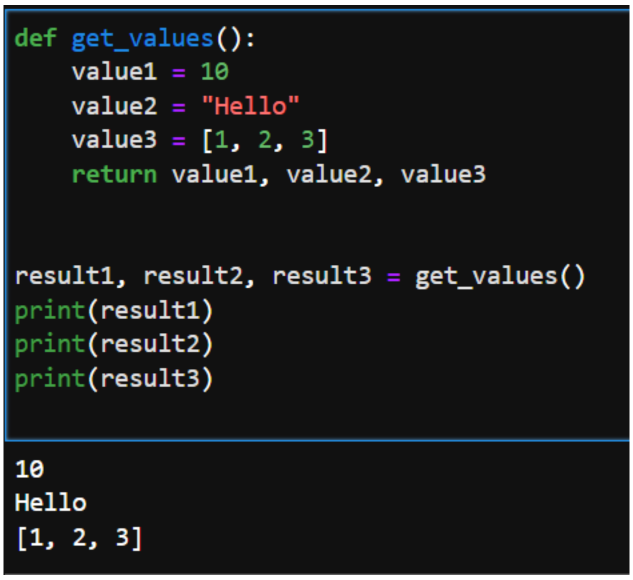


```python
def local_scope_example():
    local_variable = 10  # Variable with local scope
    print("Local variable:", local_variable)

def global_scope_example():
    print("Global variable:", global_variable)

global_variable = 20  # Variable with global scope

local_scope_example()
global_scope_example()

print("Global variable outside functions:", global_variable)
```

```
Local variable: 10
Global variable: 20
Global variable outside functions: 20
```

6. How can you use the "return" statement in a Python function to return multiple value

Ans:



```python
def get_values():
    value1 = 10
    value2 = "Hello"
    value3 = [1, 2, 3]
    return value1, value2, value3


result1, result2, result3 = get_values()
print(result1)
print(result2)
print(result3)
```

```
10
Hello
[1, 2, 3]
```

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

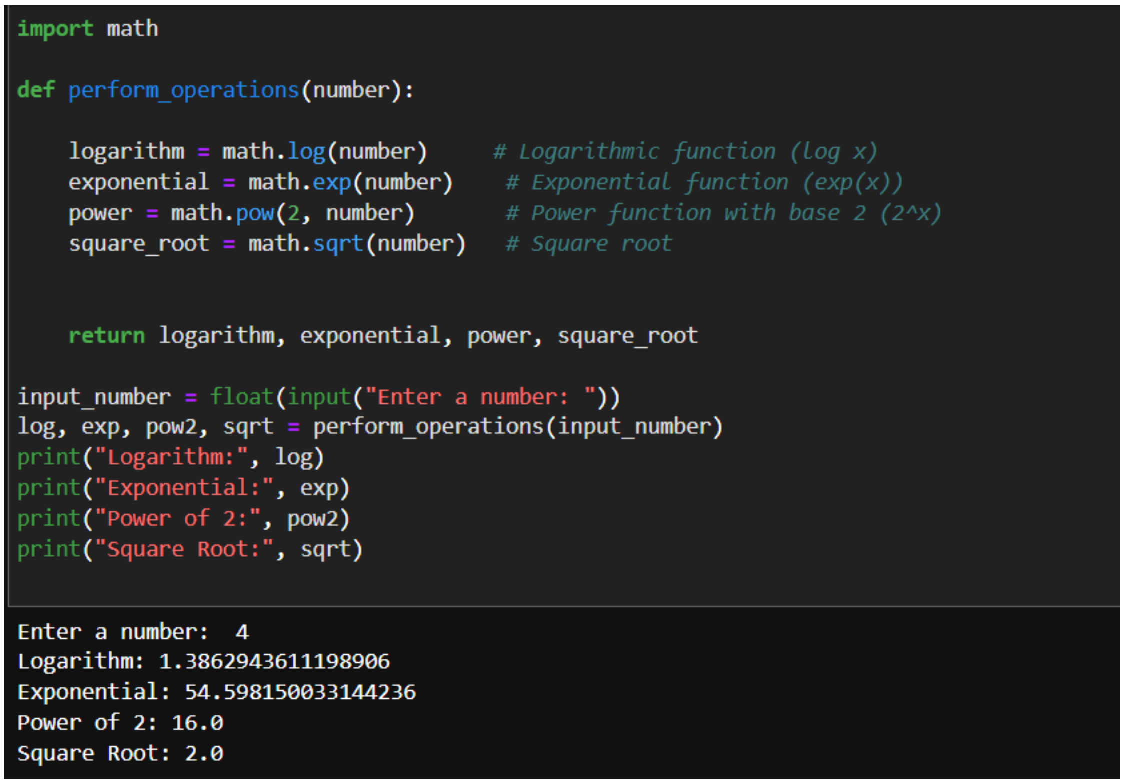
Ans:

Pass by Value: In “pass by value”, a copy of the value of the variable is passed to the function. Any modifications made to the function parameter do not affect the original variable outside the function.

Pass by Reference (Pass by Object Reference in Python): In “pass by reference” or “pass by object reference” in Python, a reference to the variable is passed to the function. Any modifications made to the function parameter can affect the original variables outside the function.

8. **Create a function that can intake integer or decimal value and do following operations:**
   a. **Logarithmic function (log x)**
   b. **Exponential function (exp(x))**
   c. **Power function with base 2 (2^x)**
   d. **Square root**

Ans:

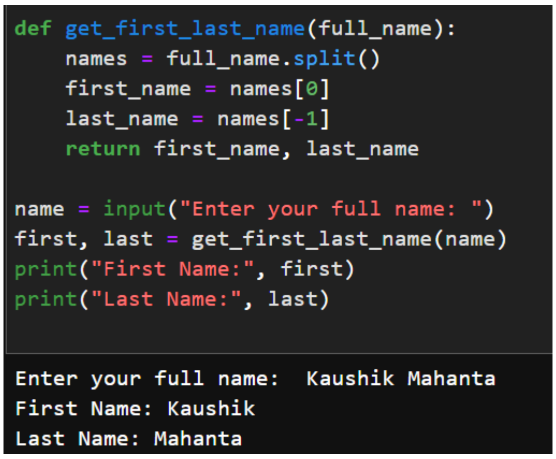


```
import math

def perform_operations(number):

    logarithm = math.log(number)      # Logarithmic function (log x)
    exponential = math.exp(number)     # Exponential function (exp(x))
    power = math.pow(2, number)        # Power function with base 2 (2^x)
    square_root = math.sqrt(number)    # Square root


    return logarithm, exponential, power, square_root

input_number = float(input("Enter a number: "))
log, exp, pow2, sqrt = perform_operations(input_number)
print("Logarithm:", log)
print("Exponential:", exp)
print("Power of 2:", pow2)
print("Square Root:", sqrt)
```

```
Enter a number:  4
Logarithm: 1.3862943611198906
Exponential: 54.598150033144236
Power of 2: 16.0
Square Root: 2.0
```

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

Ans:



```
def get_first_last_name(full_name):
    names = full_name.split()
    first_name = names[0]
    last_name = names[-1]
    return first_name, last_name

name = input("Enter your full name: ")
first, last = get_first_last_name(name)
print("First Name:", first)
print("Last Name:", last)
```

```
Enter your full name:  Kaushik Mahanta
First Name: Kaushik
Last Name: Mahanta
```