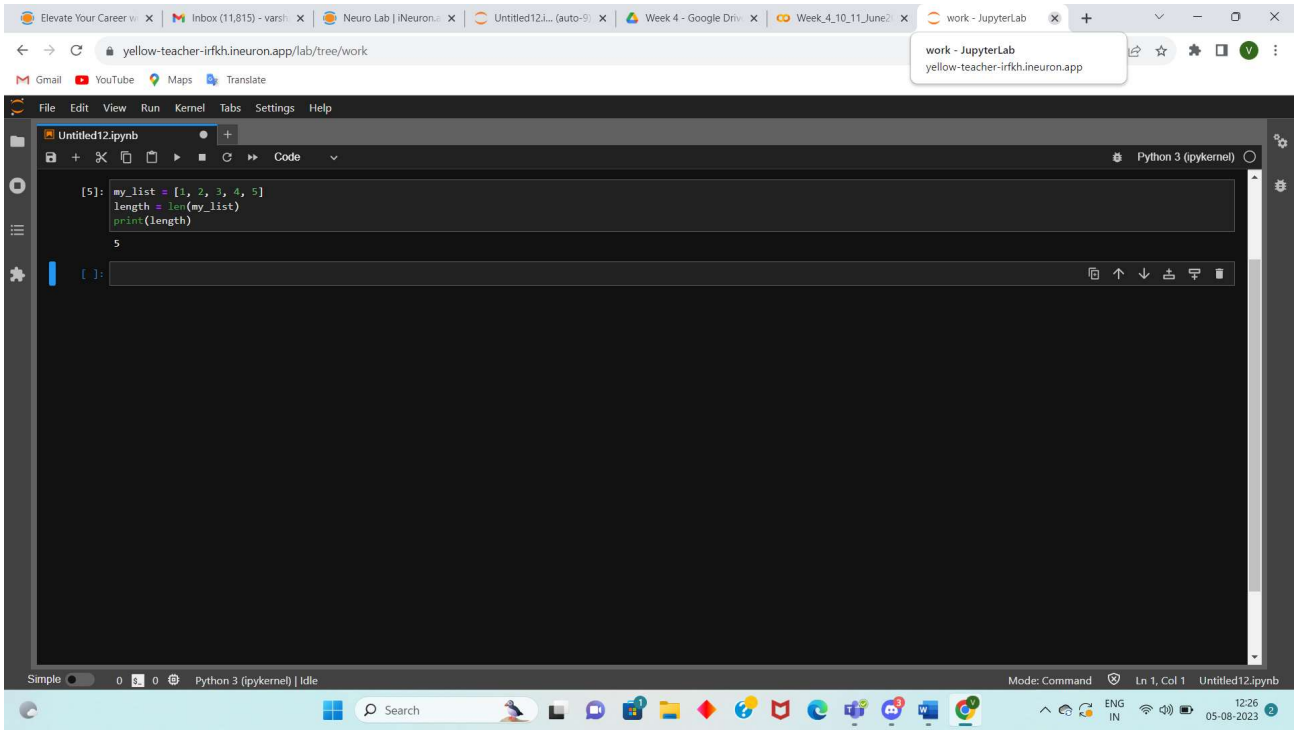
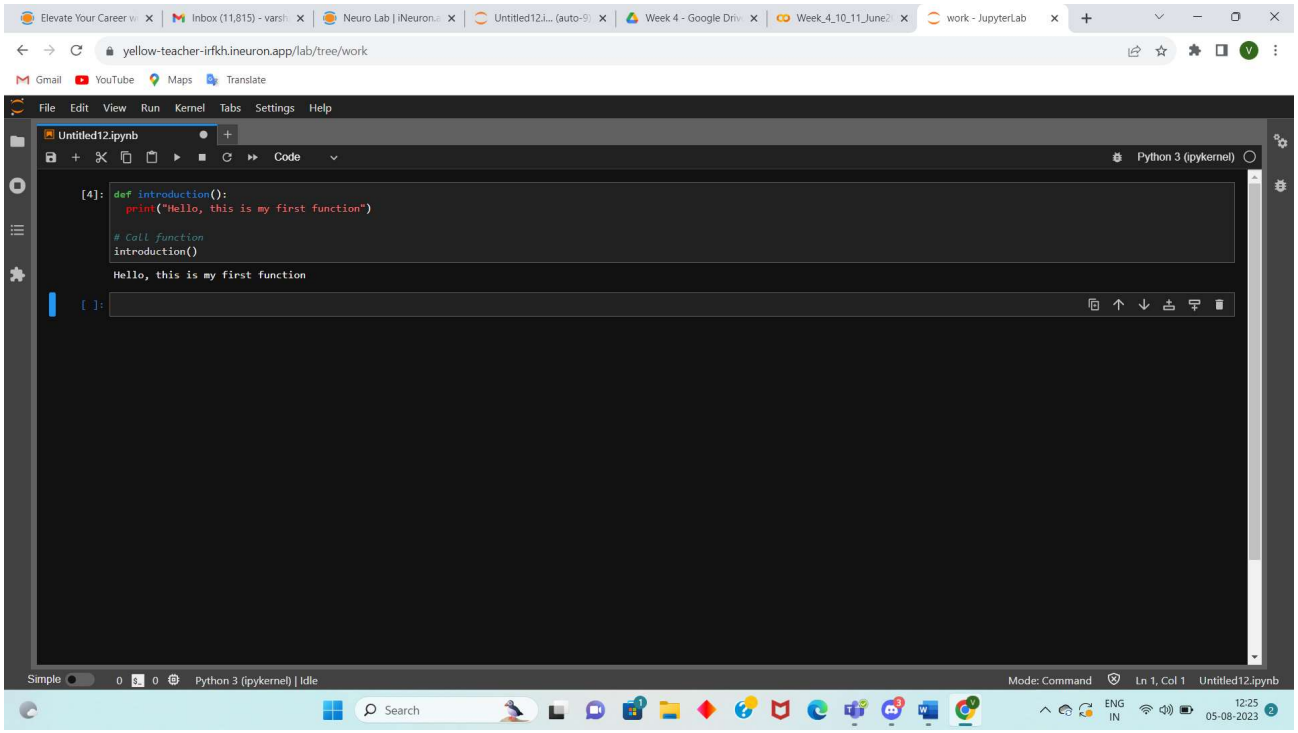
1. In Python, what is the difference between a built-in function and a user-defined function? Provide an example of each.

   Ans:- **Built in Function**:- Built in function are functions that are provided by Python itself. They are readily available for use without requiring any additional code to be written.



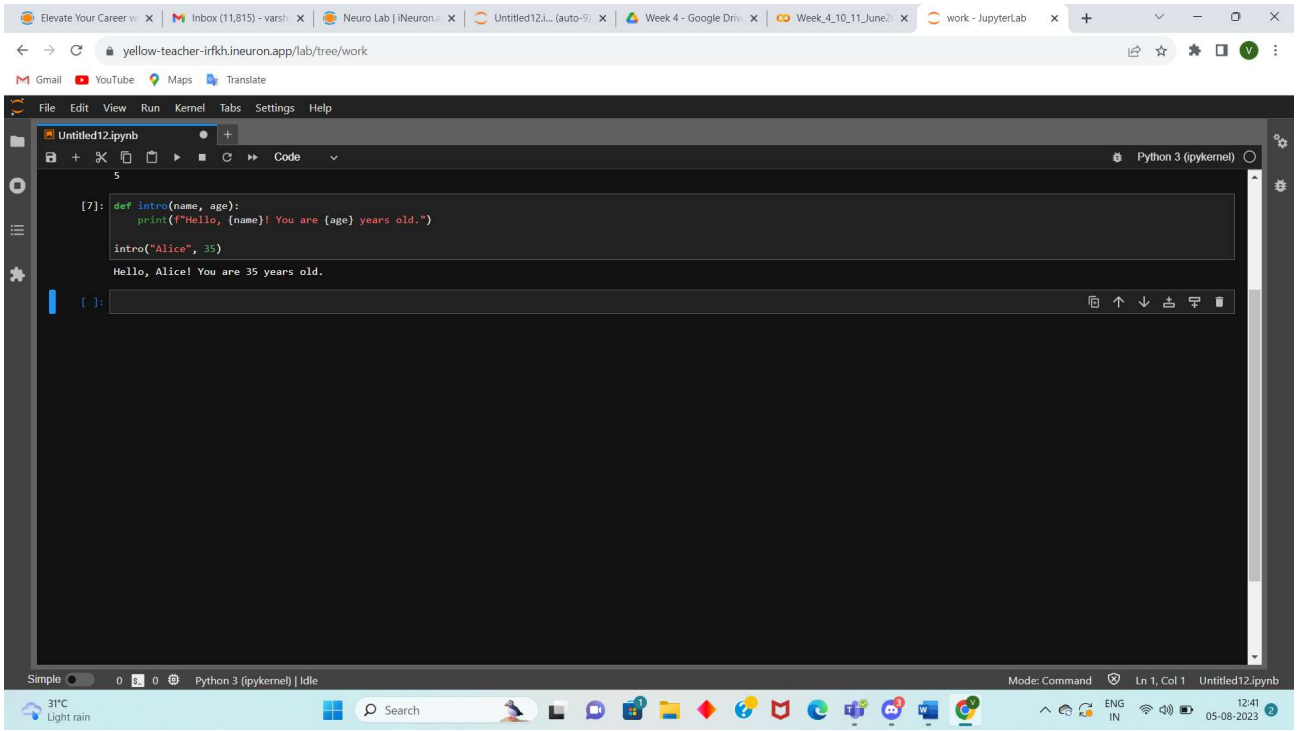
User defined function:- A user-defined function in Python is a code block that performs a specific task and is defined by the programmer. User-defined functions are created with the `def` keyword followed by a function name of our choosing.



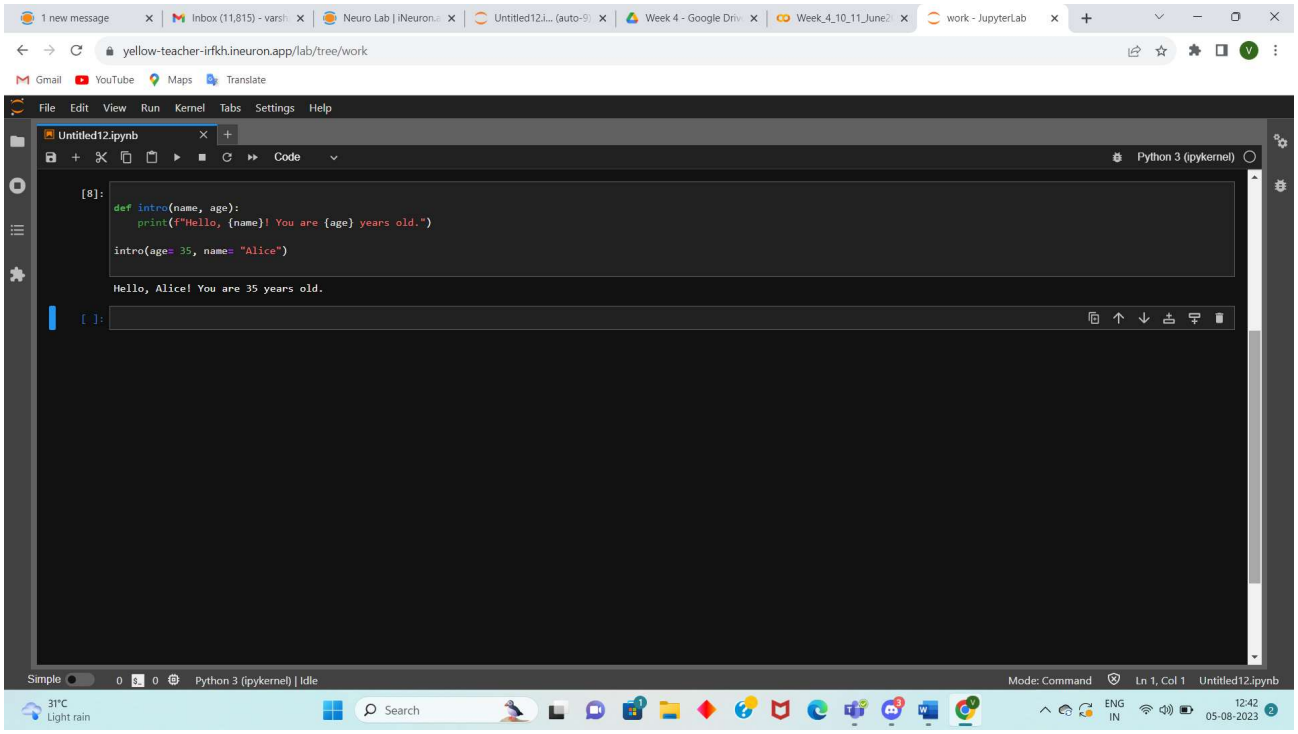
2. How can you pass arguments to a function in Python? Explain the difference between positional arguments and keyword arguments.

   Ans:- In Python , We can pass arguments to a function to provide it with necessary data it needs to perform its task.

   Positional Arguments:-It is the most basic way to pass arguments to a function. They are specified in order in which the parameters are defined in the function's parameter list.The value passed as positional arguments are assigned to the corresponding parameters based on their position.



Keyword Arguments:- Keyword arguments allow us to specify which parameter each value is assigned to. Instead of relying on the order , you use the parameter names followed by the assignment operator(=) to indicate which value corresponds to which parameter.



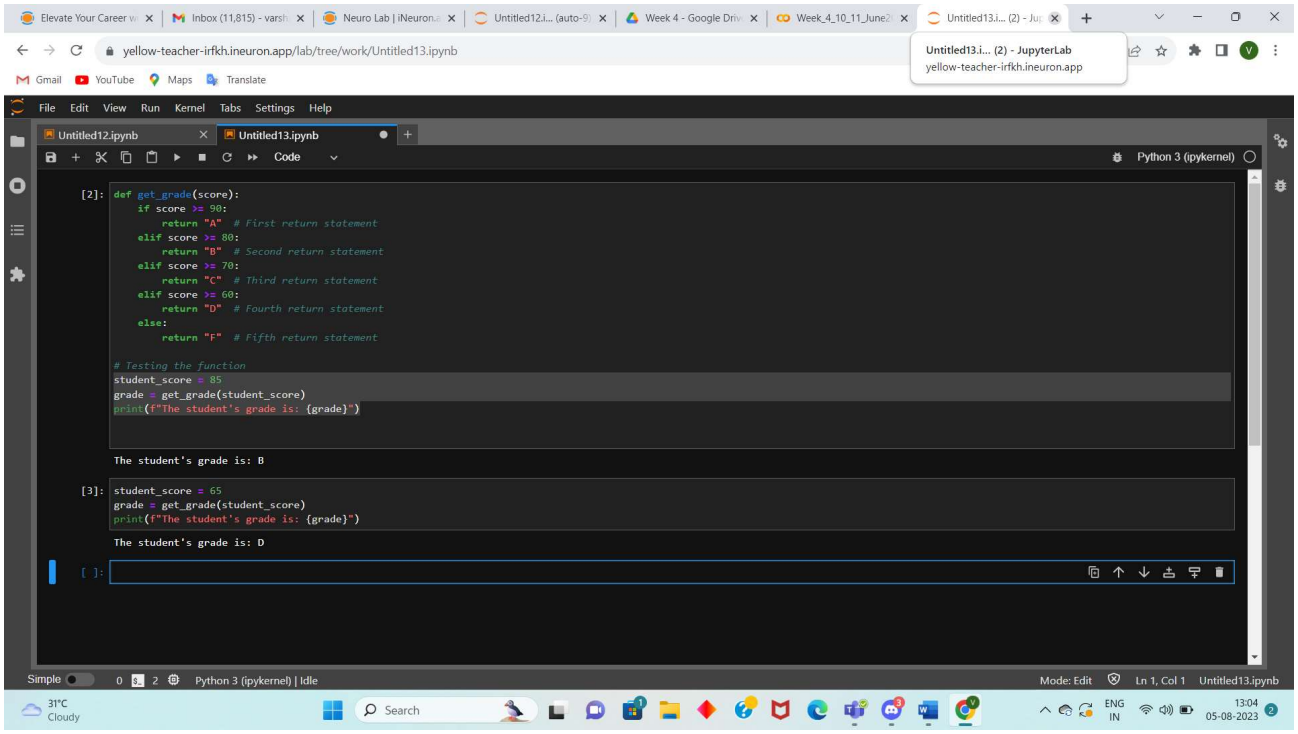
3. What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.

   Ans:-

   The return statement is crucial in user-defined functions as it enables the functions to produce outputs, pass data, allow result reusability, handle conditional returns, and control the flow of the function's execution.

   The return statement in a function serves the purpose of sending a value back to the caller of the function .when a function is called & it encounters a return statement , the function's execution is halted & the value specified in the return statement is sent back to the caller.

   Yes , a function can have multiple return statements



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

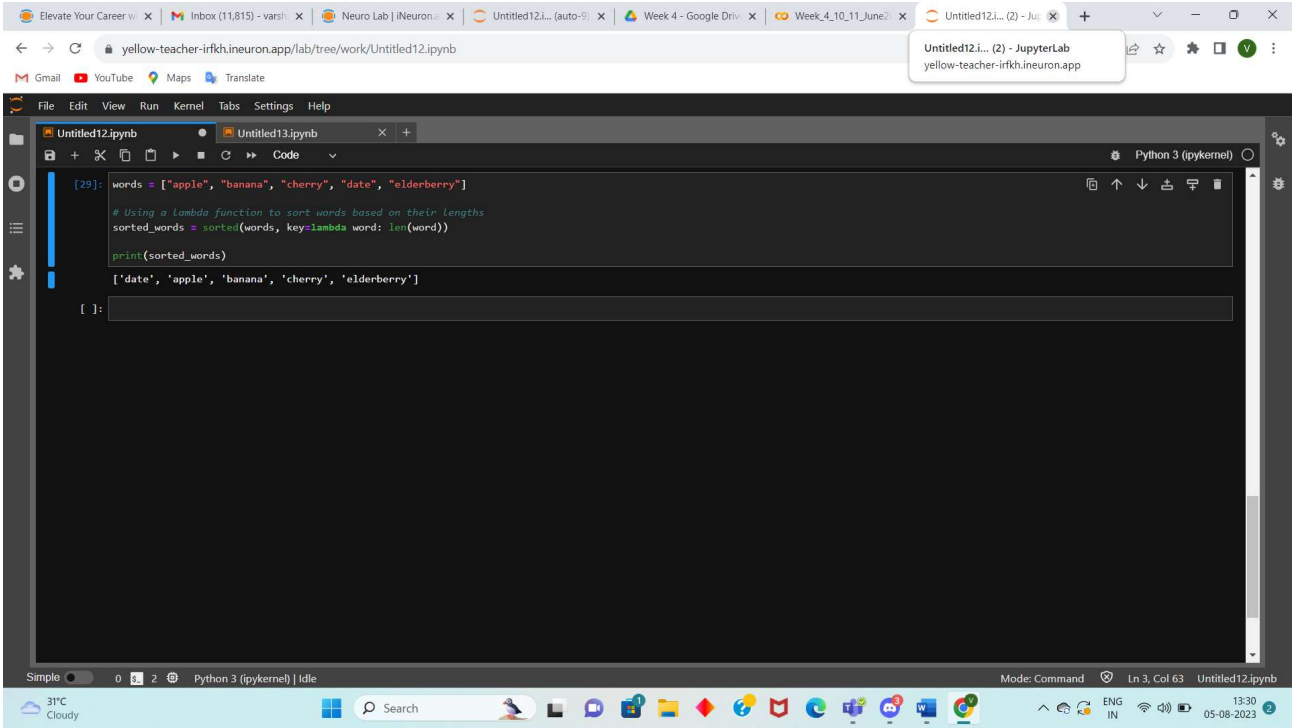
Ans:-

Here are some key points about lambda functions in Python, each point is a one-liner:

1. Lambda functions are small, anonymous functions defined with the lambda keyword.
2. They can take any number of arguments but can only have one expression.
3. The expression is evaluated and returned when the function is called.
4. Lambda functions do not require a return statement, the expression is implicitly returned.
5. You can't include statements like loops, if or else in lambda functions; only expressions are allowed.
6. Lambda functions are useful for small tasks that are not reused throughout your code.
7. They can be assigned to variables and used like regular functions.

Difference :- Lambda functions are typically more concise & are defined in a single line . Regular functions have more structured syntax with a formal header & the use of return keywords.

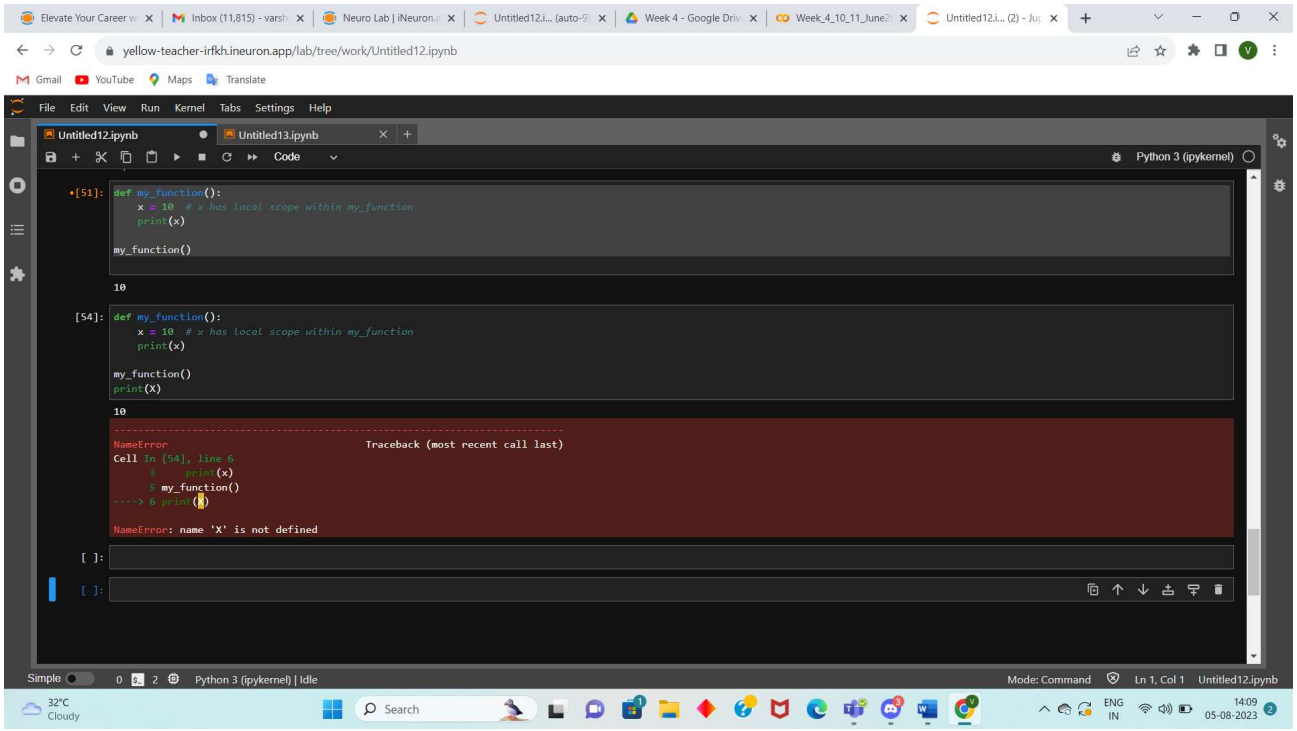
Lambda functions can only contain a single expression, while regular functions can have multiple statement & more complex logic.



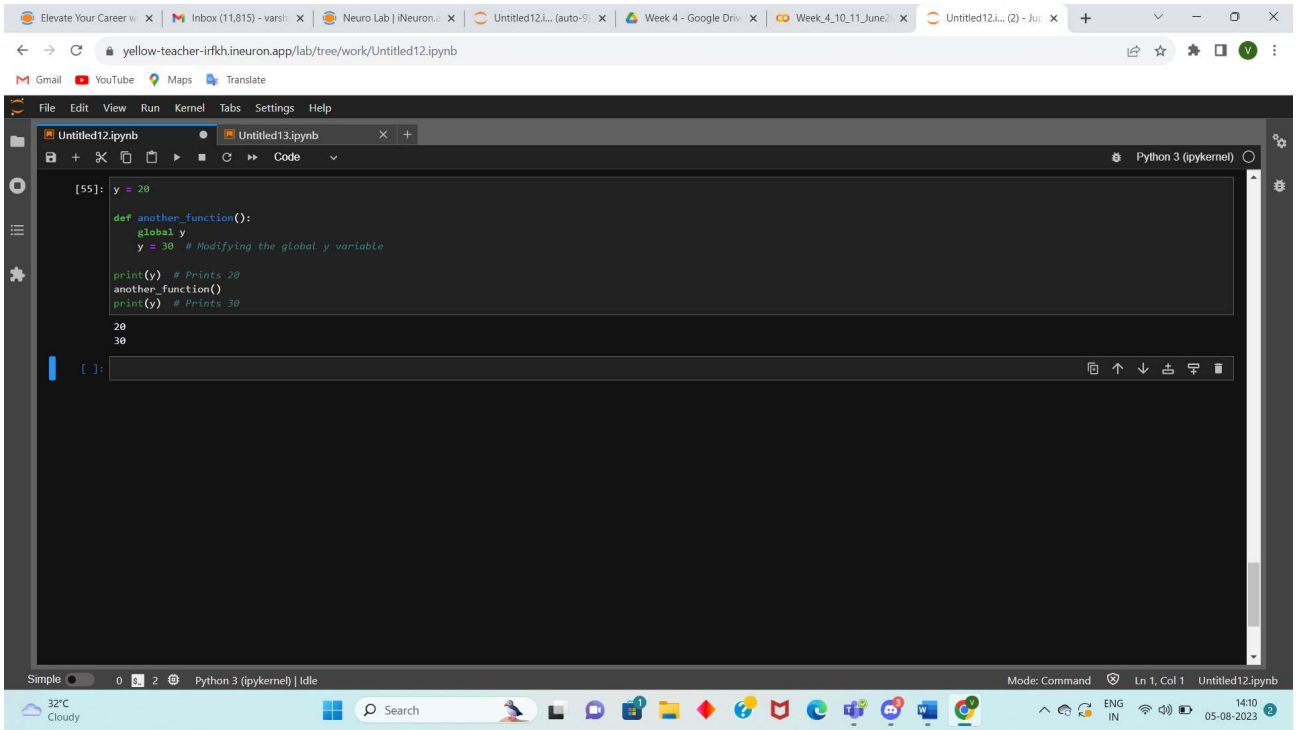
5. How does the concept of "scope" apply to functions in Python? Explain the difference between local scope and global scope.

Ans:- In Python the concept of "scope" refers to the region in which a variable or name is accessible . It defines the context in which variables can be accessed, modified or referenced.

Local Scope:- When we define a variable inside a function, that variable has local scope. It means the variable is only accessible within that specific function. Once the function finishes executing, the local variables within it are destroyed and cannot be accessed from outside the function.

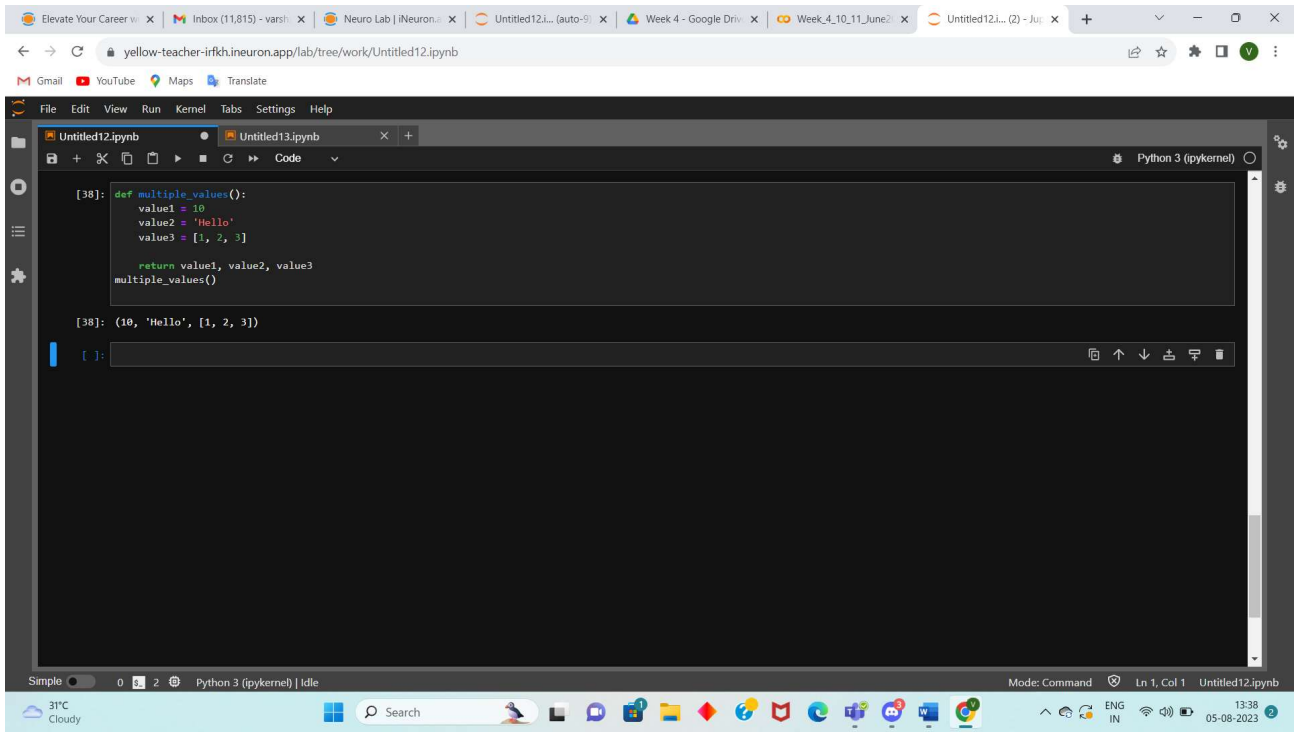


Global Scope: Variables defined outside of any function have global scope. They can be accessed from anywhere in the code, both within functions and outside functions. Global variables are useful for storing values that need to be shared among multiple functions or for maintaining state throughout the program. However, modifying global variables from within a function requires using the global keyword.



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

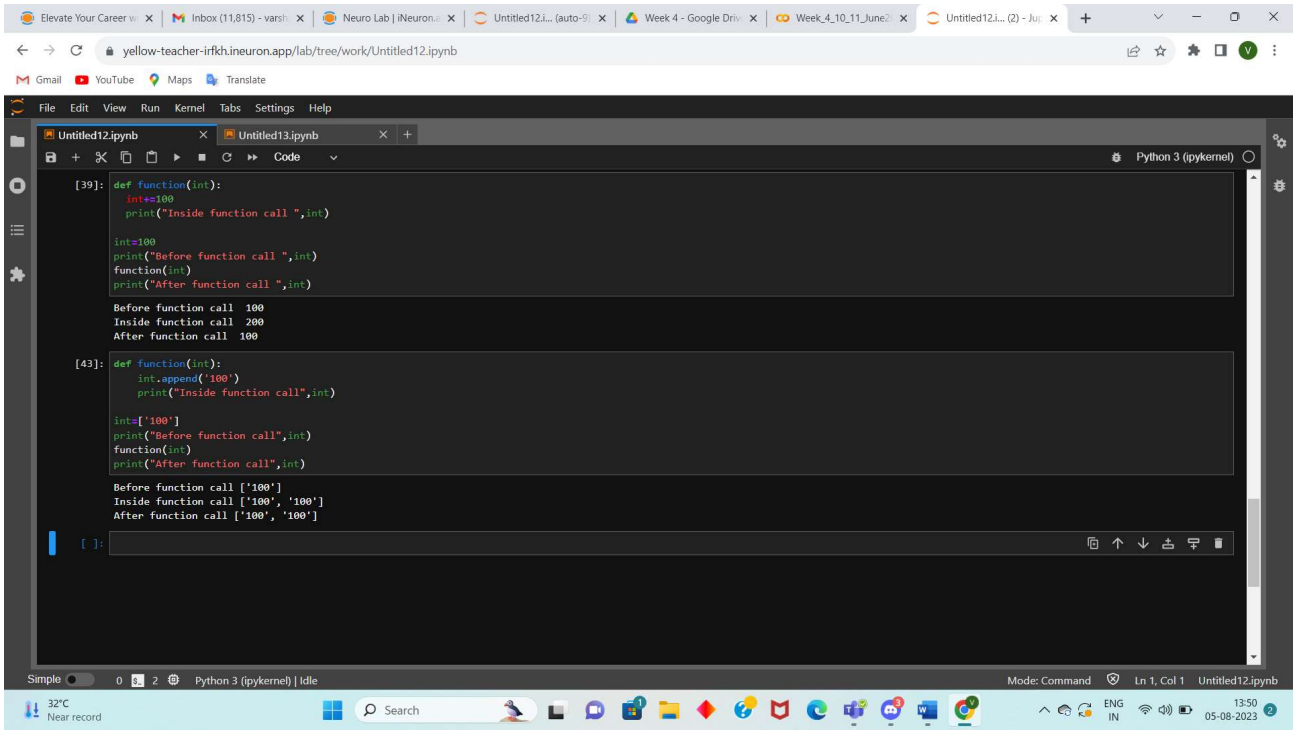
Ans:



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

Ans:-

When we pass function arguments by reference, those arguments are only references to existing values. In contrast, when we pass arguments by value, those arguments become independent copies of the original values.



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

Untitled12.ipynb

[47]:
```python
import math

def perform_operations(value):
    try:
        value = float(value)
        result = {
            "Logarithmic": math.log(value),
            "Exponential": math.exp(value),
            "Power (base 2)": 2 ** value,
            "Square Root": math.sqrt(value)
        }
        return result
    except ValueError:
        return "Invalid input. Please provide a valid numeric value."


input_value = input("Enter a numeric value: ")
operations_result = perform_operations(input_value)
if isinstance(operations_result, str):
    print(operations_result)
else:
    for operation, result in operations_result.items():
        print(f"{operation} result: {result}")
```

```
Enter a numeric value:  20
Logarithmic result: 2.995732273553991
Exponential result: 485165195.4097903
Power (base 2) result: 1048576.0
Square Root result: 4.47213595499958
```

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

Ans:-