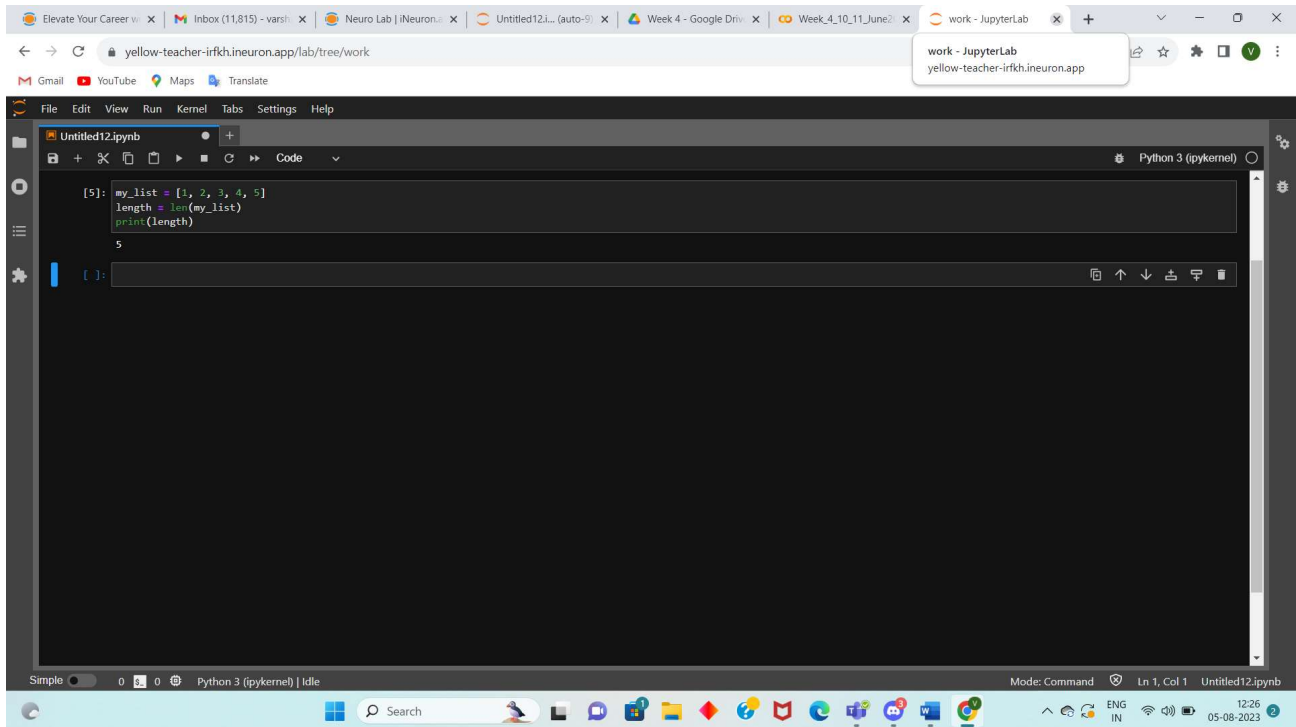


1. In Python, what is the difference between a built-in function and a user-defined function? Provide an example of each.

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.

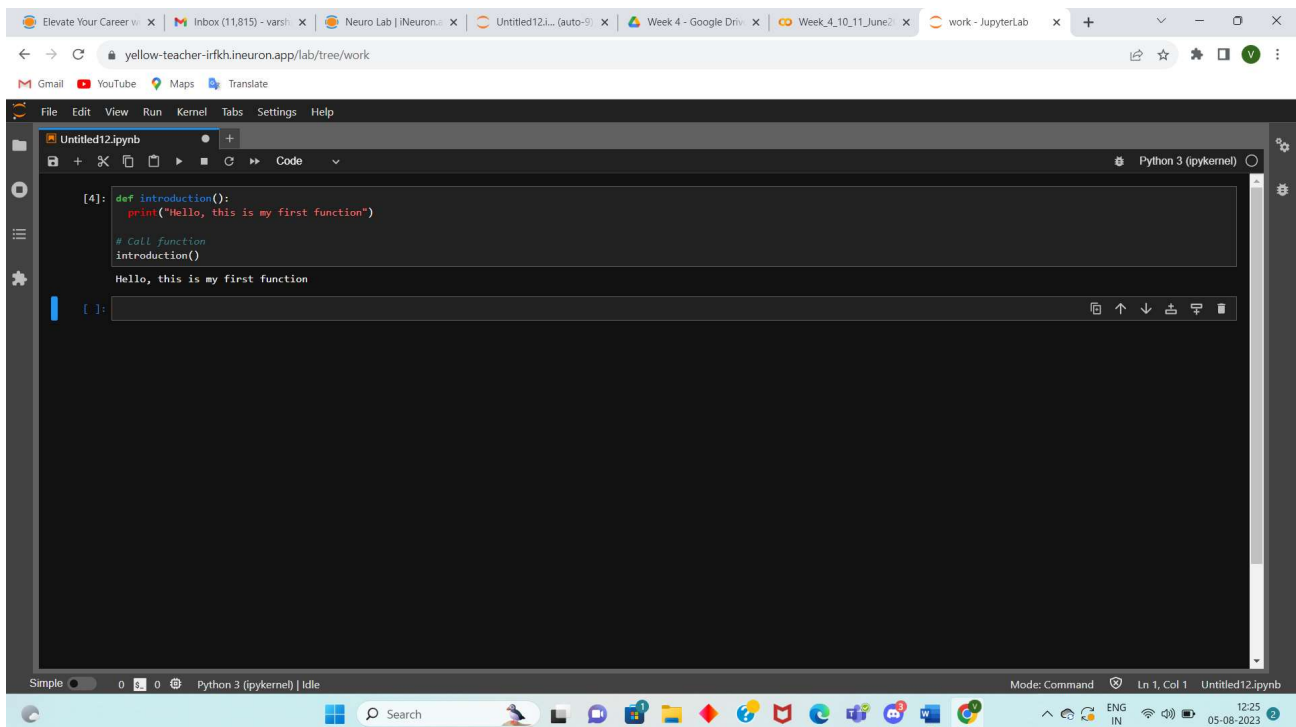


The screenshot shows a JupyterLab window with a single notebook titled 'Untitled12.ipynb'. The code cell contains the following Python code:

```
[5]: my_list = [1, 2, 3, 4, 5]
      length = len(my_list)
      print(length)
      5
```

The output of the code is the number 5, which is displayed below the code cell. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with various icons for file operations and execution. The status bar at the bottom indicates the current mode is 'Simple' and the kernel is 'Python 3 (ipykernel) | Idle'.

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.



The screenshot shows a JupyterLab window with a single notebook titled 'Untitled12.ipynb'. The code cell contains the following Python code:

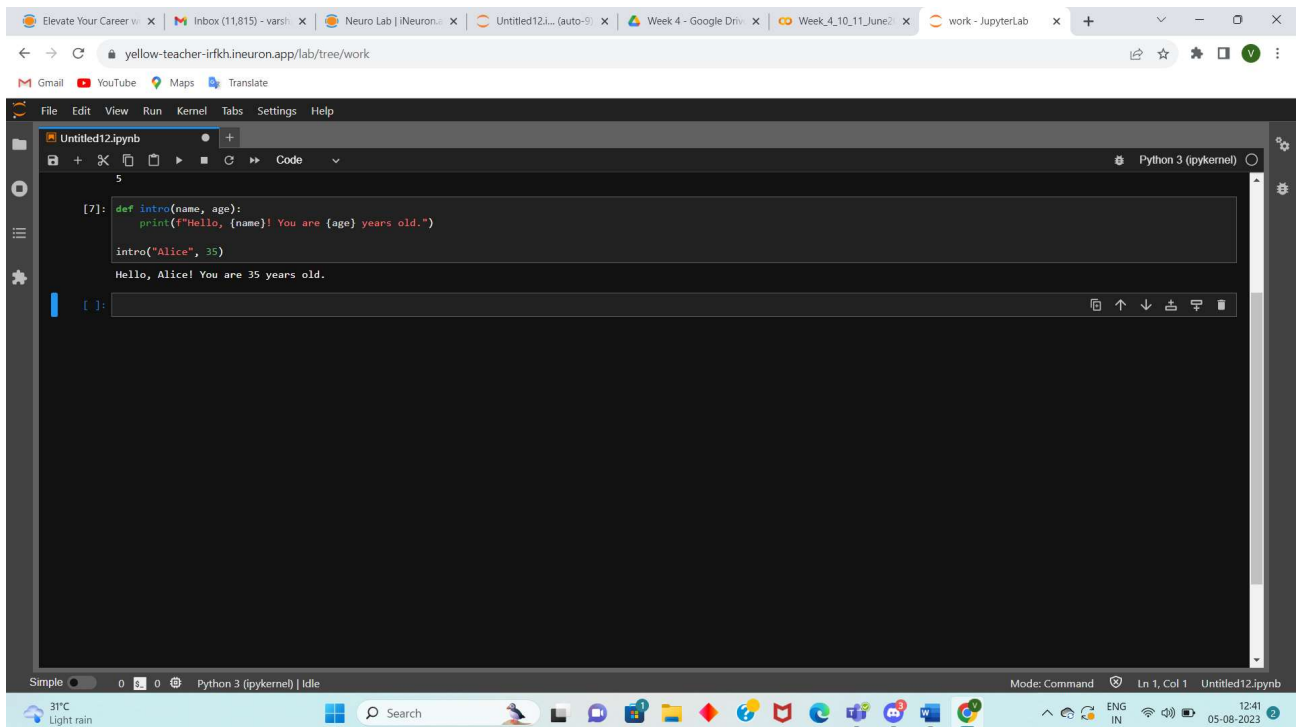
```
[4]: def introduction():
      print("Hello, this is my first function")
      # Call function
      introduction()
      Hello, this is my first function
```

The output of the code is the string 'Hello, this is my first function', which is displayed below the code cell. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with various icons for file operations and execution. The status bar at the bottom indicates the current mode is 'Simple' and the kernel is 'Python 3 (ipykernel) | Idle'.

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.

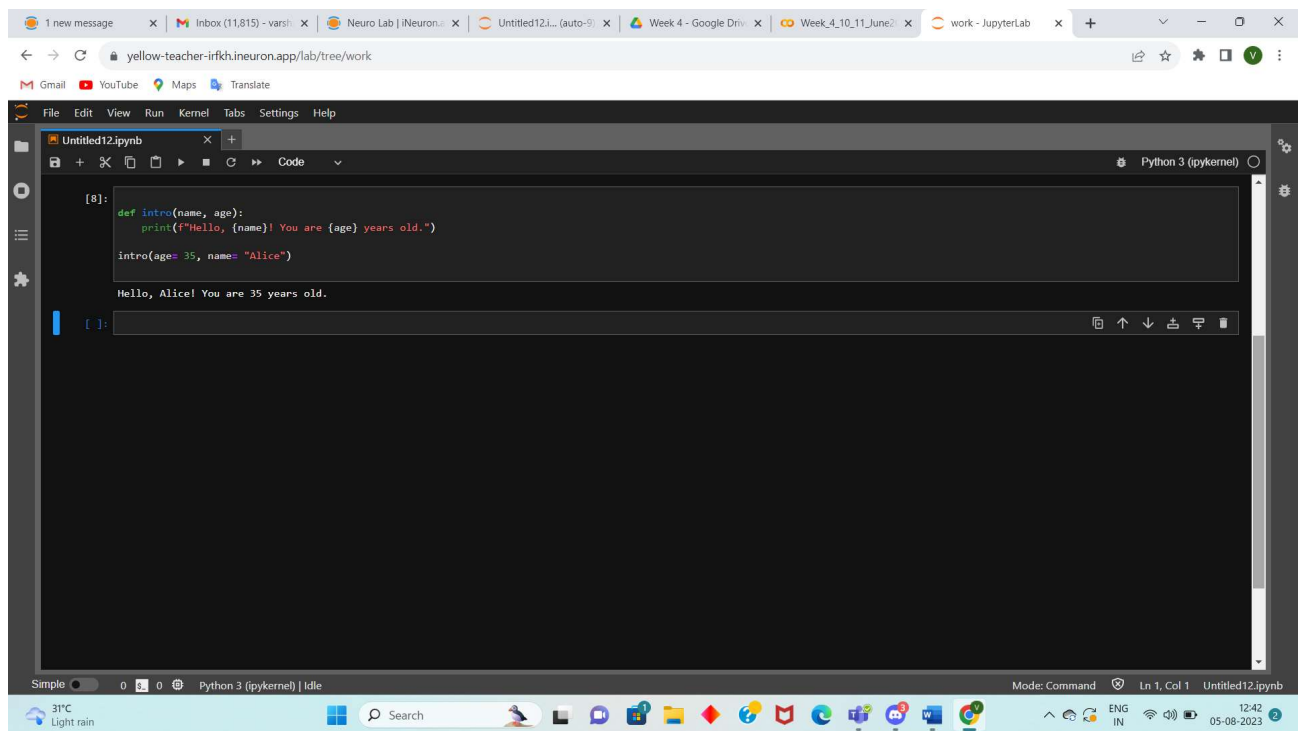


The screenshot shows a web browser window with multiple tabs. The active tab is a JupyterLab interface at the URL `yellow-techer-irfkh.neuron.app/lab/tree/work`. The JupyterLab window has a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar. The main area displays a Python notebook with the following code in a cell:

```
[7]: def intro(name, age):  
    print(f"Hello, {name}! You are {age} years old.")  
  
    intro("Alice", 35)  
  
Hello, Alice! You are 35 years old.
```

The output of the function call is displayed below the code. The bottom status bar of the JupyterLab window shows "Simple", "0", "Python 3 (ipykernel) | Idle", "Mode: Command", "Ln 1, Col 1", and "Untitled12.ipynb". The system tray at the bottom of the screen shows the date and time as "05-08-2023 12:41".

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.



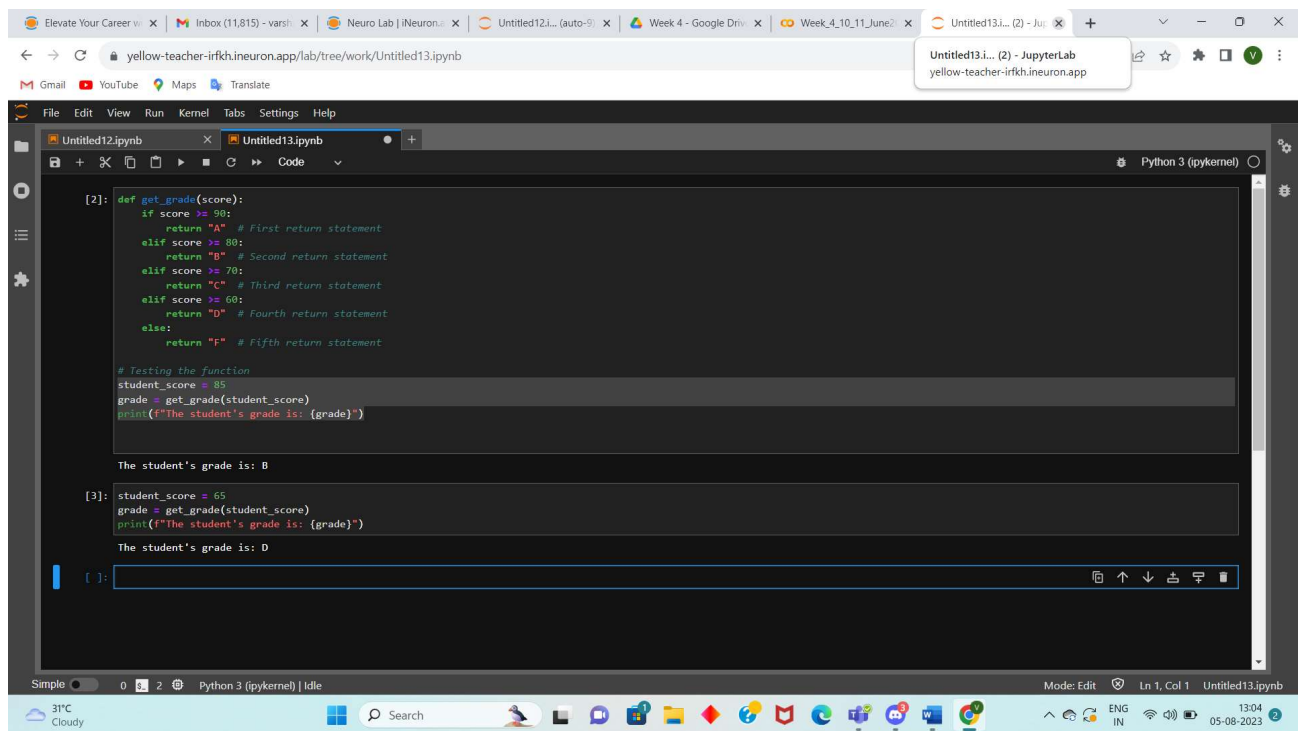
3. What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.

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



```
[2]: def get_grade(score):
    if score >= 90:
        return "A" # First return statement
    elif score >= 80:
        return "B" # Second return statement
    elif score >= 70:
        return "C" # Third return statement
    elif score >= 60:
        return "D" # Fourth return statement
    else:
        return "F" # Fifth return statement

# Testing the function
student_score = 85
grade = get_grade(student_score)
print(f"The student's grade is: {grade}")

The student's grade is: B

[3]: student_score = 65
grade = get_grade(student_score)
print(f"The student's grade is: {grade}")

The student's grade is: D

[ ]:
```

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.

The screenshot shows a JupyterLab window with a Python 3 (ipykernel) environment. The notebook contains the following code:

```
[29]: words = ["apple", "banana", "cherry", "date", "elderberry"]
      # Using a Lambda function to sort words based on their lengths
      sorted_words = sorted(words, key=lambda word: len(word))
      print(sorted_words)

['date', 'apple', 'banana', 'cherry', 'elderberry']
```

The output of the code is displayed below the cell: `['date', 'apple', 'banana', 'cherry', 'elderberry']`. The interface includes a file explorer on the left, a top bar with various icons, and a bottom status bar showing the current mode and file name.

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.

The screenshot shows a JupyterLab window with a Python 3 (ipykernel) environment. The notebook contains the following code:

```
[53]: def my_function():
      x = 10 # x has local scope within my_function
      print(x)

      my_function()

10

[54]: def my_function():
      x = 10 # x has local scope within my_function
      print(x)

      my_function()
      print(x)

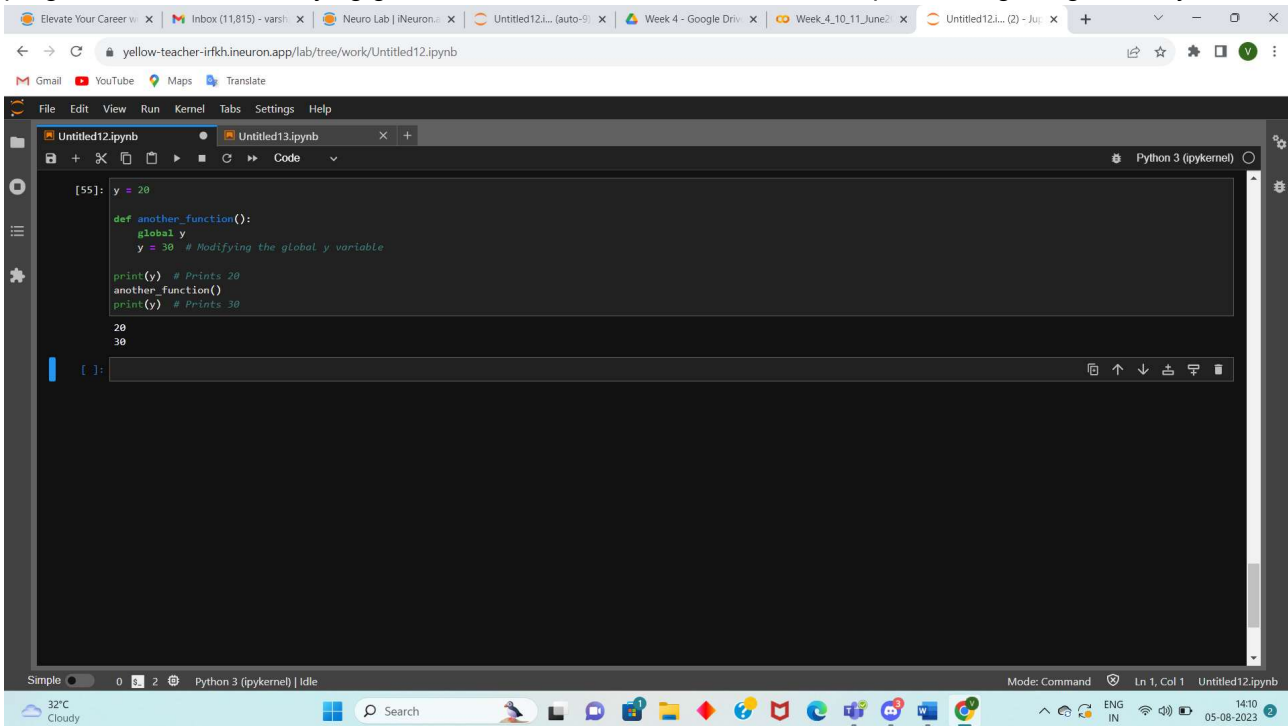
      10
```

The output of the code is displayed below the cell: `10`. The interface includes a file explorer on the left, a top bar with various icons, and a bottom status bar showing the current mode and file name.

A red error message is displayed below the code, indicating a `NameError` (name 'x' is not defined) at line 6 of cell [54]. The error message is:

```
NameError                                Traceback (most recent call last)
Cell In [54], line 6
      3 print(x)
      5 my_function()
----> 6 print(x)
NameError: name 'x' is not defined
```

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.



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser's address bar displays the URL `yellow-teacher-irfkh.inuron.app/lab/tree/work/Untitled12.ipynb`. The notebook has two tabs: 'Untitled12.ipynb' (active) and 'Untitled13.ipynb'. The active cell contains the following Python code:

```
[55]: y = 20

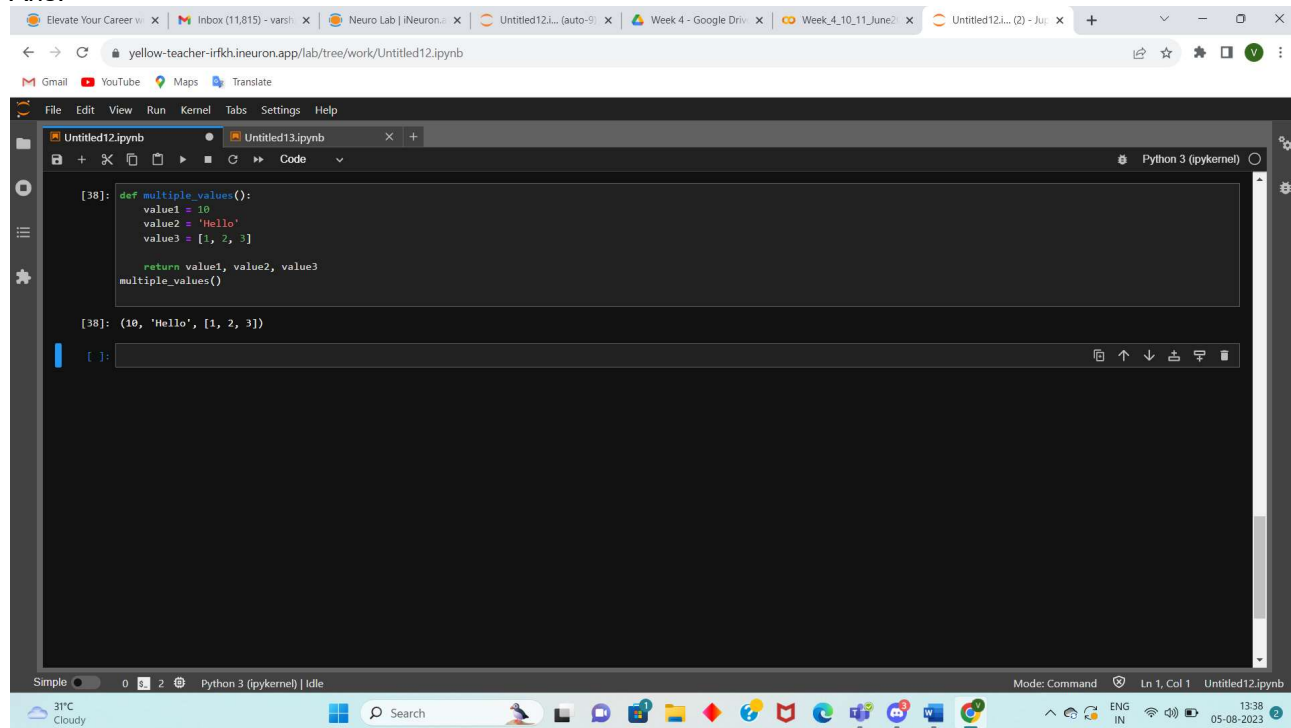
def another_function():
    global y
    y = 30 # Modifying the global y variable

print(y) # Prints 20
another_function()
print(y) # Prints 30
```

Below the code cell, the output shows the values `20` and `30` on separate lines. The bottom status bar of the notebook indicates 'Python 3 (ipykernel) | idle', 'Mode: Command', and 'Ln 1, Col 1'. The Windows taskbar at the bottom shows the system clock as 14:10 on 05-08-2023, along with weather information (32°C Cloudy) and various application icons.

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

Ans:



The screenshot shows a Jupyter Notebook interface with two tabs: 'Untitled12.ipynb' and 'Untitled13.ipynb'. The active tab is 'Untitled12.ipynb', which contains a Python code cell. The code defines a function named 'multiple_values()' that returns three values: 'value1' (10), 'value2' ('Hello'), and 'value3' ([1, 2, 3]). Below the code cell, the output of the function is displayed as a tuple: (10, 'Hello', [1, 2, 3]). The interface includes a top navigation bar with various icons and a bottom status bar showing the current mode (Simple), the kernel (Python 3 (ipykernel)), and the file name (Untitled12.ipynb).

```
[38]: def multiple_values():  
      value1 = 10  
      value2 = 'Hello'  
      value3 = [1, 2, 3]  
      return value1, value2, value3  
      multiple_values()  
  
[38]: (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:-

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.

```
[39]: def function(int):  
      int+=100  
      print("Inside function call ",int)  
  
      int=100  
      print("Before function call ",int)  
      function(int)  
      print("After function call ",int)  
  
Before function call 100  
Inside function call 200  
After function call 100  
  
[43]: def function(int):  
      int.append('100')  
      print("Inside function call",int)  
  
      int=['100']  
      print("Before function call",int)  
      function(int)  
      print("After function call",int)  
  
Before function call ['100']  
Inside function call ['100', '100']  
After function call ['100', '100']  
  
[ ]:
```

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

Elevate Your Career vi | Inbox (11,815) - vari... | Neuro Lab | iNeuron... | Untitled12.i... (auto-S) | Week 4 - Google Driv... | Week_4_10_11_June2 | Untitled12.i... (2) - JupyterLab | yellow-teacher-irfkh.ineuron.app

yellow-teacher-irfkh.ineuron.app/lab/tree/work/Untitled12.ipynb

Untitled12.i... (2) - JupyterLab
yellow-teacher-irfkh.ineuron.app

File Edit View Run Kernel Tabs Settings Help

Untitled12.ipynb x Untitled13.ipynb x +

Python 3 (ipykernel)

```
[47]: 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

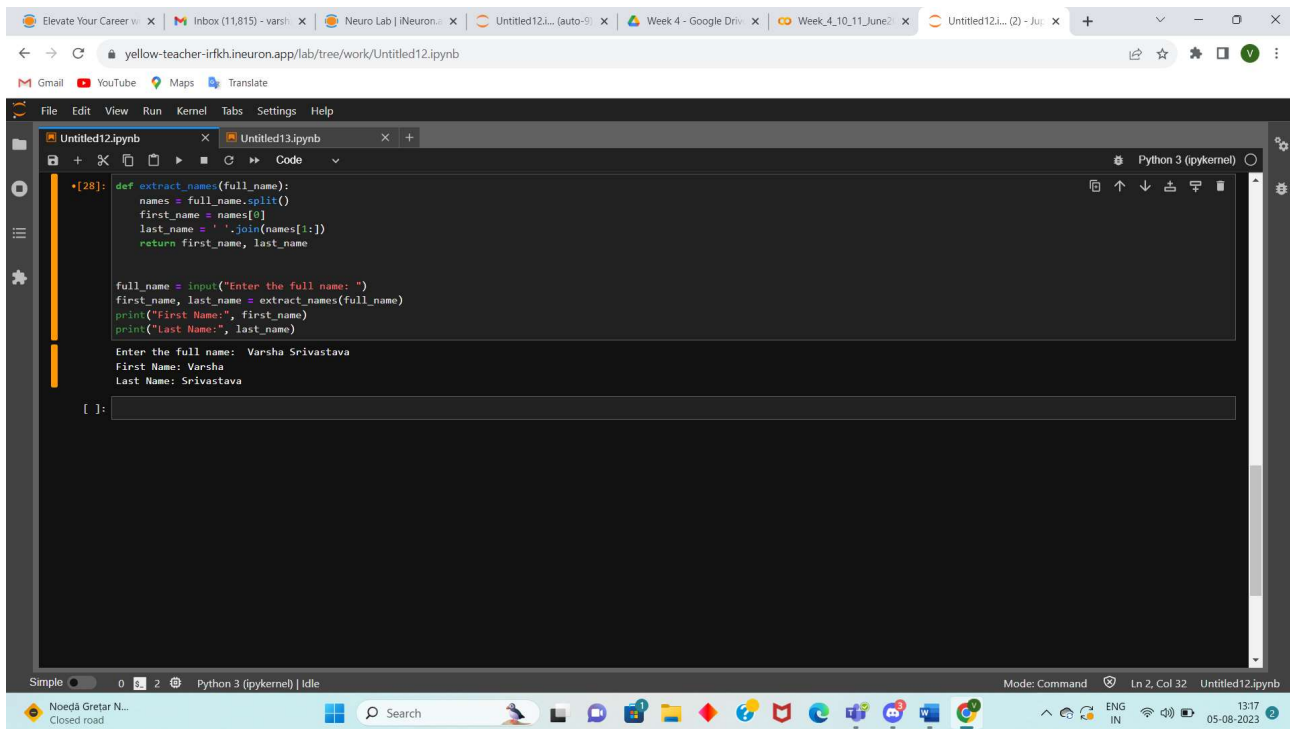
[]:

Simple 0 2 Python 3 (ipykernel) | idle Mode: Command Ln 24, Col 1 Untitled12.ipynb

32°C Cloudy Search ENG IN 13:57 05-08-2023

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

Ans:-



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser's address bar displays the URL `yellow-teacher-irfkh.ineuron.app/lab/tree/work/Untitled12.ipynb`. The notebook has two tabs: 'Untitled12.ipynb' (active) and 'Untitled13.ipynb'. The code in the active cell is as follows:

```
[28]: def extract_names(full_name):
      names = full_name.split()
      first_name = names[0]
      last_name = ' '.join(names[1:])
      return first_name, last_name

      full_name = input("Enter the full name: ")
      first_name, last_name = extract_names(full_name)
      print("First Name:", first_name)
      print("Last Name:", last_name)

      Enter the full name: Varsha Srivastava
      First Name: Varsha
      Last Name: Srivastava

      [ ]:
```

The output of the code execution is visible below the code cell, showing the input prompt and the resulting first and last names. The bottom status bar of the notebook indicates 'Python 3 (ipykernel) | Idle', 'Mode: Command', and 'Ln 2, Col 32'. The system tray at the bottom shows the date and time as '05-08-2023 13:17'.