PROGRAM : B.TECH/CSE

SPECIALIZATION : AIML

COURSE TITLE : AI ASSISTED CODING

COURSE CODE : 24CS101PC214

SEMESTER : 3RD

NAME OF THE STUDENT : KARNAKANTI GODADEVI

ENROLLMENT NO : 2403A52003

BATCH NO : 01

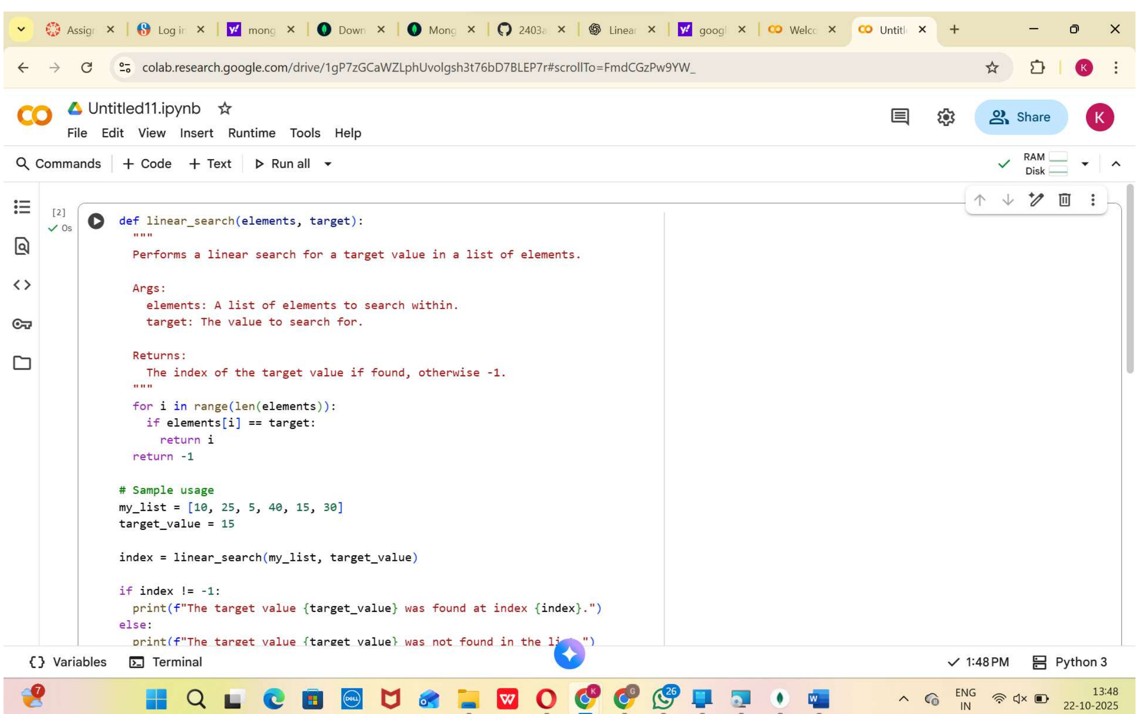
Task Description #1 – Linear Search implementation

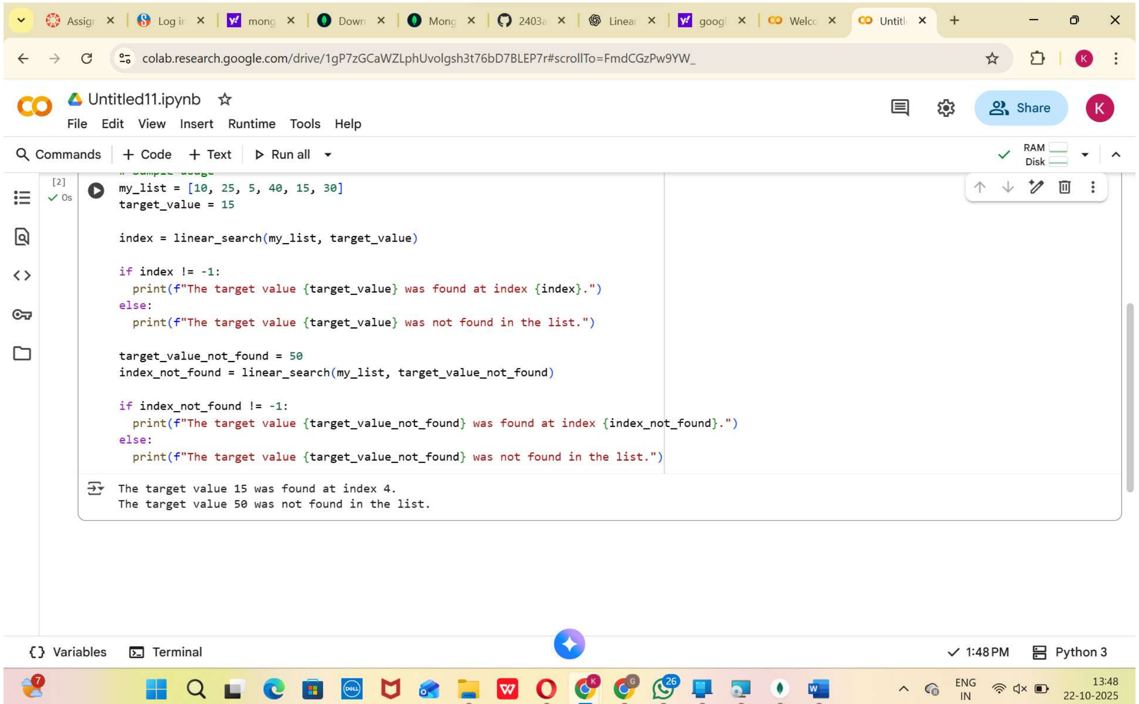
Task: Write python code for linear_search() function to search a value in a list and extract it's index.

Prompt:

Write a Python function named linear_search() that takes two arguments: a list of elements and a target value to search for. The function should iterate through the list using a linear search technique and return the index of the target value if it is found. If the value is not found in the list, the function should return -1. Also, include sample input and output demonstrating how the function works.

Code&Output:





Observation:

This code implements a linear search. It checks each element sequentially. While simple, it can be inefficient for large lists as it has a time complexity of O(n). It's best suited for small datasets or unsorted lists where other methods aren't applicable.

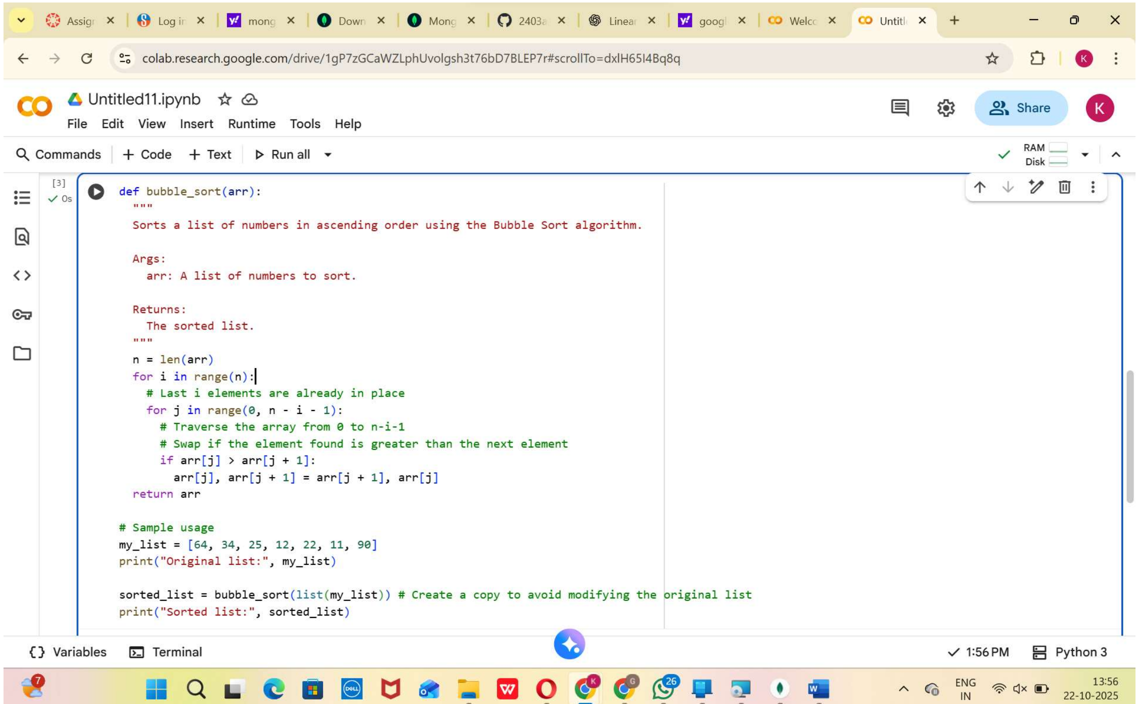
Task Description #2 – Sorting Algorithms

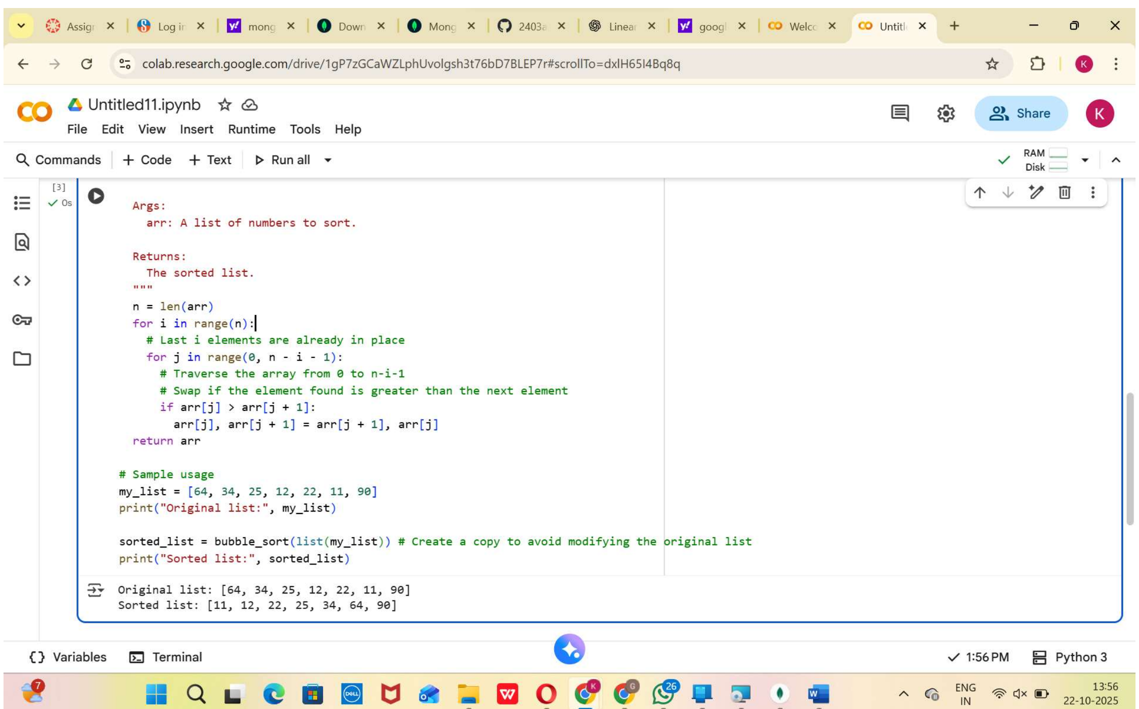
Task: Ask AI to implement Bubble Sort and check sorted output

Prompt:

Write a Python program with a function bubble_sort() that sorts a list of numbers in ascending order using the Bubble Sort algorithm. Print the original and sorted lists, and include sample input and output.

Code &Output:





Observation:

This code implements the Bubble Sort algorithm. It repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. While easy to understand, it's inefficient for large datasets, having a time complexity of O(n^2). It's mainly used for educational purposes or sorting small arrays.

Task Description #3 – Optimization

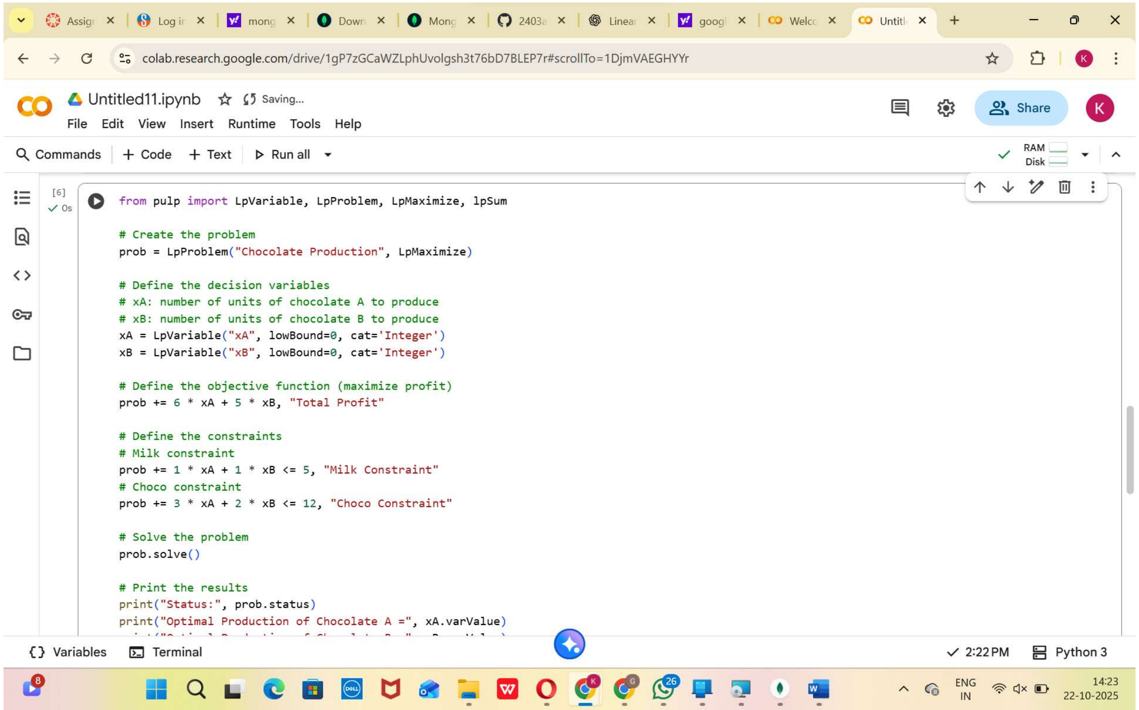
Task: Write python code to solve below case study using linear optimization

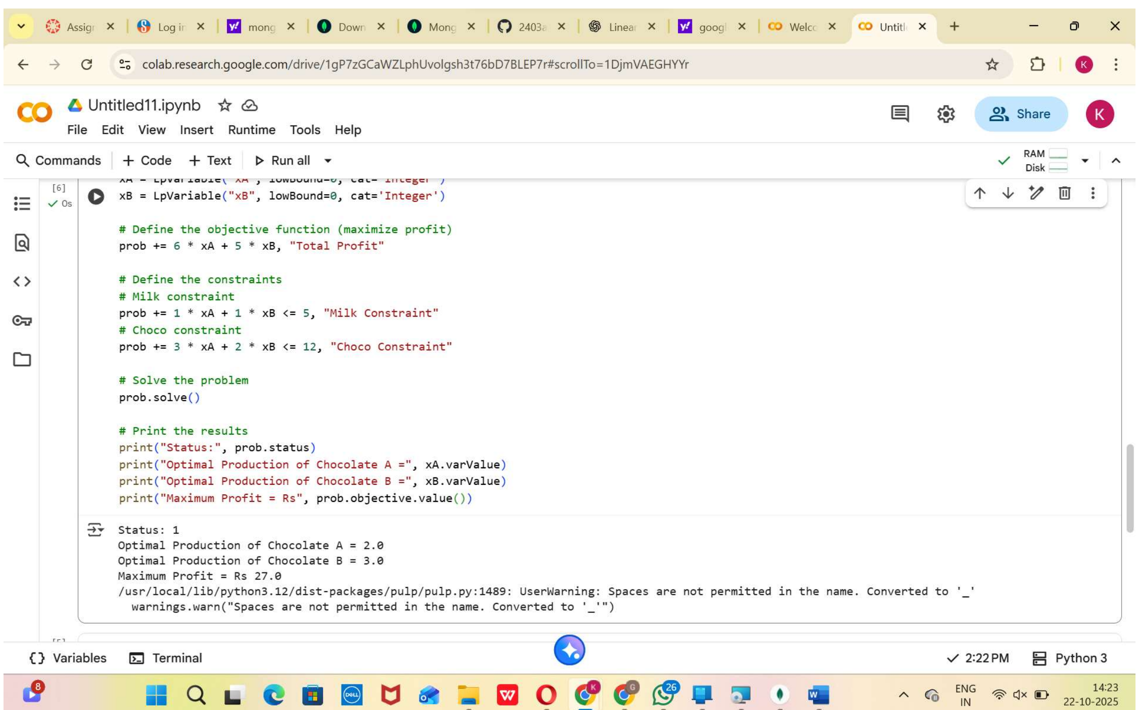
Consider a chocolate manufacturing company that produces only two types of chocolate i.e. A and B. Both the chocolates require Milk and Choco only.
To manufacture each unit of A and B, the following quantities are required:
Each unit of A requires 1 unit of Milk and 3 units of Choco
Each unit of B requires 1 unit of Milk and 2 units of Choco
The company kitchen has a total of 5 units of Milk and 12 units of Choco. On each sale, the company makes a profit of Rs 6 per unit A sold and Rs 5 per unit B sold.
Now, the company wishes to maximize its profit. How many units of A and B should it produce respectively?

Prompt:

Write a Python program that uses linear optimization to solve the following problem: A chocolate manufacturing company produces two types of chocolates: A and B. Both require Milk and Choco. Each unit of chocolate A needs 1 unit of Milk and 3 units of Choco, while each unit of chocolate B needs 1 unit of Milk and 2 units of Choco. The company has 5 units of Milk and 12 units of Choco available. The profit is Rs 6 per unit of A and Rs 5 per unit of B. Use linear optimization to calculate how many units of A and B should be produced to maximize profit. Provide the Python code and display the optimal solution.

Code&Output:





Observation:

This code uses the pulp library to solve a linear programming problem. It defines variables for the quantities of chocolates A and B. The objective is to maximize profit, subject to constraints on available milk and chocolate. The solver finds the optimal production levels to achieve the highest profit.

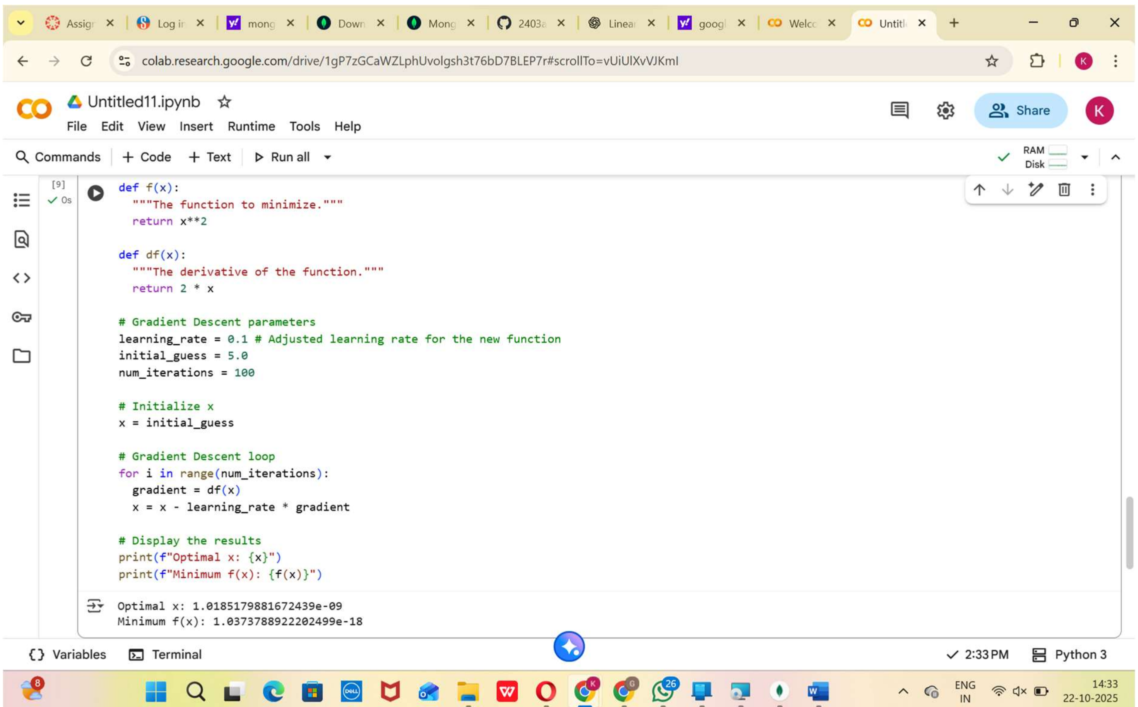
Task Description #4 – Gradient Descent Optimization

Task: Write python code to find value of x at which the function
f(x)=2X3+4x+5 will be minimum.

Prompt:

Write a Python program that applies Gradient Descent to minimize the function f(x) = 2x^3 + 4x + 5. Include the derivative, learning rate, initial guess, update loop, and display the optimal x and minimum f(x)

Code&Output:



```
def f(x):
  """The function to minimize."""
  return x**2

def df(x):
  """The derivative of the function."""
  return 2 * x

# Gradient Descent parameters
learning_rate = 0.1 # Adjusted learning rate for the new function
initial_guess = 5.0
num_iterations = 100

# Initialize x
x = initial_guess

# Gradient Descent loop
for i in range(num_iterations):
  gradient = df(x)
  x = x - learning_rate * gradient

# Display the results
print(f"Optimal x: {x}")
print(f"Minimum f(x): {f(x)}")
```

```
Optimal x: 1.0185179881672439e-09
Minimum f(x): 1.0373788922202499e-18
```

Observation:

This code implements Gradient Descent to minimize f(x) = x^2. It iteratively updates 'x' using the derivative and a learning rate. The algorithm effectively converges towards the minimum at x=0. This demonstrates how Gradient Descent finds the optimal value for convex functions.