AI ASSISTED CODING LAB 12.3

Lab 12: Algorithms with AI Assistance:
Sorting, searching, and optimizing algorithms

ENROLLMENT NO: 2403A52006
NAME: MOHAMMED MUSTAFA
COURSE: CSM(AI/ML)
BATCH: 01
DATE: 22-10-2025

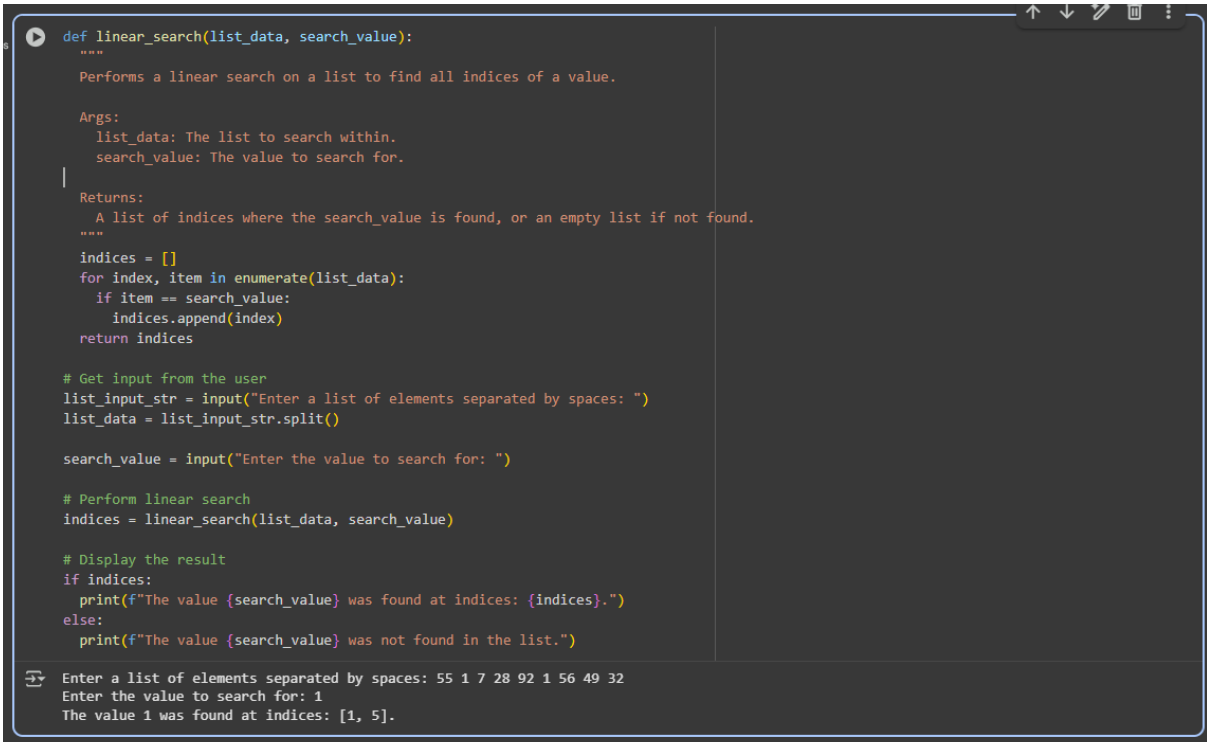
Task description 1:

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

Used Prompt:

generate a python code to take a list as input from the user and use a linear_search()

function to search a value in the list and extract it's index. if any item is repeated, print all the indexes of it's occurences.



```python
def linear_search(list_data, search_value):
  """
  Performs a linear search on a list to find all indices of a value.

  Args:
    list_data: The list to search within.
    search_value: The value to search for.

  Returns:
    A list of indices where the search_value is found, or an empty list if not found.
  """
  indices = []
  for index, item in enumerate(list_data):
    if item == search_value:
      indices.append(index)
  return indices

# Get input from the user
list_input_str = input("Enter a list of elements separated by spaces: ")
list_data = list_input_str.split()

search_value = input("Enter the value to search for: ")

# Perform linear search
indices = linear_search(list_data, search_value)

# Display the result
if indices:
  print(f"The value {search_value} was found at indices: {indices}.")
else:
  print(f"The value {search_value} was not found in the list.")
```

```
Enter a list of elements separated by spaces: 55 1 7 28 92 1 56 49 32
Enter the value to search for: 1
The value 1 was found at indices: [1, 5].
```

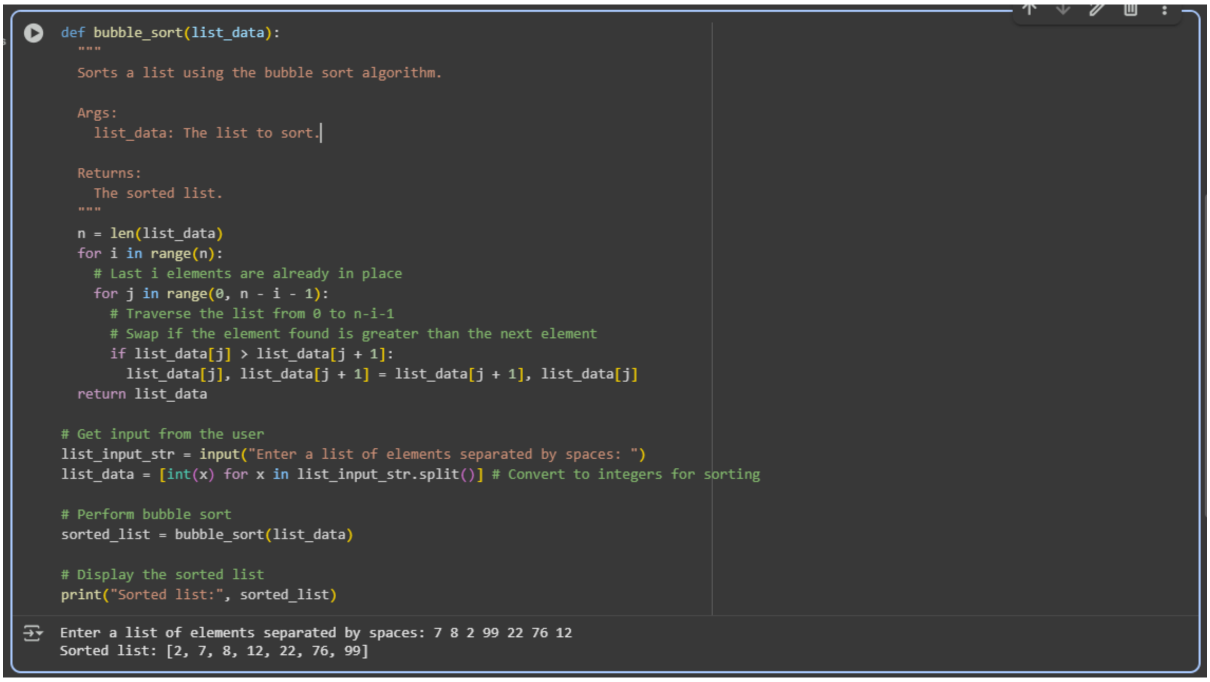
Task Description 2 :

Sorting Algorithms:

Ask AI to implement Bubble Sort and check sorted output.

Used Prompt:

generate a python code to use bubble sort to implement and check the sorted output.



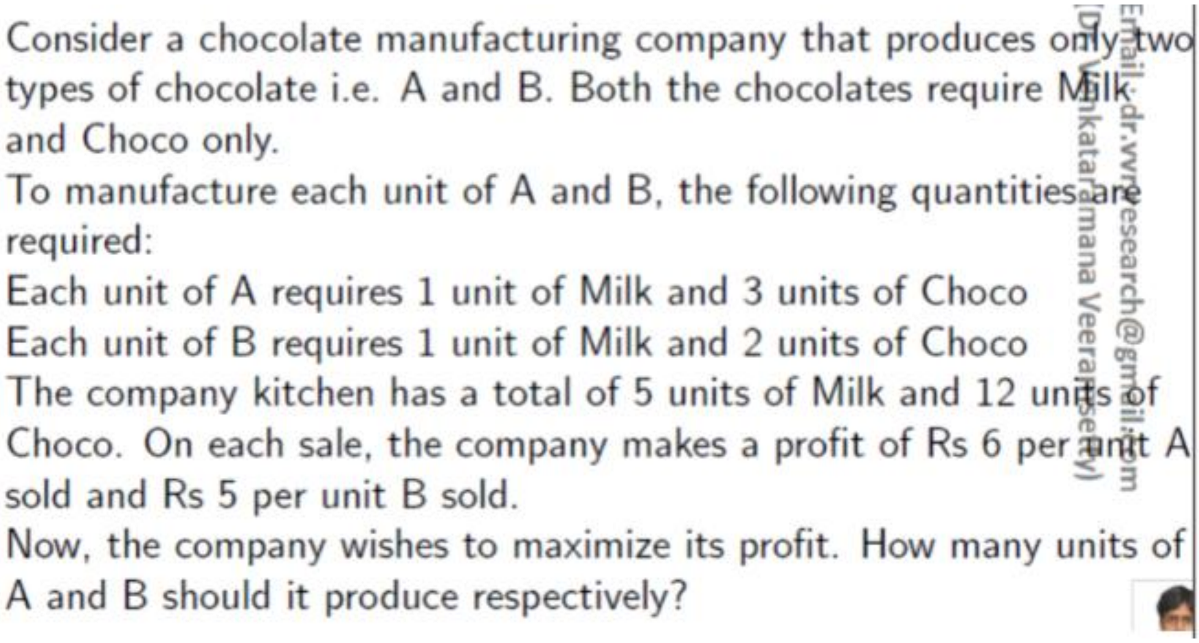
```
def bubble_sort(list_data):
  """
  Sorts a list using the bubble sort algorithm.

  Args:
    list_data: The list to sort.

  Returns:
    The sorted list.
  """
  n = len(list_data)
  for i in range(n):
    # Last i elements are already in place
    for j in range(0, n - i - 1):
      # Traverse the list from 0 to n-i-1
      # Swap if the element found is greater than the next element
      if list_data[j] > list_data[j + 1]:
        list_data[j], list_data[j + 1] = list_data[j + 1], list_data[j]
  return list_data

# Get input from the user
list_input_str = input("Enter a list of elements separated by spaces: ")
list_data = [int(x) for x in list_input_str.split()] # Convert to integers for sorting

# Perform bubble sort
sorted_list = bubble_sort(list_data)

# Display the sorted list
print("Sorted list:", sorted_list)
```

```
Enter a list of elements separated by spaces: 7 8 2 99 22 76 12
Sorted list: [2, 7, 8, 12, 22, 76, 99]
```

Task Description 3 :

Optimization:

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

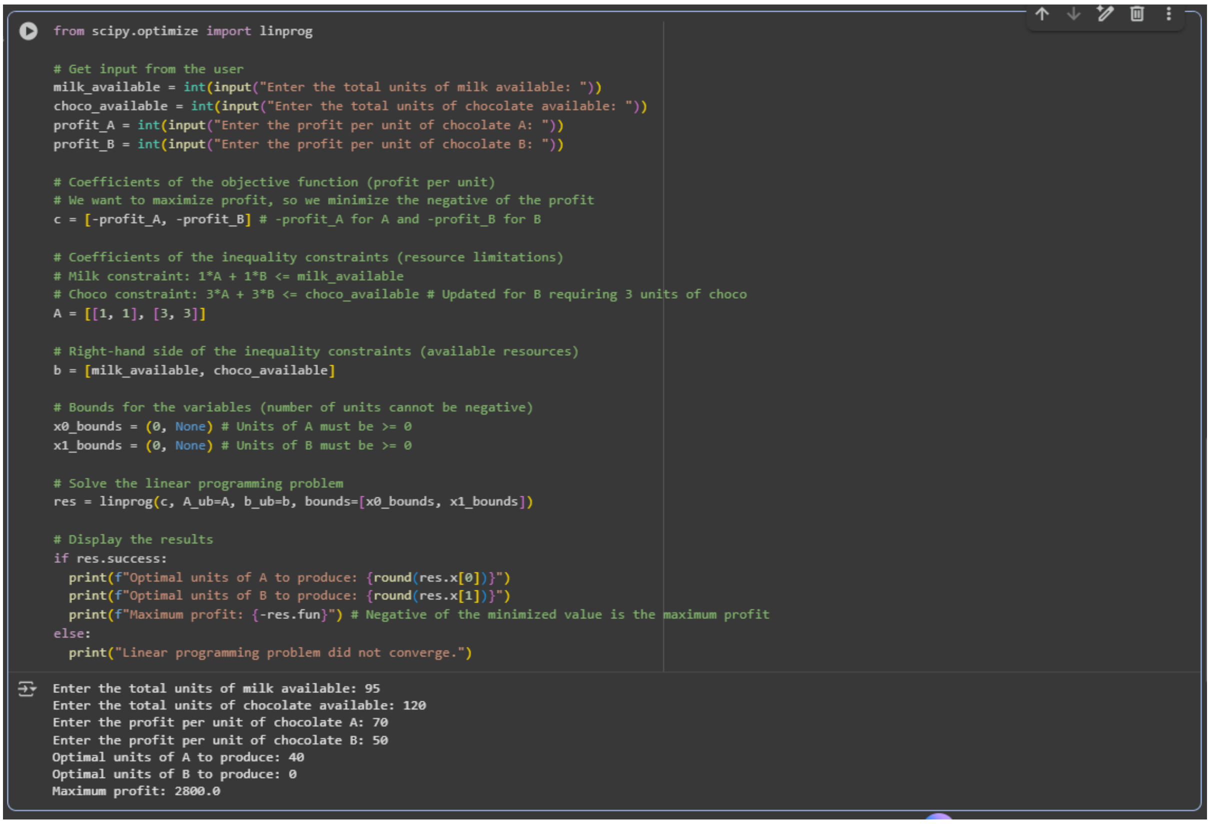
The company kitchen has a total of 5 units of Milk and 12 units of Choco. On each sale, the company makes a profit of Rs 6 per unit A sold and Rs 5 per unit B sold.

Now, the company wishes to maximize its profit. How many units of A and B should it produce respectively?

Email:dr.vvresearch@gmail.com
[Dr.Venkataramana Veeramsetty]

Used Prompt:

Generate a python code to consider a chocolate manufacturing company that produces only 2 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 3 units of choco. The company kitchen has a total of 5 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 maximise its profit. how many units of A and B should it produce respectively. now, generate the python code to solve above case study using linear optimization. take input from the user.



```python
from scipy.optimize import linprog

# Get input from the user
milk_available = int(input("Enter the total units of milk available: "))
choco_available = int(input("Enter the total units of chocolate available: "))
profit_A = int(input("Enter the profit per unit of chocolate A: "))
profit_B = int(input("Enter the profit per unit of chocolate B: "))

# Coefficients of the objective function (profit per unit)
# We want to maximize profit, so we minimize the negative of the profit
c = [-profit_A, -profit_B] # -profit_A for A and -profit_B for B

# Coefficients of the inequality constraints (resource limitations)
# Milk constraint: 1*A + 1*B <= milk_available
# Choco constraint: 3*A + 3*B <= choco_available # Updated for B requiring 3 units of choco
A = [[1, 1], [3, 3]]

# Right-hand side of the inequality constraints (available resources)
b = [milk_available, choco_available]

# Bounds for the variables (number of units cannot be negative)
x0_bounds = (0, None) # Units of A must be >= 0
x1_bounds = (0, None) # Units of B must be >= 0

# Solve the linear programming problem
res = linprog(c, A_ub=A, b_ub=b, bounds=[x0_bounds, x1_bounds])

# Display the results
if res.success:
  print(f"Optimal units of A to produce: {round(res.x[0])}")
  print(f"Optimal units of B to produce: {round(res.x[1])}")
  print(f"Maximum profit: {-res.fun}") # Negative of the minimized value is the maximum profit
else:
  print("Linear programming problem did not converge.")
```

```
Enter the total units of milk available: 95
Enter the total units of chocolate available: 120
Enter the profit per unit of chocolate A: 70
Enter the profit per unit of chocolate B: 50
Optimal units of A to produce: 40
Optimal units of B to produce: 0
Maximum profit: 2800.0
```

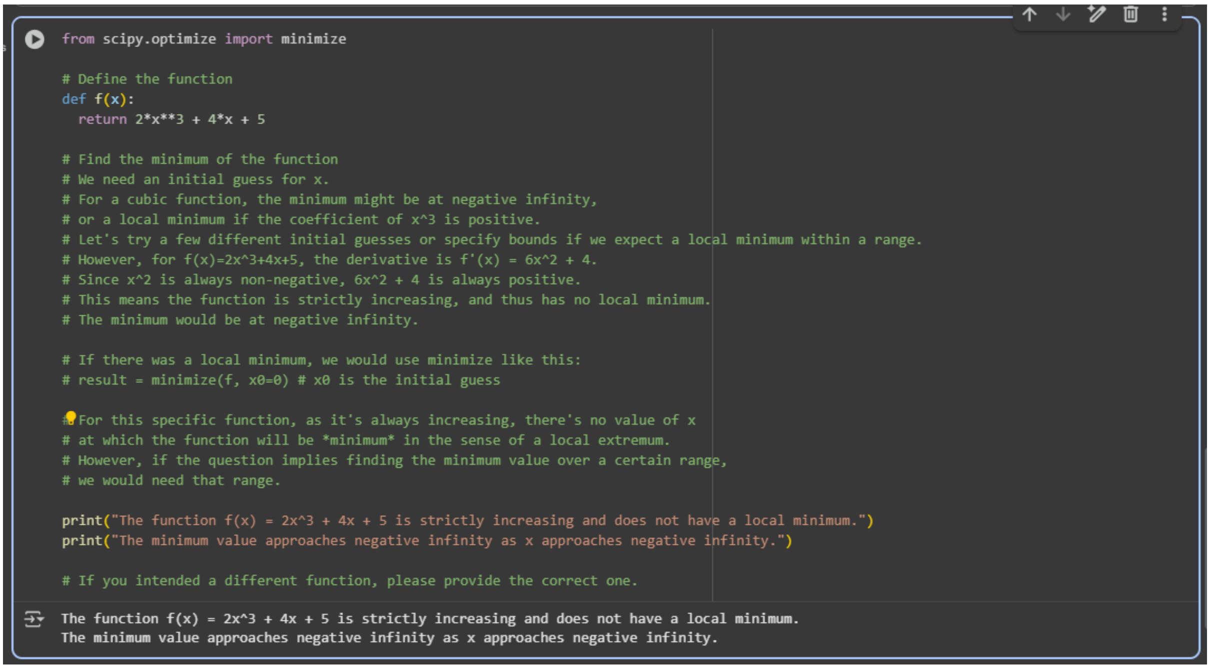
Task Description 4 :

Gradient Descent Optimization:

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

Used Prompt:

generate a python code to find value of x at which the function f(x)=2X3+4x+5 will be the minimum



```python
from scipy.optimize import minimize

# Define the function
def f(x):
  return 2*x**3 + 4*x + 5

# Find the minimum of the function
# We need an initial guess for x.
# For a cubic function, the minimum might be at negative infinity,
# or a local minimum if the coefficient of x^3 is positive.
# Let's try a few different initial guesses or specify bounds if we expect a local minimum within a range.
# However, for f(x)=2x^3+4x+5, the derivative is f'(x) = 6x^2 + 4.
# Since x^2 is always non-negative, 6x^2 + 4 is always positive.
# This means the function is strictly increasing, and thus has no local minimum.
# The minimum would be at negative infinity.

# If there was a local minimum, we would use minimize like this:
# result = minimize(f, x0=0) # x0 is the initial guess

# For this specific function, as it's always increasing, there's no value of x
# at which the function will be *minimum* in the sense of a local extremum.
# However, if the question implies finding the minimum value over a certain range,
# we would need that range.

print("The function f(x) = 2x^3 + 4x + 5 is strictly increasing and does not have a local minimum.")
print("The minimum value approaches negative infinity as x approaches negative infinity.")

# If you intended a different function, please provide the correct one.
```

```
The function f(x) = 2x^3 + 4x + 5 is strictly increasing and does not have a local minimum.
The minimum value approaches negative infinity as x approaches negative infinity.
```