AI ASSISTED CODING

ASSIGNMENT: 12.3

NAME: KOUDAGANI SAHITHYA

HT NO: 2403A52063

BATCH NO: 02

TASK 1~

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

PROMPT:

Write Python code to implement a linear_search() function that searches for a value in a list and returns its index

CODE & OUTPUT:

```python
def linear_search(data, value):
    """
    Searches for a value in a list using linear search and returns its index.

    Args:
        data: The list to search within.
        value: The value to search for.

    Returns:
        The index of the value if found, otherwise -1.
    """
    for index, item in enumerate(data):
        if item == value:
            return index
    return -1

# Example usage:
my_list = [10, 20, 30, 40, 50]
search_value = 30
index = linear_search(my_list, search_value)

if index != -1:
    print(f"Value {search_value} found at index {index}")
else:
    print(f"Value {search_value} not found in the list")

search_value = 60
index = linear_search(my_list, search_value)

if index != -1:
    print(f"Value {search_value} found at index {index}")
else:
    print(f"Value {search_value} not found in the list")
```

```
Value 30 found at index 2
Value 60 not found in the list
```

CODE EXPLANATION:

This code defines a linear_search function that iterates through a list (data) to find a specific value. It returns the index if the value is found, otherwise it returns -1. The example usage demonstrates searching for values

present and not present in a sample list, printing the results

TASK 2~

Ask AI to implement Bubble Sort and check sorted output.

PROMPT:

Write a Python program to implement Bubble Sort and display the sorted output.

CODE & OUTPUT:

```python
def bubble_sort(data):
    """
    Sorts a list using the Bubble Sort algorithm.

    Args:
        data: The list to be sorted.

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

# Example usage:
my_list_to_sort = [64, 34, 25, 12, 22, 11, 90]
sorted_list = bubble_sort(my_list_to_sort)

print("Original list:", [64, 34, 25, 12, 22, 11, 90]) # Print original list for comparison
print("Sorted list:", sorted_list)
```

```
Original list: [64, 34, 25, 12, 22, 11, 90]
Sorted list: [11, 12, 22, 25, 34, 64, 90]
```

CODE EXPLANATION:

The 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. This process is repeated until the list is sorted. The outer loop controls the number of passes, and the inner loop performs the comparisons and swaps. Finally, the sorted list is returned and printed along with the original list for comparison

TASK 3~

Write python code to solve below case study using linear optimization

Consider a chocolate manufacturing company that produces only wo 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 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 unes Choco. On each sale, the company makes a profit of Rs 6 per A sold and Rs 5 per unit B sold.

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

PROMPT:

Write Python code using linear optimization to solve this problem:

A chocolate company makes two types of chocolates, A and B.
Each unit of A needs 1 unit of Milk and 3 units of Choco.
Each unit of B needs 1 unit of Milk and 2 units of Choco.
Available resources: 5 units of Milk and 12 units of Choco.
Profit: Rs 6 per A, Rs 5 per B.
Find how many units of A and B should be produced to maximize profit

CODE & OUTPUT:

```python
from pulp import *

# Create the LP problem object
prob = LpProblem("Chocolate Manufacturing Problem", LpMaximize)

# Define the decision variables
A = LpVariable("A", 0, None, LpInteger)
B = LpVariable("B", 0, None, LpInteger)

# Define the objective function
prob += 6 * A + 5 * B, "Total Profit"

# Define the constraints
prob += 1 * A + 1 * B <= 5, "Milk Constraint"
prob += 3 * A + 2 * B <= 12, "Choco Constraint"

# Solve the problem
prob.solve()

# Print the status of the solution
print("Status:", LpStatus[prob.status])

# Print the optimal values of the decision variables
print("Optimal number of units of A:", A.varValue)
print("Optimal number of units of B:", B.varValue)

# Print the maximum profit
print("Maximum Profit:", value(prob.objective))
```

```
Status: Optimal
Optimal number of units of A: 2.0
Optimal number of units of B: 3.0
Maximum Profit: 27.0
/usr/local/lib/python3.12/dist-packages/pulp/pulp.py:1489: UserWarning: Spaces are not permitted in the name.
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
```

CODE EXPLANATION:

This code uses the pulp library to solve a chocolate manufacturing linear optimization problem. It defines decision variables (units of A and B), the objective function (maximize profit), and constraints (milk and choco limitations). It then solves the problem and prints the solution status, optimal production quantities for A and B, and the resulting maximum profit.

TASK 4~

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

PROMPT:

Use Python to Zcalculate the x value where the function f(x) = 2*x**3 + 4x + 5* reaches its minimum point.

CODE & OUTPUT:

```python
import numpy as np
from scipy.optimize import minimize

def f(x):
  """
  The function to be minimized.
  """
  return 2 * x**3 + 4 * x + 5

# Find the minimum of the function
initial_guess = 0
result = minimize(f, initial_guess)

# The result object contains information about the optimization process
# The optimal value of x is stored in result.x
optimal_x = result.x[0] # result.x is an array, so we take the first element

# Print the optimal value of x
print(f"The function is minimized at x = {optimal_x}")
```

The function is minimized at x = -1034.24

CODE EXPLANATION:

This code uses scipy.optimize.minimize to find where the function f(x) = 2x³ + 4x + 5 is smallest. It defines the function, sets an initial guess for x, runs the

optimization, and prints the resulting x value where the minimum is found.