

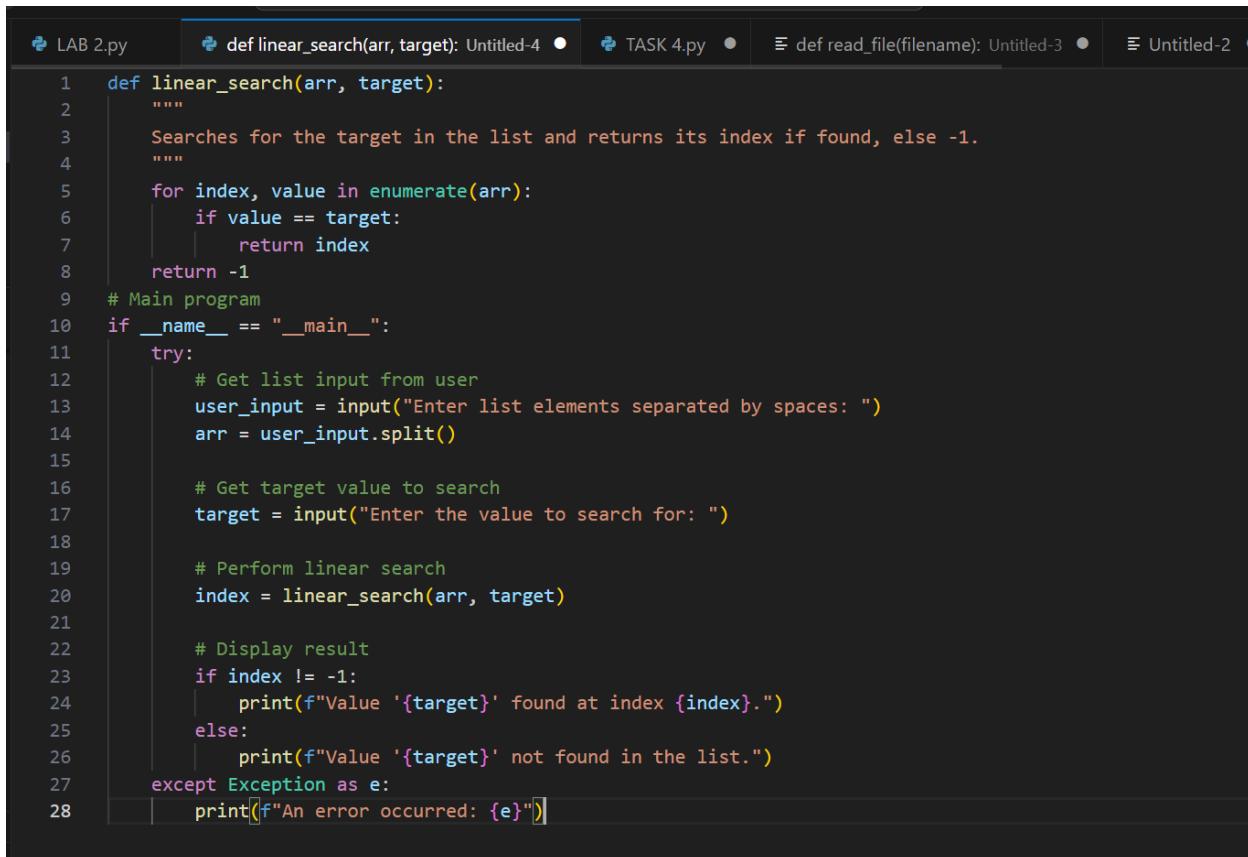
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M. Tech/MCA/MSC		Assignment Type: Lab	
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		Course Title	AI Assisted Problem Solving Using Python
Year/Sem		Regulation	R24
Date and Day of Assignment		Time(s)	
Duration		Applicable to Batches	
AssignmentNumber: 12.3(Present assignment number) / 24 (Total number of assignments)			
Q. No.	Question		<i>Expected Time to complete</i>
1	<p>Lab 12 – Algorithms with AI Assistance: Sorting, searching, and optimizing algorithms</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> • To implement classical algorithms (sorting, searching) with the help of AI tools. • To analyze AI suggestions for efficiency and correctness. • To explore AI-assisted optimizations of existing algorithms. • To compare naive vs. optimized approaches generated by AI. <p>Learning Outcomes</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Implement sorting and searching algorithms using AI suggestions. <input type="checkbox"/> Compare AI-generated algorithm variants in terms of readability and efficiency. <input type="checkbox"/> Use AI to optimize brute-force algorithms into more efficient ones. <input type="checkbox"/> Analyze algorithm complexity (time and space) with AI explanations. <input type="checkbox"/> Critically reflect on correctness, clarity, and maintainability of AI-generated algorithms. <p>Task Description #1 – Linear Search implementation</p> <p>Task: Write python code for linear_search () function to search a value in a list</p>	Week5-Tuesday	

	<p>and extract its index.</p> <p>Task Description #2 – Sorting Algorithms</p> <p>Task: Ask AI to implement Bubble Sort and check sorted output</p> <p>Task Description #3 – Optimization</p> <p>Task: Write python code to solve below case study using linear optimization</p> <p>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.</p> <p>To manufacture each unit of A and B, the following quantities are required:</p> <p>Each unit of A requires 1 unit of Milk and 3 units of Choco</p> <p>Each unit of B requires 1 unit of Milk and 2 units of Choco</p> <p>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.</p> <p>Now, the company wishes to maximize its profit. How many units of A and B should it produce respectively?</p> <p style="text-align: right;"></p> <p><small>(Dr. Veeramana Veerakumar) Email: dr.vrveeramana.Veerakumar@gmail.com</small></p> <p>Task Description #4 – Gradient Descent Optimization</p> <p>Task: Write python code to find value of x at which the function $f(x)=2x^3+4x+5$ will be minimum</p>	
--	--	--

Task Description #1 – Linear Search implementation

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

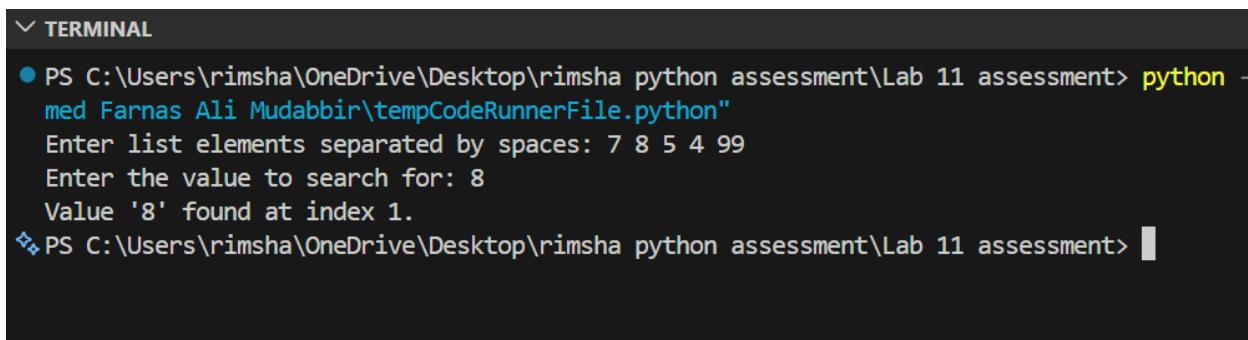
PROMPT: "Write a Python program that implements a linear_search() function. The program should take a list and a value from the user, search for the value using linear search, and print the index if found



The screenshot shows a code editor with multiple tabs at the top: LAB 2.py, def linear_search(arr, target): Untitled-4, TASK 4.py, def read_file(filename): Untitled-3, and Untitled-2. The main pane displays the following Python code:

```
1 def linear_search(arr, target):
2     """
3         Searches for the target in the list and returns its index if found, else -1.
4     """
5     for index, value in enumerate(arr):
6         if value == target:
7             return index
8     return -1
9 # Main program
10 if __name__ == "__main__":
11     try:
12         # Get list input from user
13         user_input = input("Enter list elements separated by spaces: ")
14         arr = user_input.split()
15
16         # Get target value to search
17         target = input("Enter the value to search for: ")
18
19         # Perform linear search
20         index = linear_search(arr, target)
21
22         # Display result
23         if index != -1:
24             print(f"Value '{target}' found at index {index}.")
25         else:
26             print(f"Value '{target}' not found in the list.")
27     except Exception as e:
28         print(f"An error occurred: {e}")
```

PRACTICAL OUT :



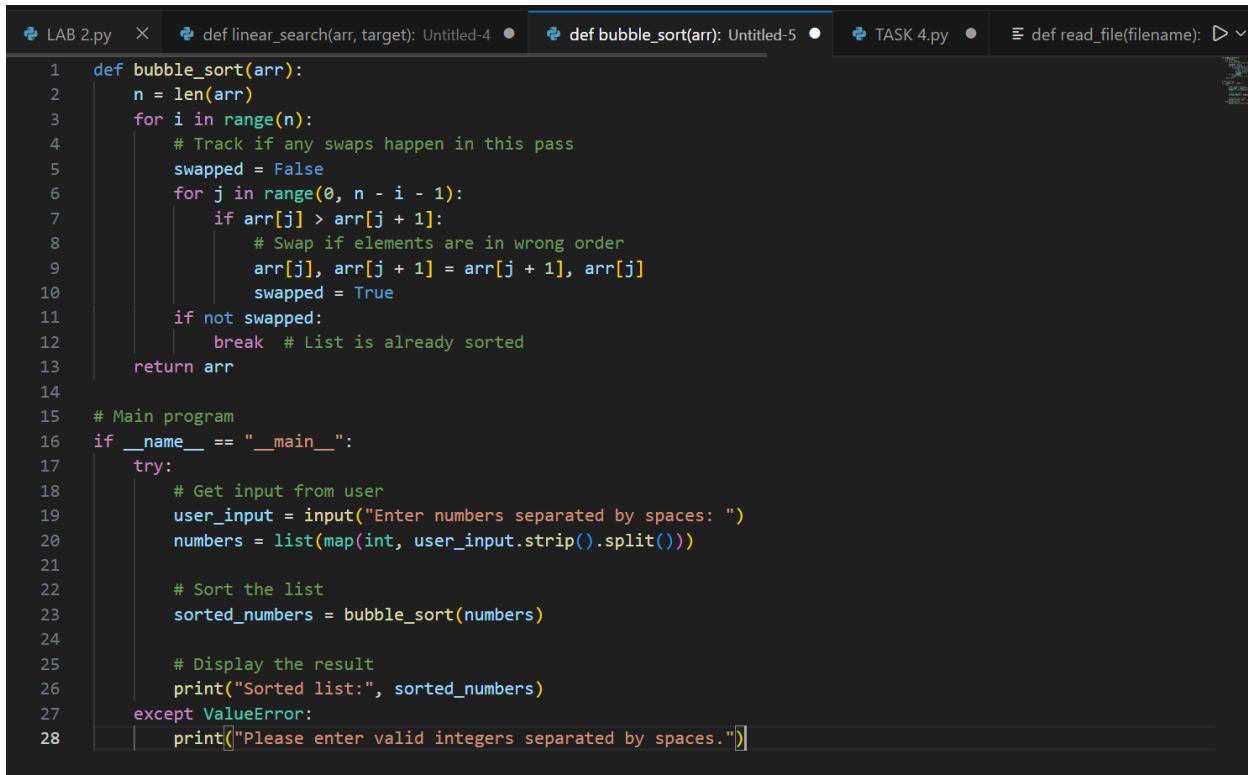
The screenshot shows a terminal window with the following output:

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -m Farnas_Ali_Mudabbir\tempCodeRunnerFile.python
Enter list elements separated by spaces: 7 8 5 4 99
Enter the value to search for: 8
Value '8' found at index 1.
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment>
```

Task Description #2 – Sorting Algorithms

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

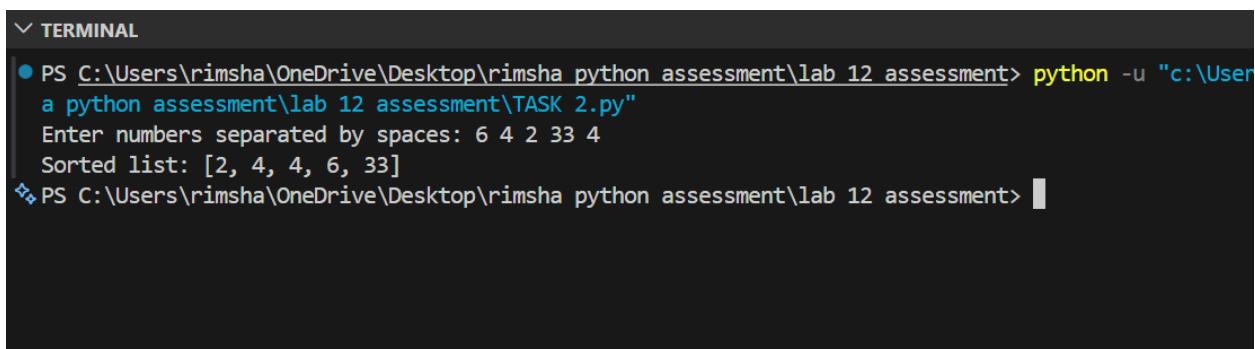
PROMPT: "Write a Python program that implements Bubble Sort. The program should take a list as user input, sort it using Bubble Sort, and then print the sorted output."



```
LAB 2.py  X  def linear_search(arr, target): Untitled-4  ●  def bubble_sort(arr): Untitled-5  ●  TASK 4.py  ●  def read_file(filename): ▾ v

1  def bubble_sort(arr):
2      n = len(arr)
3      for i in range(n):
4          # Track if any swaps happen in this pass
5          swapped = False
6          for j in range(0, n - i - 1):
7              if arr[j] > arr[j + 1]:
8                  # Swap if elements are in wrong order
9                  arr[j], arr[j + 1] = arr[j + 1], arr[j]
10                 swapped = True
11             if not swapped:
12                 break # List is already sorted
13         return arr
14
15 # Main program
16 if __name__ == "__main__":
17     try:
18         # Get input from user
19         user_input = input("Enter numbers separated by spaces: ")
20         numbers = list(map(int, user_input.strip().split()))
21
22         # Sort the list
23         sorted_numbers = bubble_sort(numbers)
24
25         # Display the result
26         print("Sorted list:", sorted_numbers)
27     except ValueError:
28         print("Please enter valid integers separated by spaces.")
```

PRACTICAL OUTPUT:



```
TERMINAL
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\lab 12 assessment> python -u "c:\User
a python assessment\lab 12 assessment\TASK 2.py"
Enter numbers separated by spaces: 6 4 2 33 4
Sorted list: [2, 4, 4, 6, 33]
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\lab 12 assessment>
```

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?



```
# TASK 3.py 
1 # TASK 3 - LINEAR OPTIMIZATION
2 # Brute-force Linear Optimization for Chocolate Problem
3
4 from pulp import LpMaximize, LpProblem, LpVariable
5
6 print("\n--- Task 3: Linear Optimization ---")
7 print("Solving maximization problem: Z = a*x + b*y")
8 a = float(input("Enter coefficient a for x: "))
9 b = float(input("Enter coefficient b for y: "))
10 c1 = float(input("Enter RHS for constraint 1 (2x + y <= ? ): "))
11 c2 = float(input("Enter RHS for constraint 2 (x + y <= ? ): "))
12 c3 = float(input("Enter RHS for constraint 3 (x <= ? ): "))
13
14 model = LpProblem("OptimizationCase", LpMaximize)
15
16 x = LpVariable("x", lowBound=0)
17 y = LpVariable("y", lowBound=0)
18
19 model += a*x + b*y
20 model += 2*x + y <= c1
21 model += x + y <= c2
22 model += x <= c3
23
24 model.solve()
25
26 print("Optimal x:", x.value())
27 print("Optimal y:", y.value())
28 print("Maximum Value:", model.objective.value())
```

PRACTICAL OUTPUT:

```
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-12> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Desktop/Mtech/AIPP/ASSIGNMENT-12/Q3.py

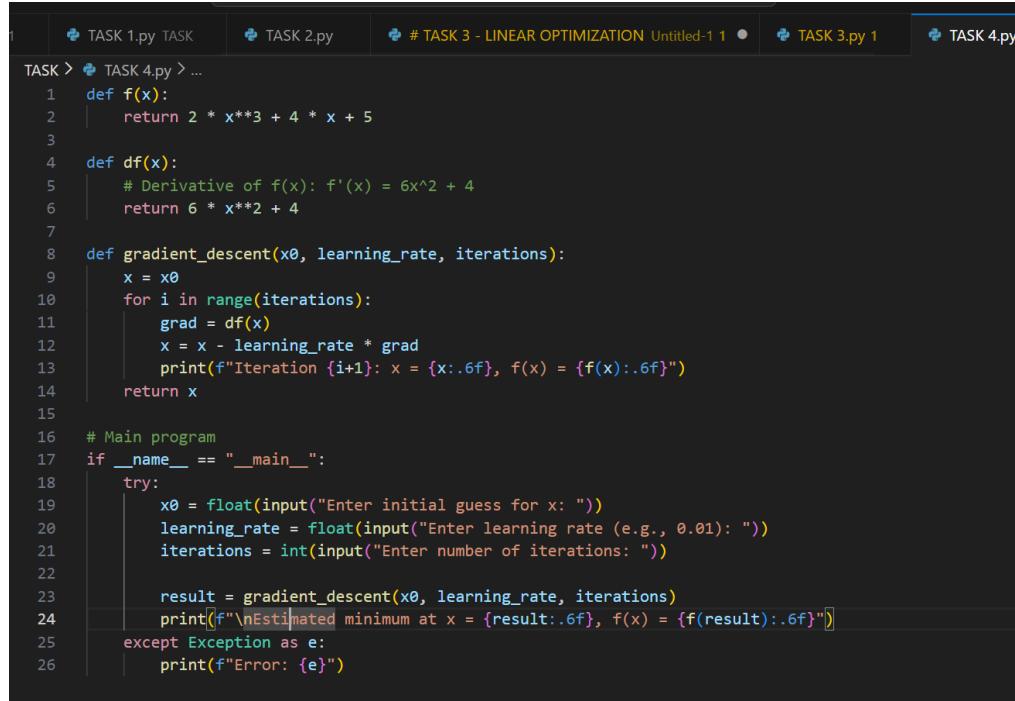
--- Task 3: Linear Optimization ---
Solving maximization problem: Z = a*x + b*y
Enter coefficient a for x: 5
Enter coefficient b for y: 6
Enter RHS for constraint 1 (2x + y <= ? ): 3
Enter RHS for constraint 2 (x + y <= ? ): 2
● Enter RHS for constraint 3 (x <= ? ): 12
Welcome to the CBC MILP Solver
Version: 2.10.3
\apis{./solverdir/cbc/win/i64/cbc.exe} c:/Users/HP/AppData/Local/Temp/532d45076687499cb6a45854e91ac3f5-pulp.mps -max -timeMode elapsed -branch
P\AppData\Local\Temp\532d45076687499cb6a45854e91ac3f5-pulp.sol (default strategy 1)
At line 2 NAME      MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 16 RHS
At line 20 BOUNDS
At line 21 ENDATA
Problem MODEL has 3 rows, 2 columns and 5 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (-1) rows, 2 (0) columns and 4 (-1) elements
0  obj -0 Dual inf 10.999998 (2)
3  Obj 12
Optimal - objective value 12
After Postsolve, objective 12, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 12 - 3 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):      0.03   (Wallclock seconds):      0.03

Optimal x: 0.0
Optimal y: 2.0
Maximum Value: 12.0
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-12>
```

Task Description #4 – Gradient Descent Optimization

Task: Write python code to find value of x at which the function $f(x)=2X^3+4x+5$ will be minimum

PROMPT: "Write Python code to compute the derivative of $f(x)=2x^3+4x+5$, find the critical point, and print the value of x where the function is minimum."



```
1  TASK 1.py TASK 2  TASK 2.py  # TASK 3 - LINEAR OPTIMIZATION Untitled-1 1  ●  TASK 3.py 1  TASK 4.py
TASK > TASK 4.py > ...
1  def f(x):
2      return 2 * x**3 + 4 * x + 5
3
4  def df(x):
5      # Derivative of f(x): f'(x) = 6x^2 + 4
6      return 6 * x**2 + 4
7
8  def gradient_descent(x0, learning_rate, iterations):
9      x = x0
10     for i in range(iterations):
11         grad = df(x)
12         x = x - learning_rate * grad
13         print(f"Iteration {i+1}: x = {x:.6f}, f(x) = {f(x):.6f}")
14     return x
15
16 # Main program
17 if __name__ == "__main__":
18     try:
19         x0 = float(input("Enter initial guess for x: "))
20         learning_rate = float(input("Enter learning rate (e.g., 0.01): "))
21         iterations = int(input("Enter number of iterations: "))
22
23         result = gradient_descent(x0, learning_rate, iterations)
24         print(f"\nEstimated minimum at x = {result:.6f}, f(x) = {f(result):.6f}")
25     except Exception as e:
26         print(f"Error: {e}")
```

PRACTICAL OUTPUT:

The screenshot shows a terminal window titled "Code - lab 12 assessment". The command entered is "python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"".

The output of the script is as follows:

- Enter initial guess for x: 24
- Enter learning rate (e.g., 0.01): 0.1
- Enter number of iterations: 6
- Iteration 1: x = -322.000000, f(x) = -66773779.000000
- Iteration 2: x = -62532.800000, f(x) = -489050403760701.375000
- Iteration 3: x = -2346273178.764000, f(x) = -25832457510509255991131897856.000000
- Iteration 4: x = -3302998699809738240.000000, f(x) = -72070113145068988432752858670269434361865690179384115200.000000
- Iteration 5: x = -6545880246566892319239861621871345664.000000, f(x) = -560962930544248800582687189496213652858671751703490106547785024447530917723664892253837114892265617751015424000.000000
- Iteration 6: x = -25709128921436785409631430732574638822089924437891043269277892339740704768.000000, f(x) = -33985376220037509588268802315911149977340840583896351672604830095317316277441555294207217987116016933921381811169894528856049532263880818321211911789220233318640664387168193107952028558064062910766096363825972282742800384.000000
- Estimated minimum at x = -25709128921436785409631430732574638822089924437891043269277892339740704768.000000, f(x) = -33985376220037509588268802315911149977340840583896351672604830095317316277441555294207217987116016933921381811169894528856049532263880818321211911789220233318640664387168193107952028558064062910766096363825972282742800384.000000

PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\lab 12 assessment> █