**HALL TICKET NUMBER: 2403A51365**

**NAME: RAYAPUDI HARITHA**

**BATCH: 24BTCAICSB14**

**AssignmentNumber:12.5**

**Lab 12: Algorithms with AI Assistance – Sorting, Searching, and  
Optimizing Algorithms  
Lab Objectives:  
• Apply AI-assisted programming to implement and optimize  
sorting and searching algorithms.  
Week6 -  
Friday**

**• Compare different algorithms in terms of efficiency and use  
cases.  
• Understand how AI tools can suggest optimized code and  
complexity improvements.**

Task 1: Sorting Student Records for Placement Drive  
Scenario:  
SR University is preparing for a campus placement drive. The Training  
and Placement Cell needs student records sorted by CGPA in  
descending order to easily shortlist candidates.  
• Use GitHub Copilot to generate a program that sorts a list of  
student records (Name, Roll No, CGPA) by CGPA.  
• Implement both Quick Sort and Merge Sort using AI assistance.  
• Compare the runtime performance of both algorithms on large  
datasets.  
• Write a function that outputs the top 10 students with the highest  
CGPA.

**PROMPT:-**

SR University is organizing a campus placement drive and needs to prepare student records for shortlisting. Please write a Python program that: Takes a list of student records, where each record includes the student's Name, Roll Number, and CGPA. Implements both Quick Sort and Merge Sort algorithms to sort the student records by CGPA in descending order. Compares the runtime performance of Quick Sort and Merge Sort on a large dataset of student records. Includes a function that prints the top 10 students with the highest CGPA from the sorted list. Please use random data generation to simulate a large dataset (e.g., 50,000 students). Print the time taken by each sorting algorithm and display the details (Name, Roll No, CGPA) of the top 10 stude

**Code:-**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Output:-**

A screenshot of a computer

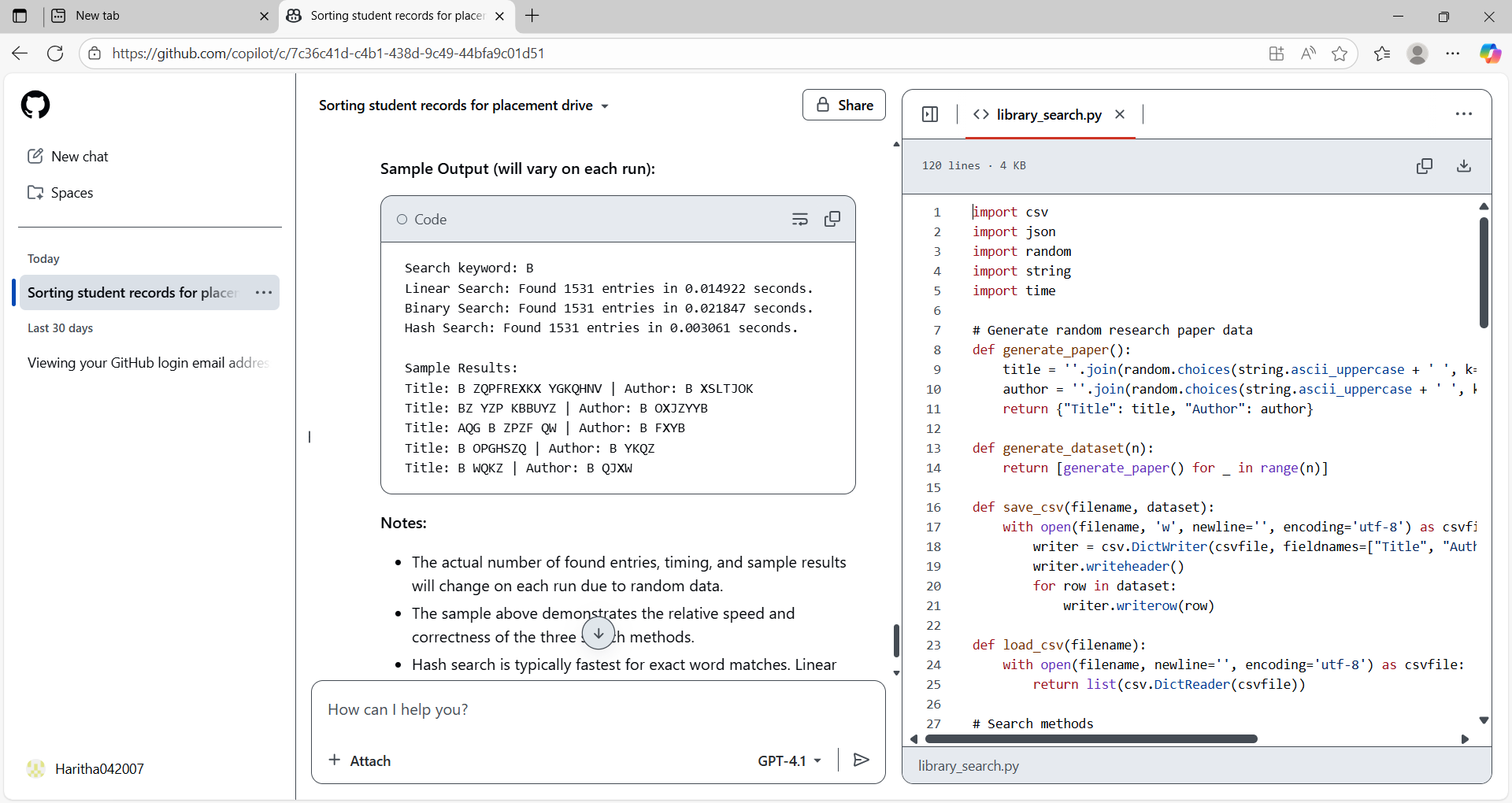
AI-generated content may be incorrect.

Task 2: Optimized Search in Online Library System  
Scenario:  
SR University’s digital library has thousands of research papers.  
Students frequently search for a paper by title or author name. The  
current linear search is too slow.  
• Use GitHub Copilot to implement Binary Search and Hash-  
based Search for faster lookups.  
• Load a dataset of book titles and authors (CSV or JSON file).  
• Allow the user to input a keyword and return all matching entries.  
• Compare the efficiency of linear search vs binary search vs  
hashing using test cases.

**Prompt:-**

SR University’s digital library system needs an efficient way to search for research papers by title or author. Please write a Python program that: Loads a dataset of research papers from a CSV or JSON file, where each record has at least a "Title" and "Author" field. Implements three search methods: Linear Search Binary Search (requires sorted data) Hash-based Search (using a hash table/dictionary) Allows a user to input a search keyword and returns all matching entries (either in the title or author). Compares the performance (e.g., time taken) of all three search methods using realistic test cases with a large dataset (e.g., thousands of records). Prints the time taken by each search and the found entries. Please use simulated/random data if a real dataset is not available.

**Code:-**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Output:-**

A screenshot of a computer

AI-generated content may be incorrect.

Task 3: Route Optimization for AUV Swarm  
Scenario:  
A research team at SR University is simulating Autonomous  
Underwater Vehicle (AUV) swarms. Each AUV must visit multiple  
underwater sensors, and the goal is to minimize travel distance (like  
the Traveling Salesman Problem).

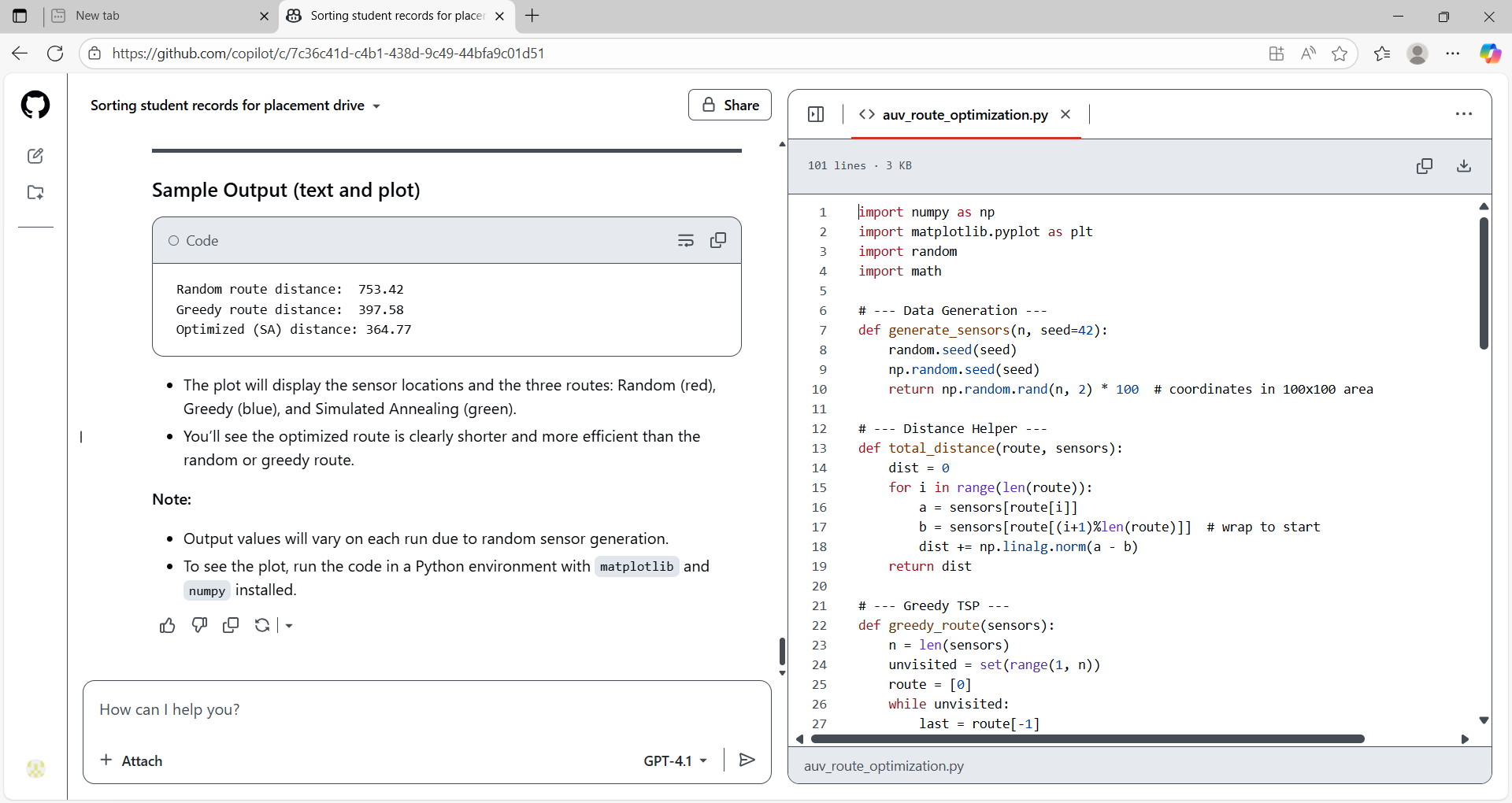
• With GitHub Copilot, implement an algorithm to optimize the  
route:  
1. Start with a Greedy approach.  
2. Improve with Genetic Algorithm (GA) or Simulated  
Annealing (SA).  
• Use a dataset of sensor coordinates (x, y).  
• Visualize the optimized route using a plotting library (e.g.,  
Matplotlib).  
• Compare the optimized solution with a random path in terms of  
distance trave

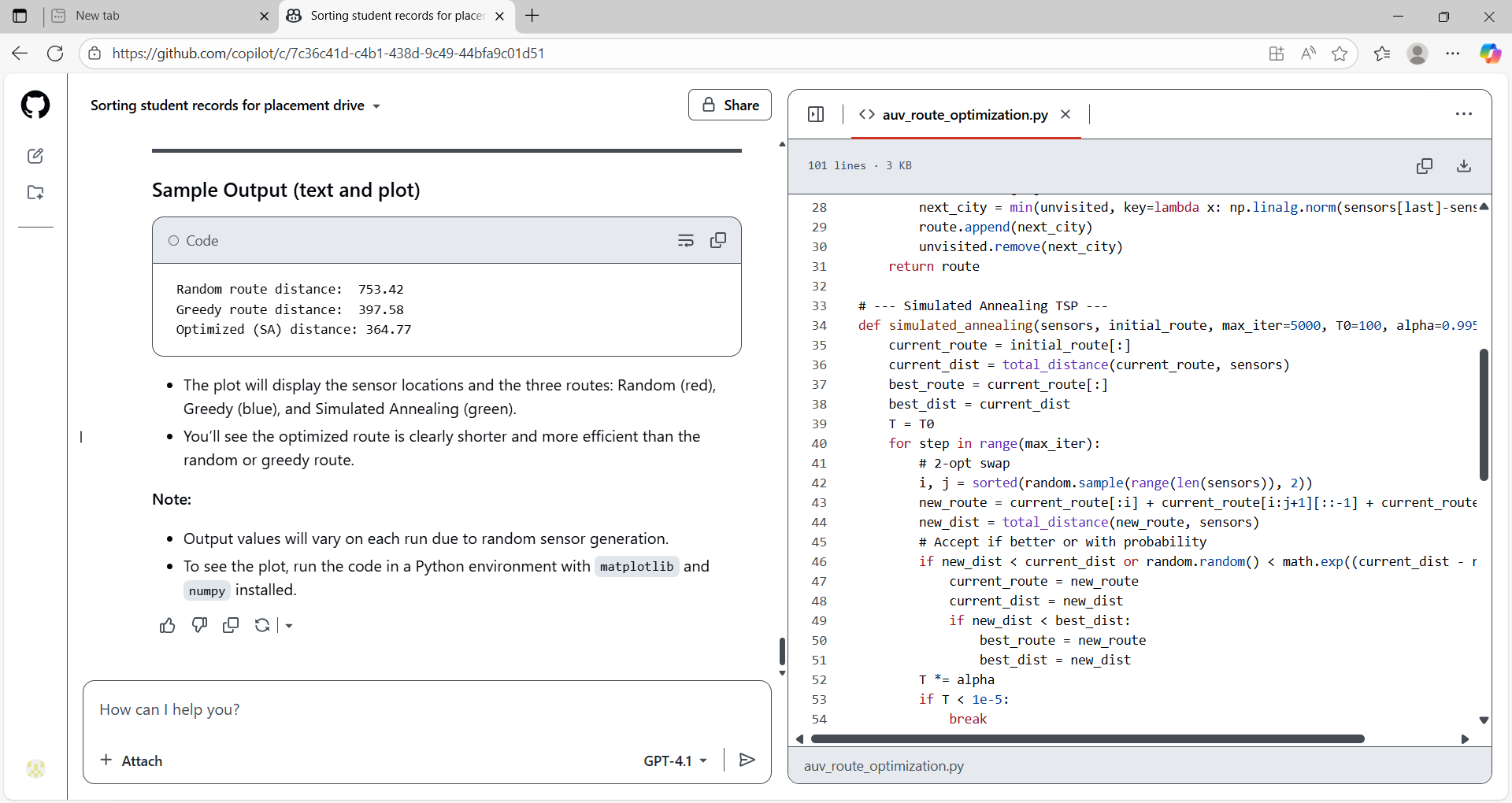
**Prompt:-**

SR University’s research team is simulating Autonomous Underwater Vehicle (AUV) swarms, where each AUV must visit multiple underwater sensors, aiming to minimize total travel distance (similar to the Traveling Salesman Problem). Please write a Python program that:

1. Loads or generates a dataset of underwater sensor coordinates (x, y).
2. Implements a Greedy algorithm to find an initial route visiting all sensors.
3. Improves the route using either a Genetic Algorithm (GA) or Simulated Annealing (SA).
4. Visualizes the sensor locations and the optimized route using Matplotlib.
5. Compares the total distance of the random route, Greedy route, and the optimized route.
6. Prints the total travel distance for each approach.

**Code:-**





A screenshot of a computer

AI-generated content may be incorrect.

**Output:-**

A screenshot of a computer

AI-generated content may be incorrect.

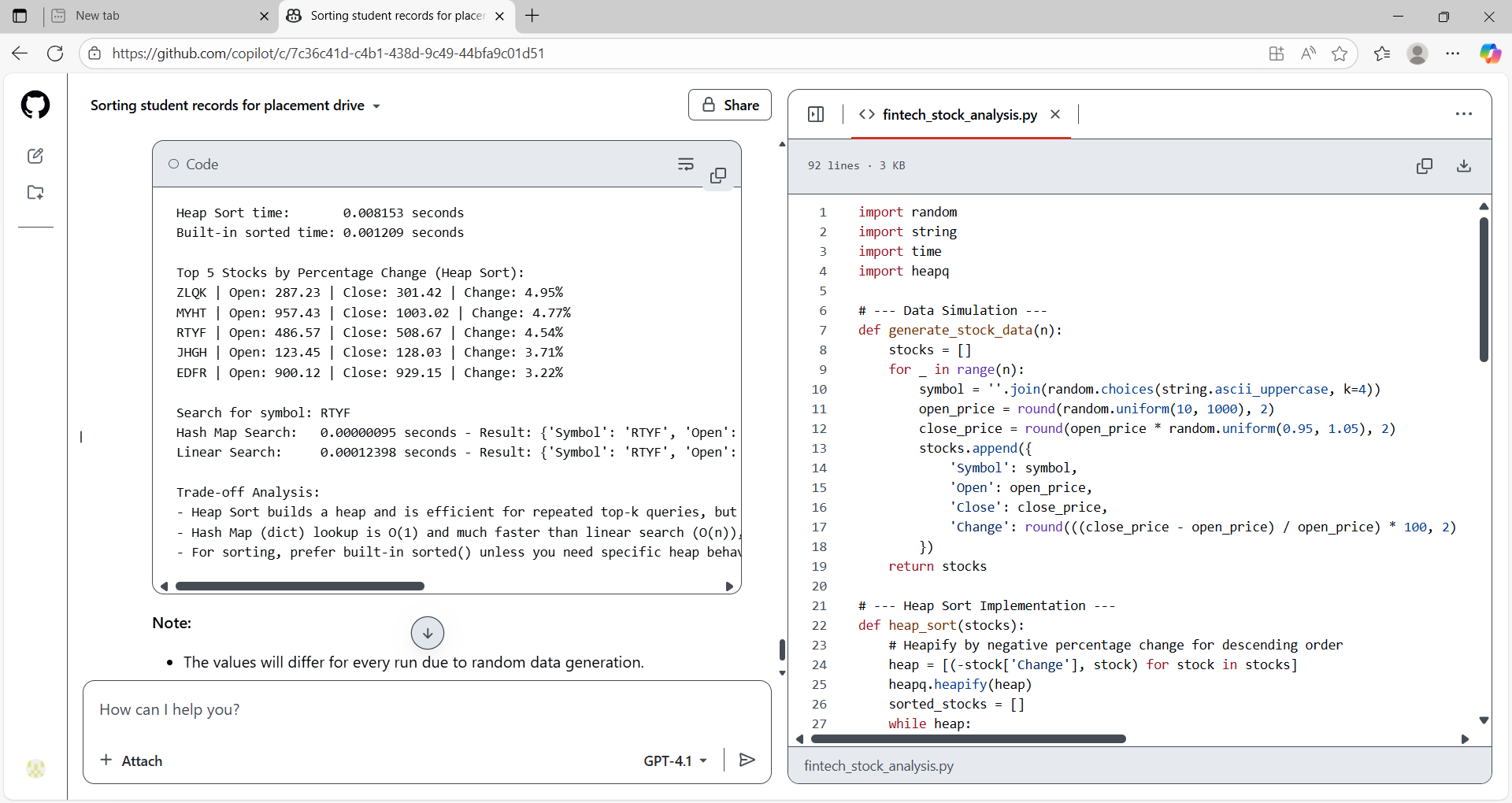
Task 4: Real-Time Stock Data Sorting & Searching  
Scenario:  
An AI-powered FinTech Lab at SR University is building a tool for  
analyzing stock price movements. The requirement is to quickly sort  
stocks by daily gain/loss and search for specific stock symbols  
efficiently.  
• Use GitHub Copilot to fetch or simulate stock price data (Stock  
Symbol, Opening Price, Closing Price).  
• Implement sorting algorithms to rank stocks by percentage  
change.  
• Implement a search function that retrieves stock data instantly  
when a stock symbol is entered.  
• Optimize sorting with Heap Sort and searching with Hash  
Maps.  
• Compare performance with standard library functions (sorted(),  
dict lookups) and analyze trade-offs.

**Prompt:-**

SR University’s AI-powered FinTech Lab needs a Python program to efficiently analyze and search stock price data. Please write a Python program that:

1. Fetches or simulates stock price data with each record containing: Stock Symbol, Opening Price, and Closing Price.
2. Calculates the daily percentage change for each stock and implements Heap Sort to rank stocks by this percentage gain/loss.
3. Implements a search function that retrieves stock data instantly when a stock symbol is entered, using a Hash Map (dictionary).
4. Compares the performance of Heap Sort with Python’s built-in sorted(), and compares Hash Map lookups with linear search for searching stock symbols.
5. Prints the sorted stock list and the time taken by each sorting/searching approach, and describes the trade-offs.

**Code:-**

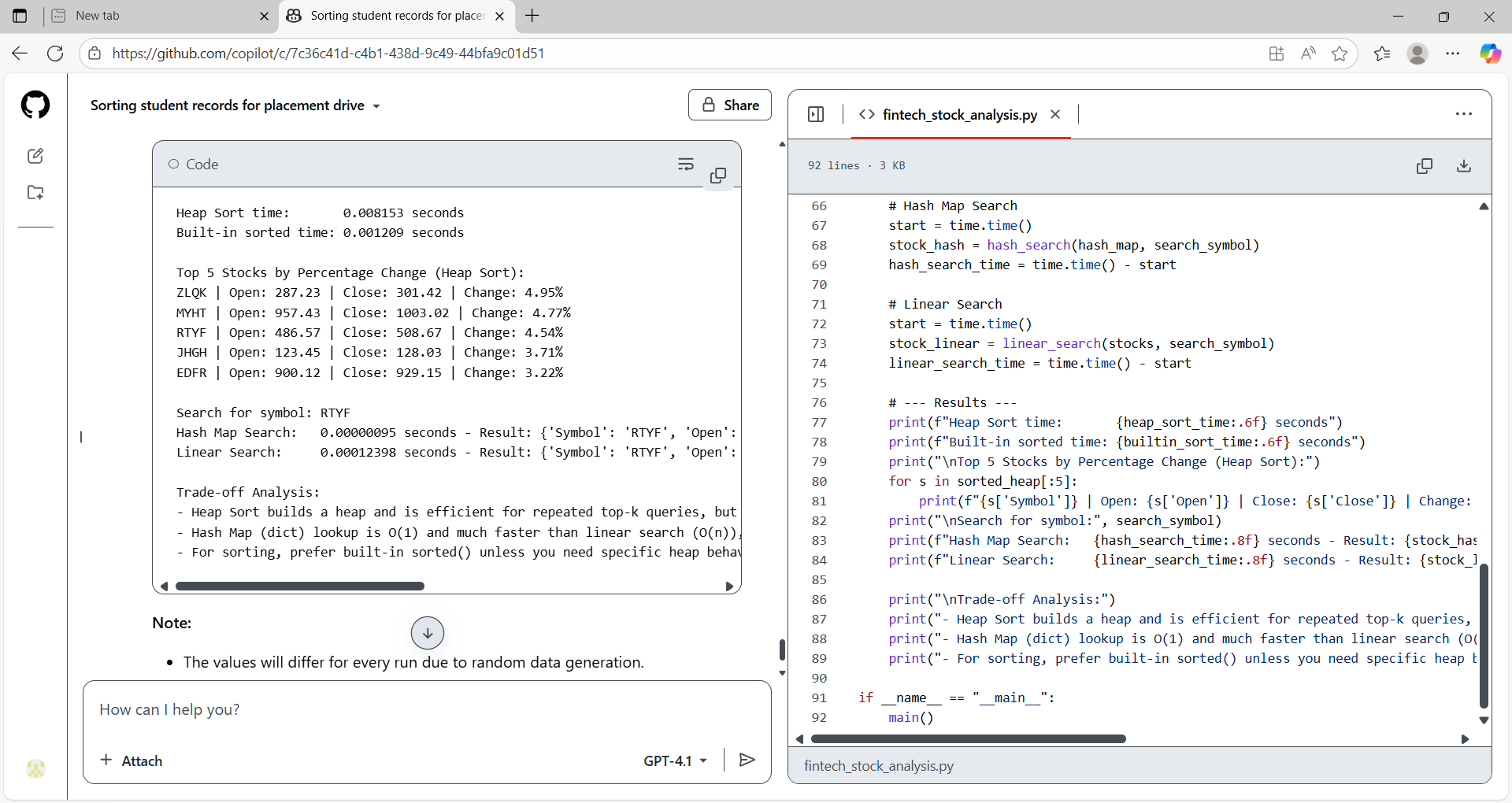
****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**Output:-**

**A screenshot of a computer

AI-generated content may be incorrect.**