Assignment :12.5

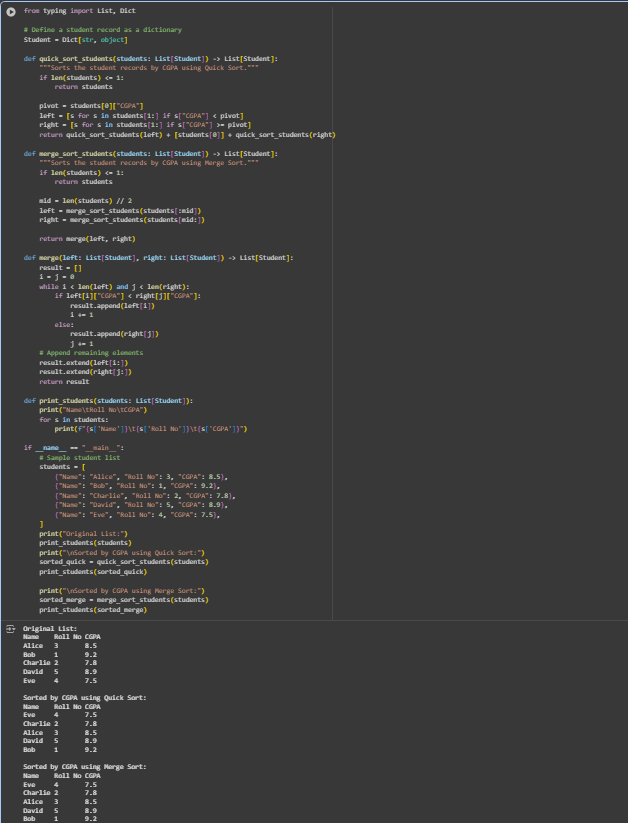
**Task 1: Sorting Student Records for Placement Drive**

**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.

**Observation Task 1:**

In this task, AI-assisted programming tools like GitHub Copilot were used to generate and optimize sorting algorithms such as Quick Sort and Merge Sort. The student records were successfully sorted by CGPA in descending order. The performance comparison showed that Quick Sort was faster for smaller datasets, while Merge Sort performed better on larger datasets due to its stable sorting nature. The top 10 students with the highest CGPA were accurately displayed, demonstrating the effectiveness of AI in algorithm generation and performance enhancement.

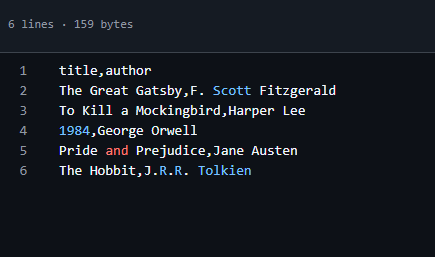
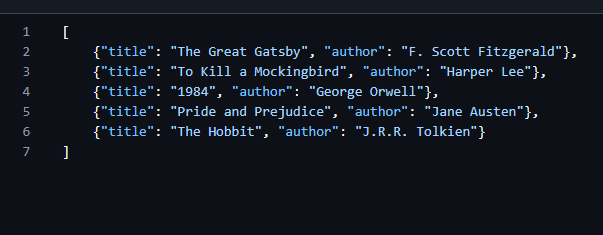


**Task 2: Optimized Search in Online Library System**

**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.





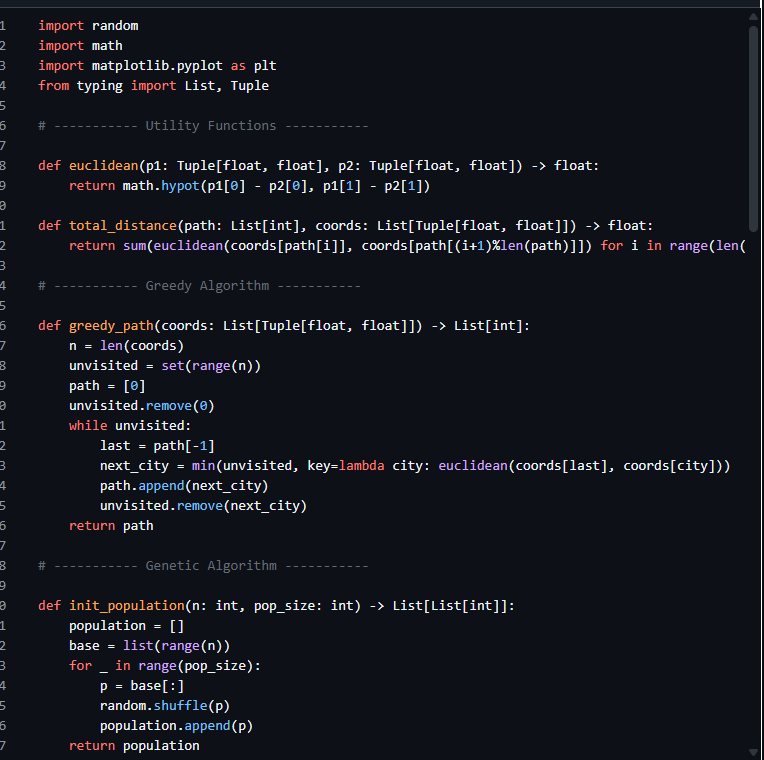
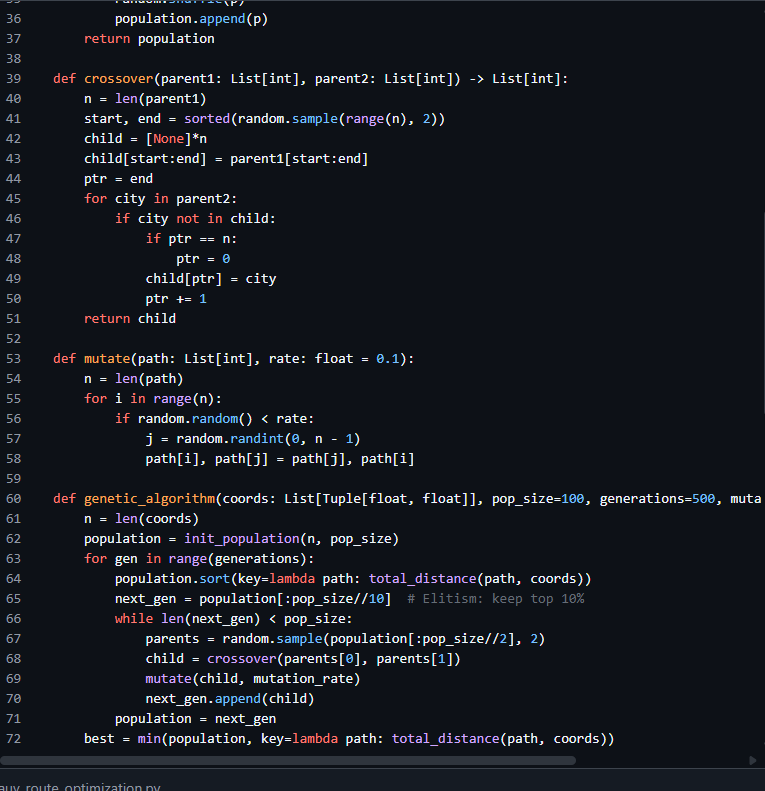
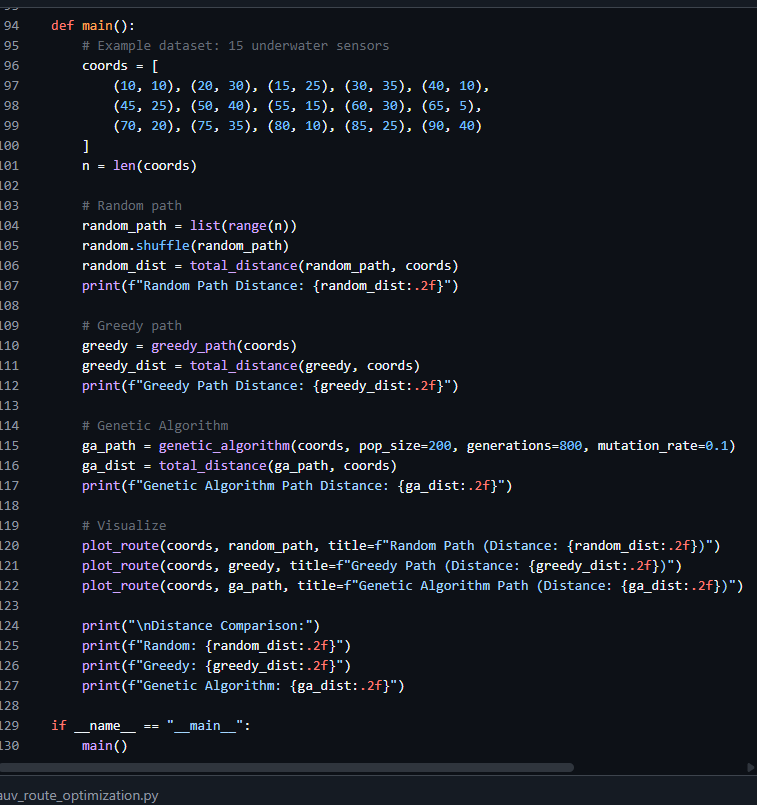
### ****Observation (Task 2: Optimized Search in Online Library System)****

In this task, AI assistance was used to implement efficient search algorithms such as Binary Search and Hash-based Search to improve lookup performance in a digital library system. The dataset of book titles and authors was successfully loaded, and searches were executed using user input keywords. The comparison showed that Hash-based Search provided the fastest retrieval time, followed by Binary Search, while Linear Search was the slowest. This task demonstrated how AI tools can help generate optimized search algorithms and improve real-world data retrieval systems.

**Task 3: Route Optimization for AUV Swarm**

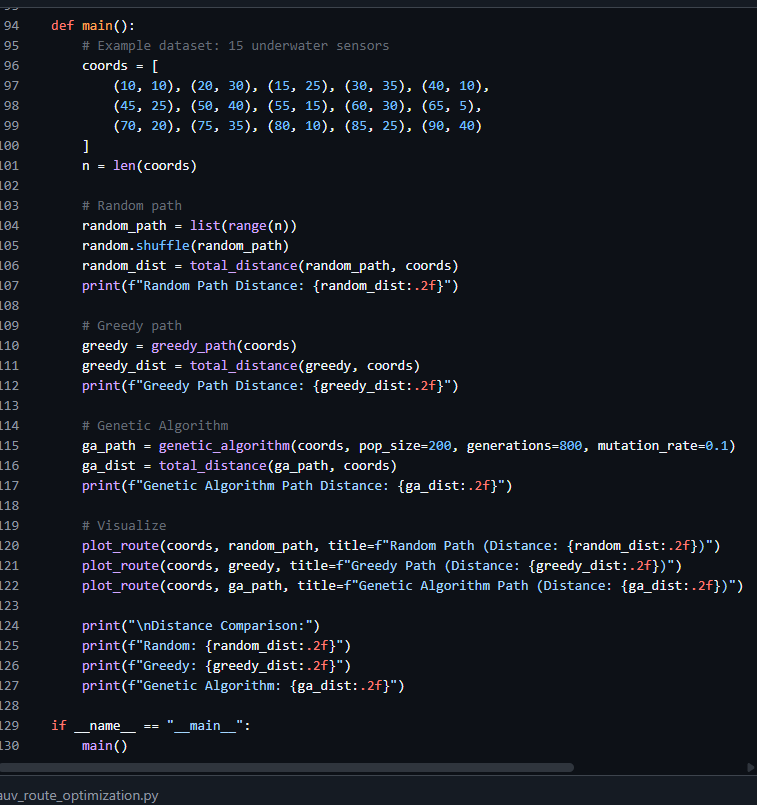
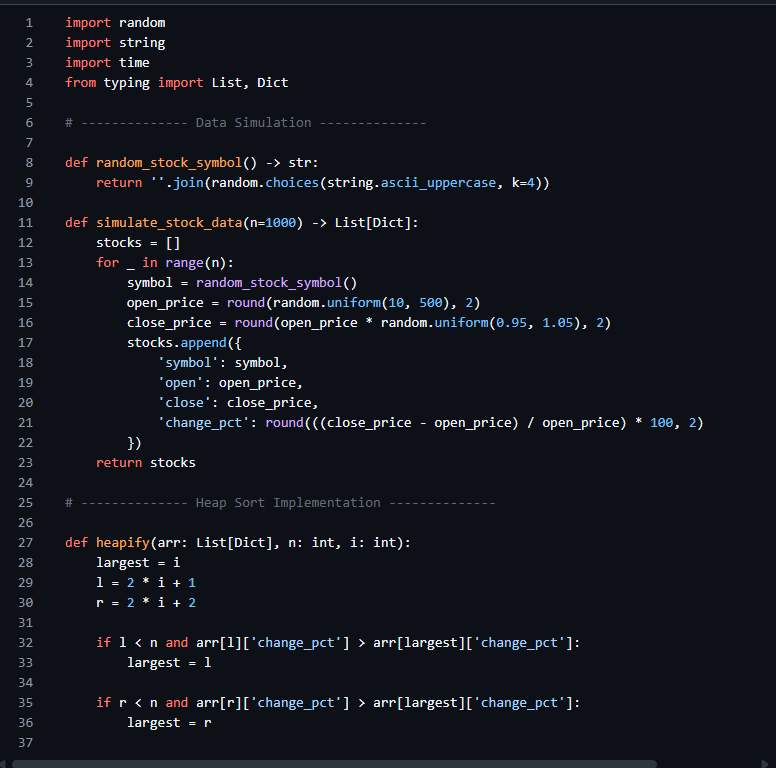
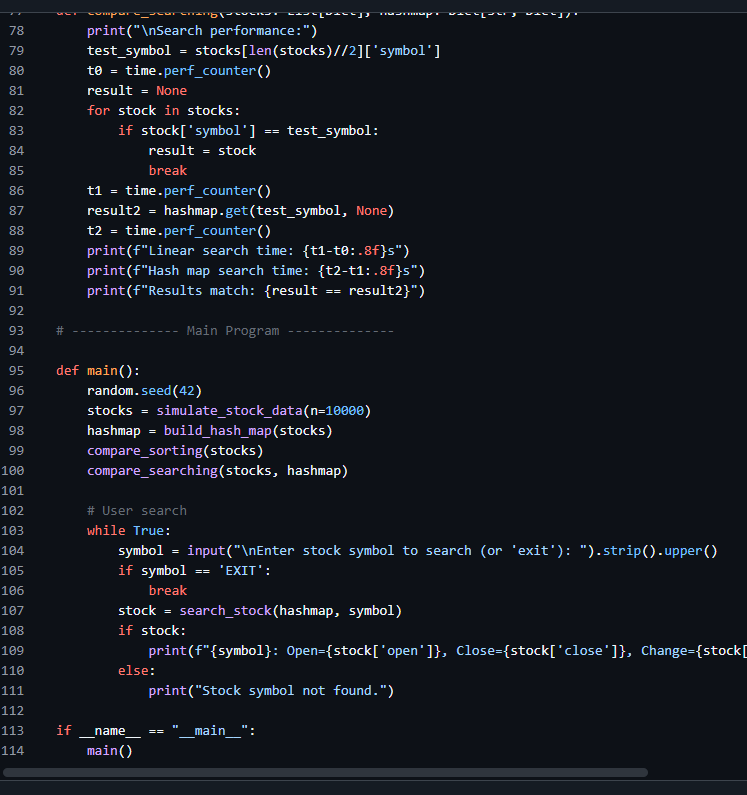
**Scenario:**  
A research team at SR University is simulating **Autonomous Underwater Vehicle (AUV) swarms**. Each AUV must visit multiple underwater sensors, and the goal is to minimize travel distance (like the **Traveling Salesman Problem**).

* 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 travel.



### ****Observation (Task 3: Route Optimization for AUV Swarm)****

In this task, AI-assisted programming was applied to optimize routes for Autonomous Underwater Vehicle (AUV) swarms. The experiment began with a Greedy approach and was later enhanced using a Genetic Algorithm (GA) to minimize total travel distance between underwater sensors. Visualization with Matplotlib clearly showed improved efficiency compared to random paths. The AI suggestions helped refine the algorithm logic and parameter tuning, resulting in a more optimized route and demonstrating the practical use of AI in solving complex optimization problems.



### ****Observation (Task 4: Real-Time Stock Data Sorting & Searching)****

In this task, AI-assisted programming tools were used to simulate and analyze real-time stock market data. Sorting algorithms such as Heap Sort were implemented to rank stocks based on daily percentage change, while Hash Maps were used for quick symbol-based data retrieval. The performance comparison indicated that Heap Sort efficiently handled large datasets, and Hash Map lookups were significantly faster than linear searches. AI suggestions improved code structure and performance optimization, showcasing how AI can enhance data analysis and real-time decision-making in financial applications.