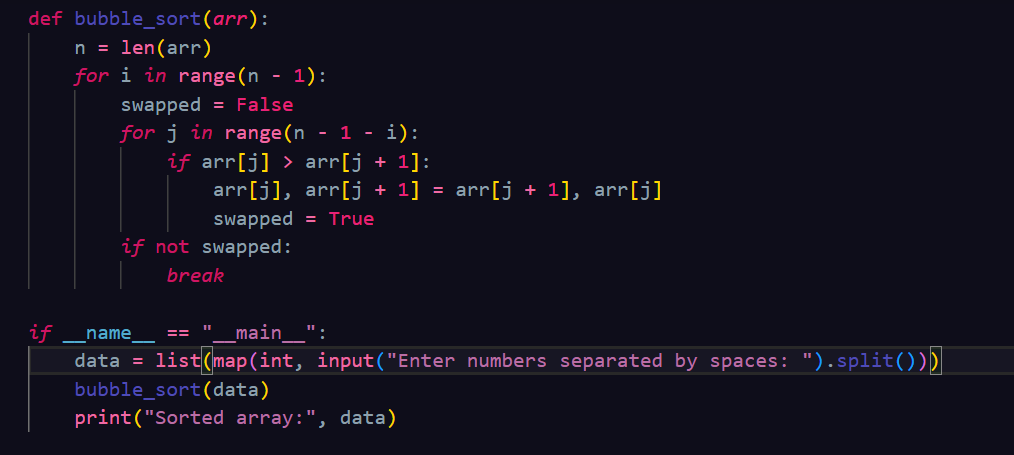
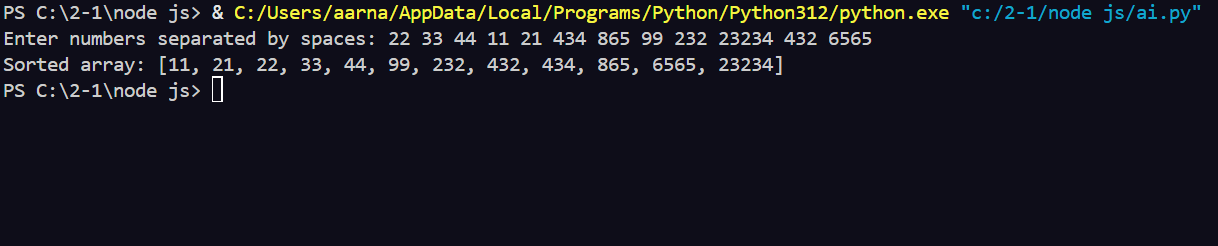
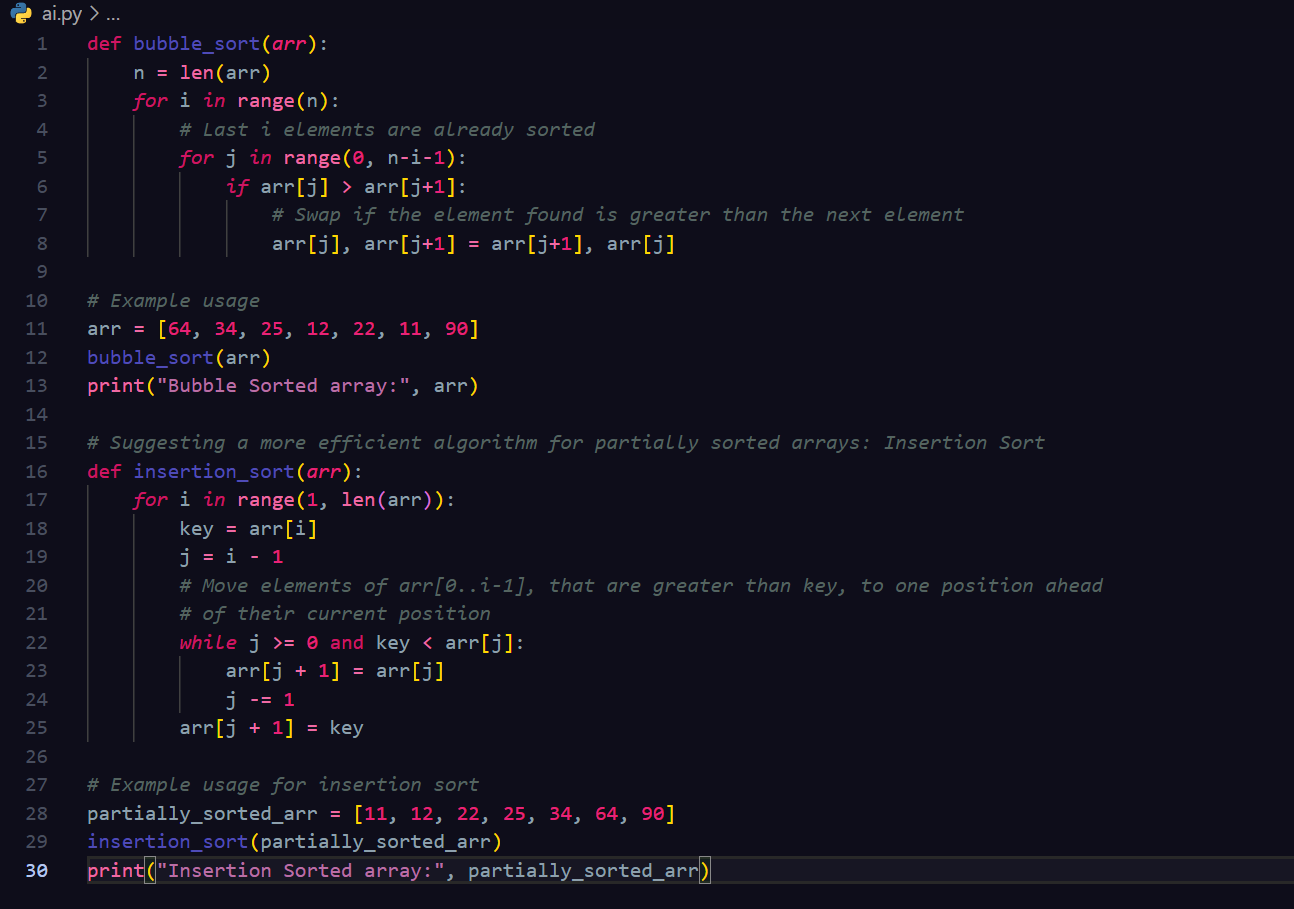
**AI ASSISTED CODING   
12.4**

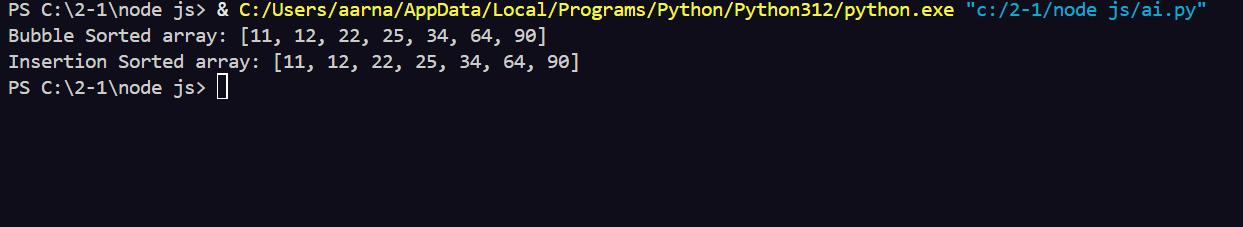
**TASK #1  
  
Prompt:  
  
write a python code to implement bubble sort such that the bubble sort should be implemented manually and there should be key comments explaining key logic ( like swapping , passes and termination ) and also provide time complexity analysis  
  
Code:**

****

**Output:  
  
Task #2  
  
Prompt:**

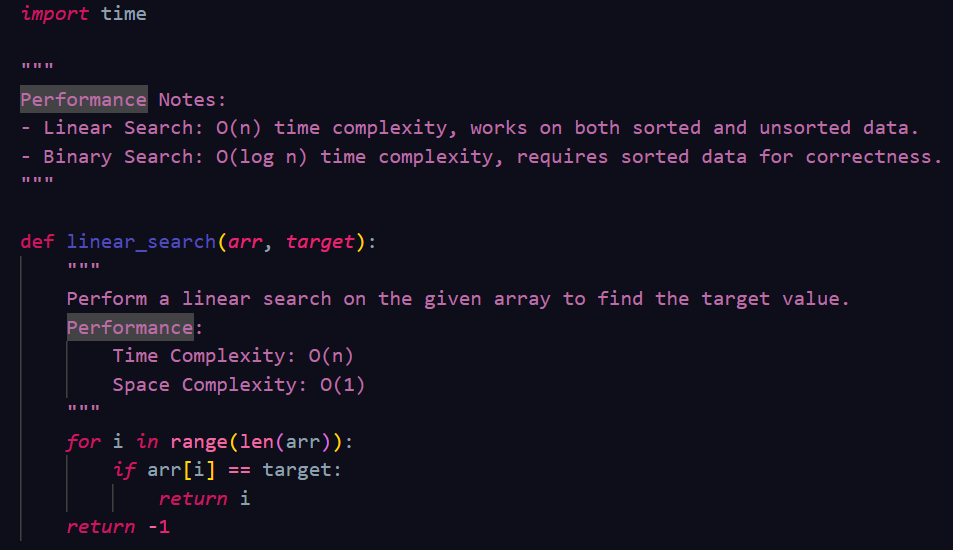
**write a python code to implement bubble sort and then suggest a more efficient algorithm for partially sorted arrays ( insertion sort ) also provide both codes and explain why insertion sort is more efficient for partially sorted data**

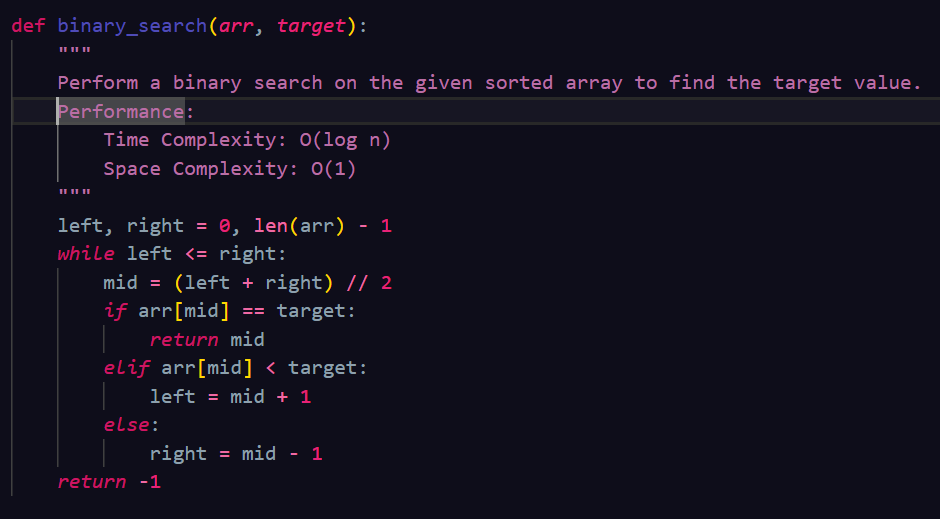
**Code:  
  
**

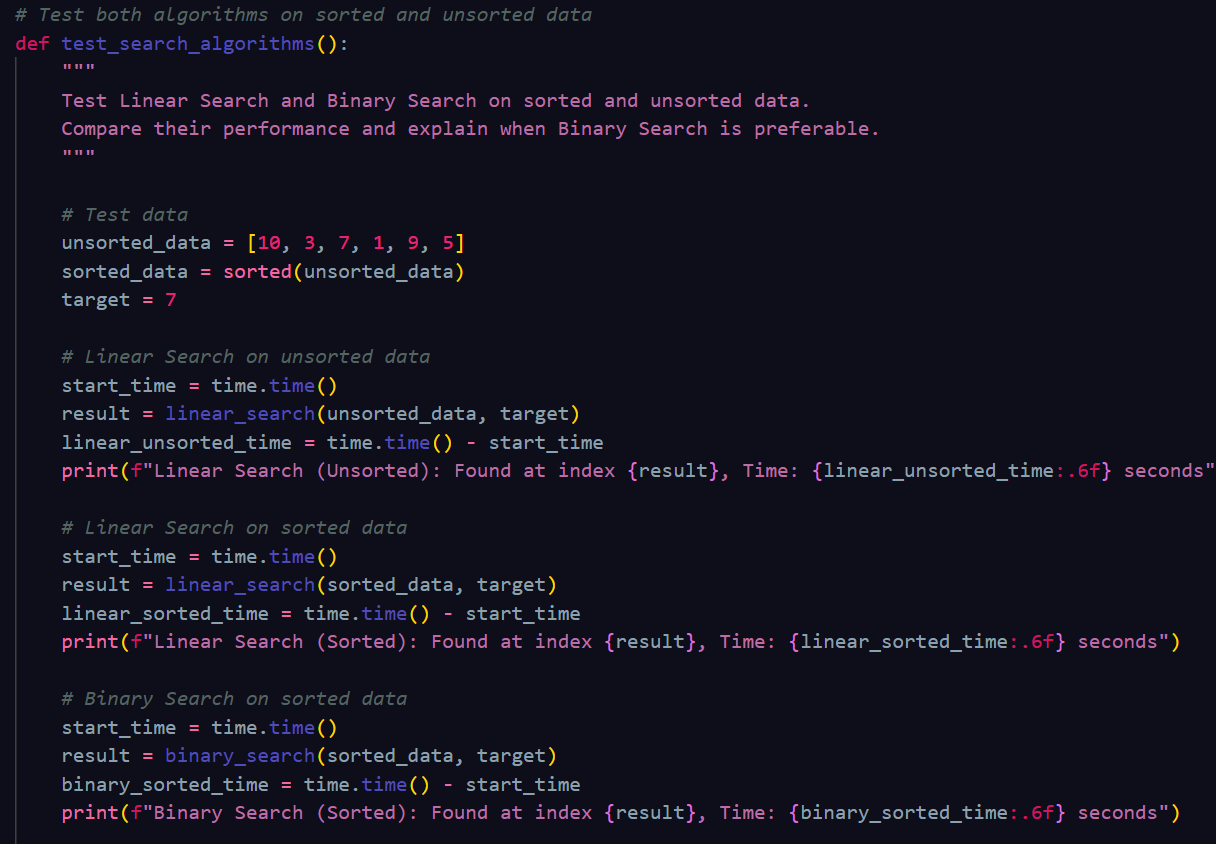
**Output:  
  
**

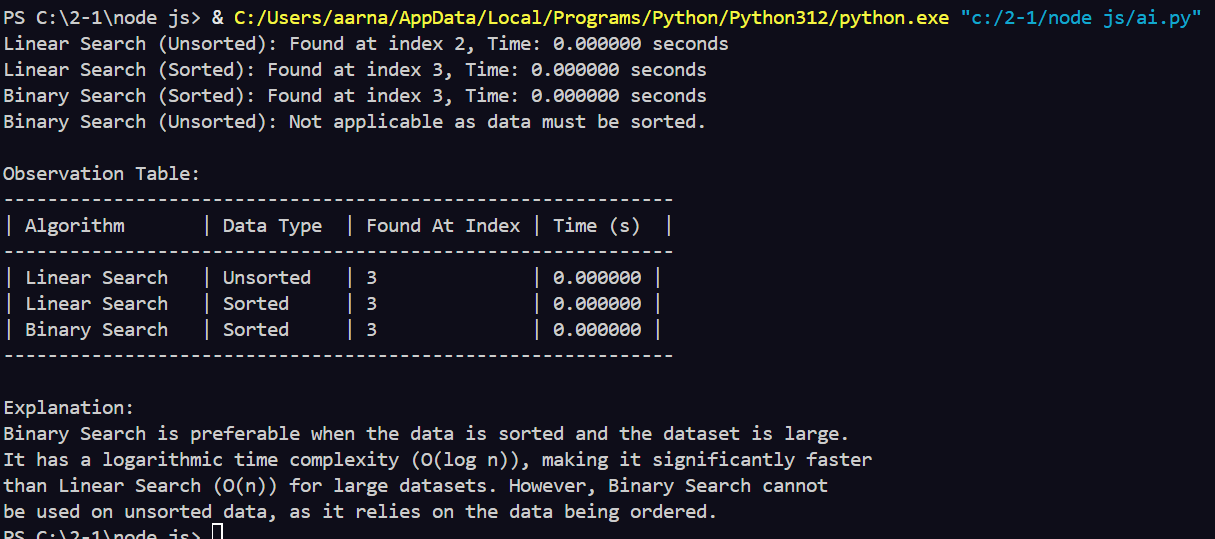
**Task #3  
  
Prompt:**

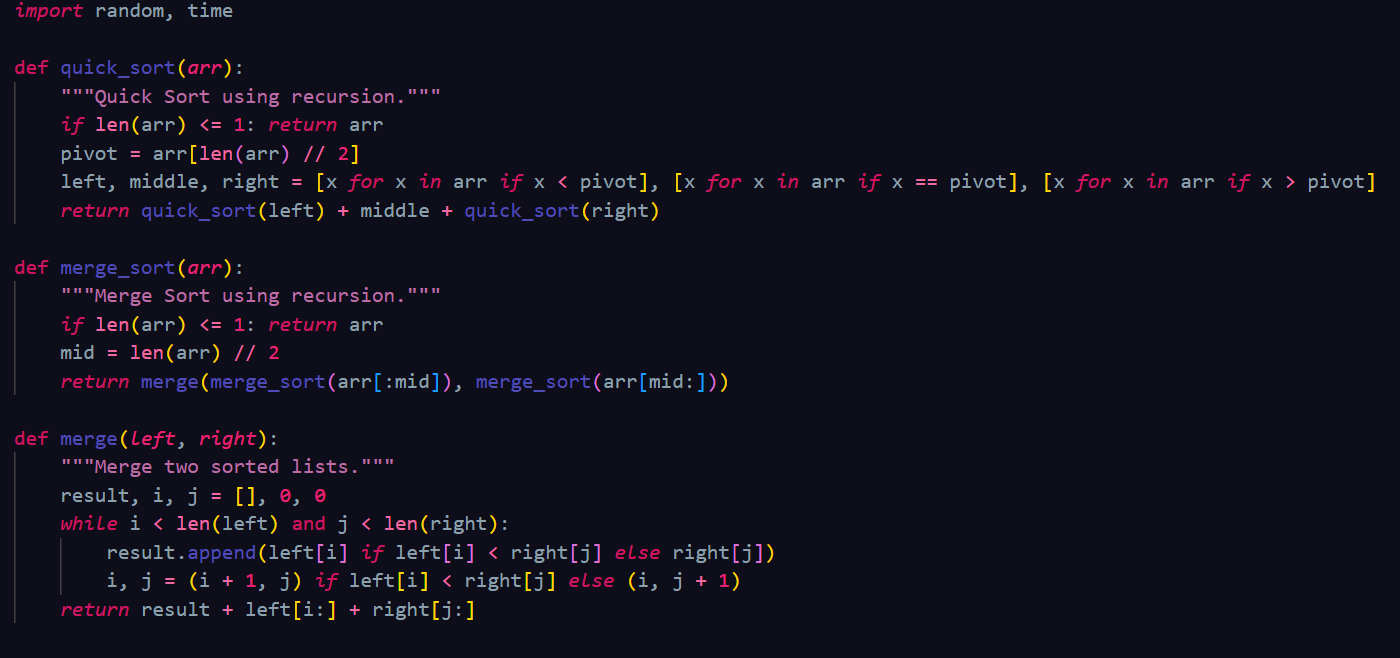
**write a python code to implement both linear search and binary search such that there should be ai generated docstrings and performance notes also test both algorithms on sorted and unsorted data and explain when binary search is preferable also provide a student observation table comparing performance if linear and binary search.**

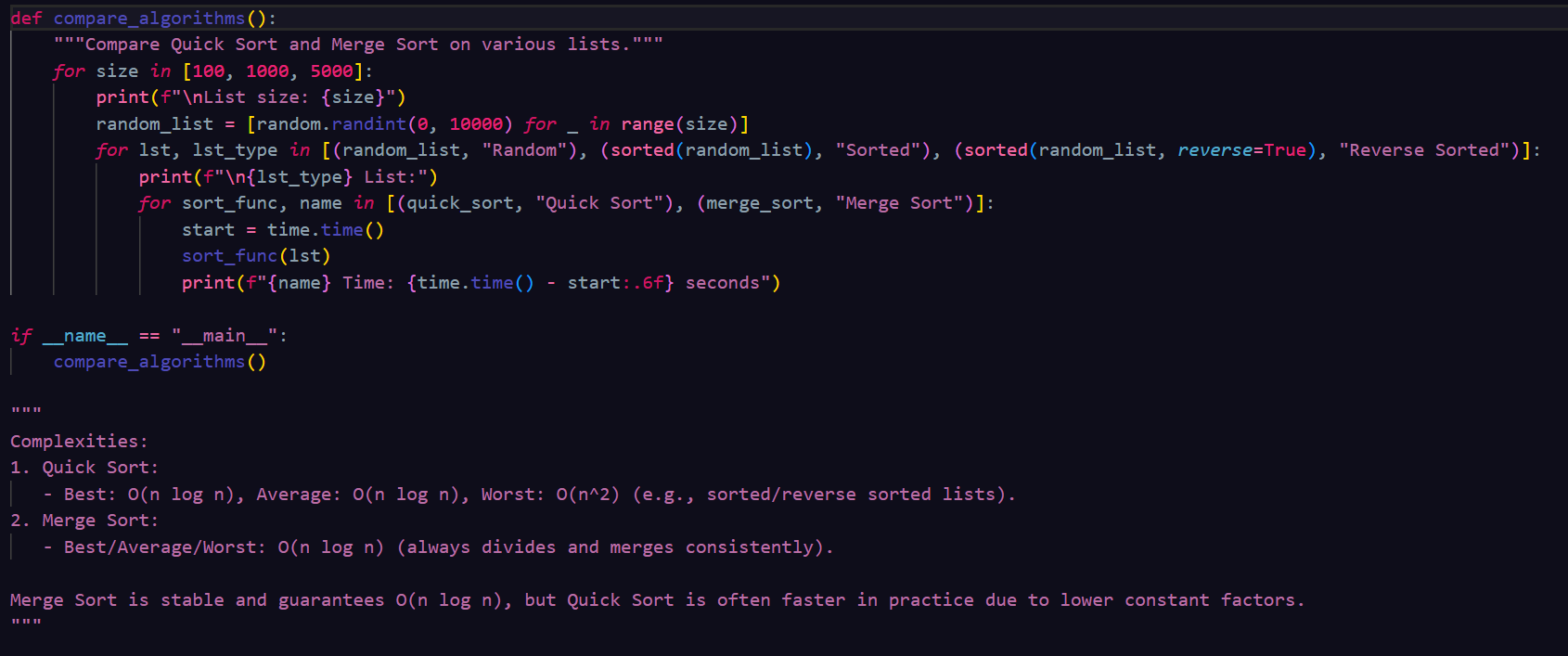
**Code:  
  
**

****

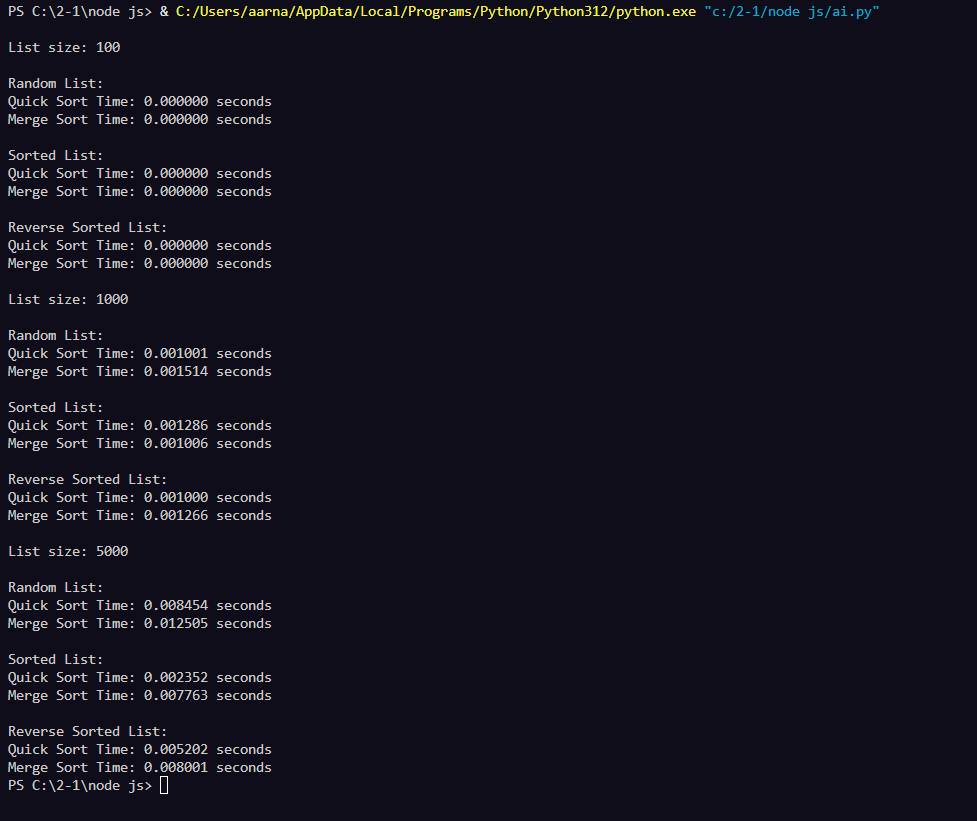
**  
  
  
Output:**

****

**Task #4  
  
Prompt:  
write a python code to implement quick sort and merge sort using recursion such that ai should complete the missing logic and add docstrings also compare both algorithms on random sorted and reverse sorted lists and provide explanation of average best and worst case complexities  
  
Code:  
  
**

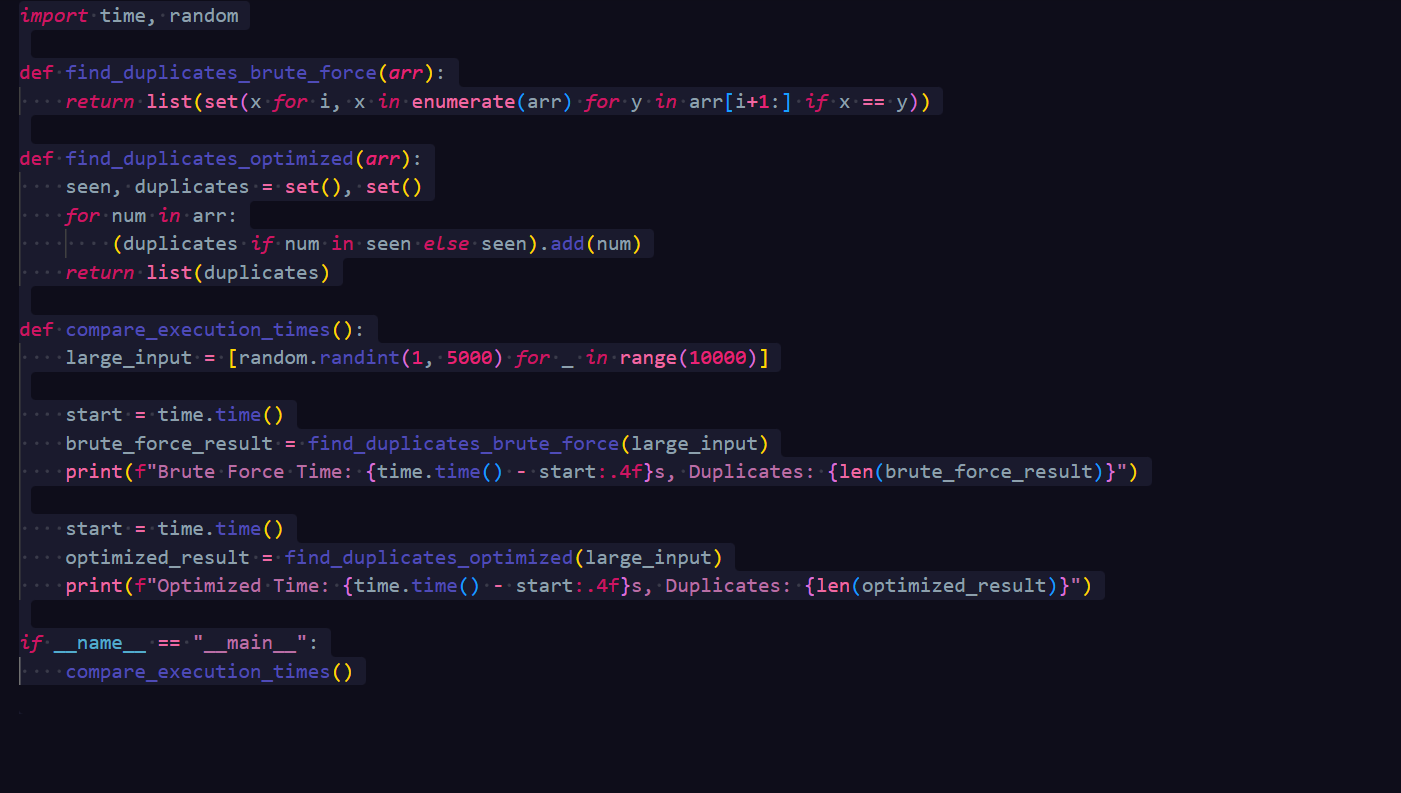
****

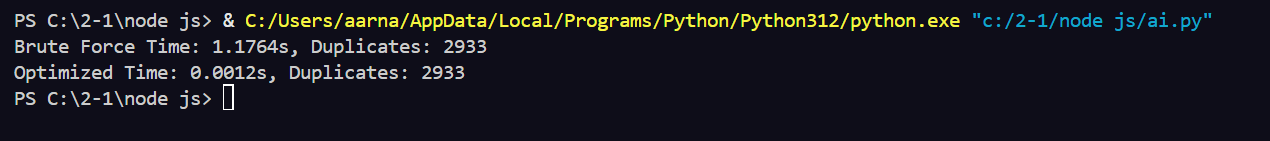
**Output:**

****

**Task #5  
  
Prompt:**

**write a python code to implement a brute force duplicate finder algorithm ( o(n^2) ) and then optimize it using sets or dictionaries ( o(n) ) also compare execution times with large input sizes and explain how the complexity was improved.**

**Code:  
  
**

**Output:  
  
**