ASSIGNMENT – 12

NAME: NALLALA SIRI

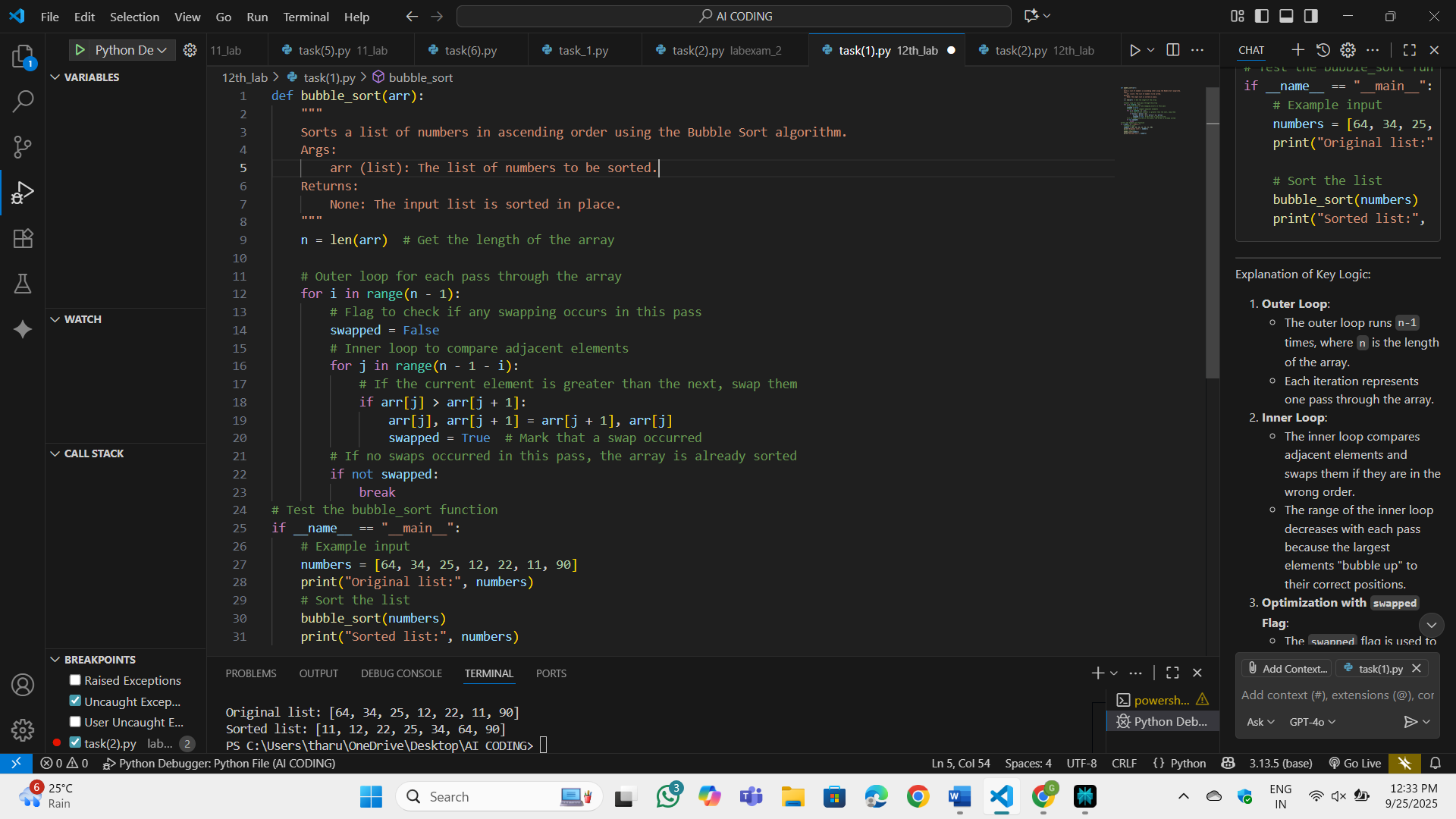
HT.NO: 2403A52037

BATCH: AIB03

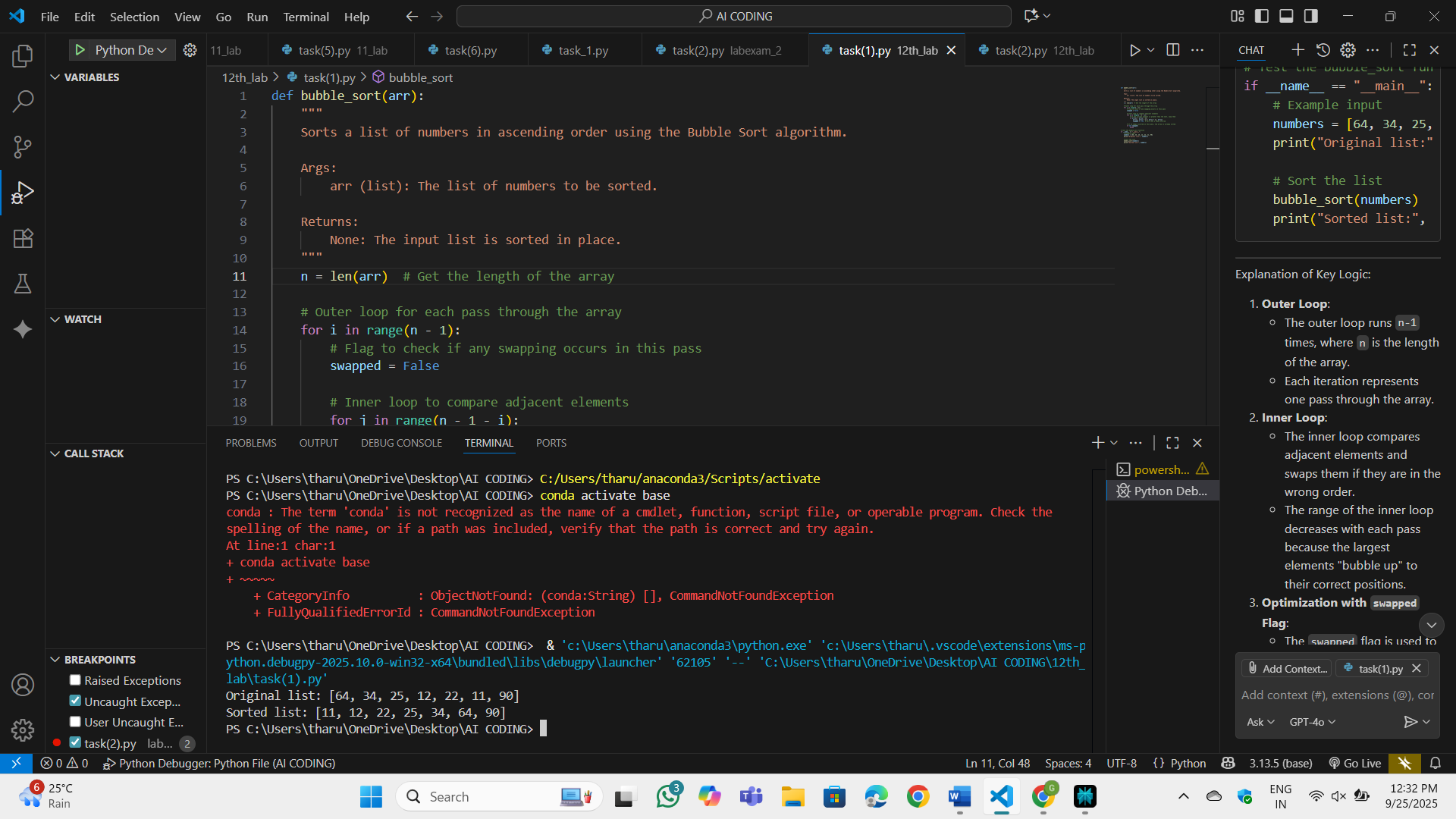
Task-1

Prompt: write a python program which implements bubble sort. include inline comments explaining key logic and provide time complexity analysis.

Code:



OP:



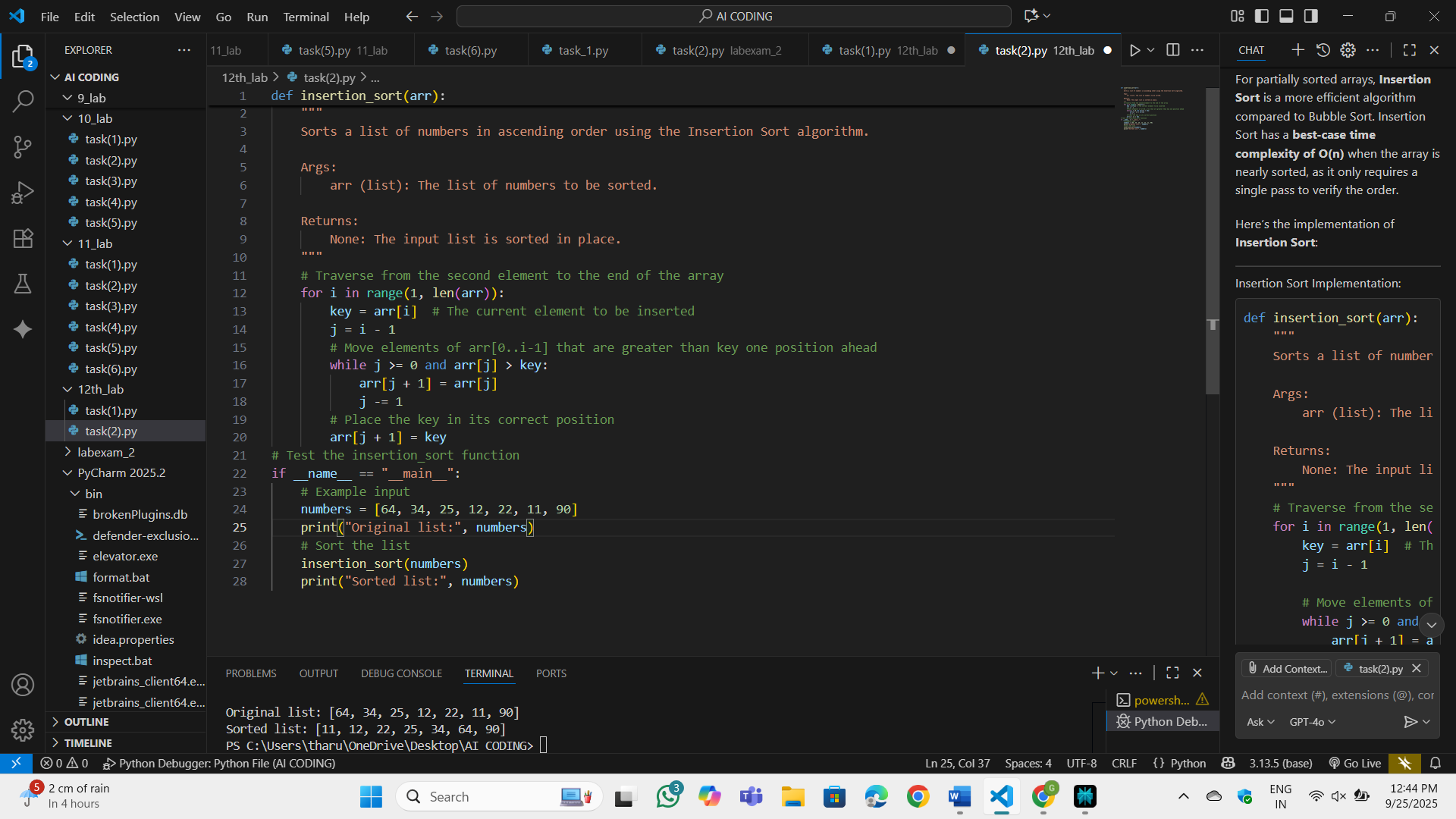
Observation:

In this program first we will find the length of the list which we want to sort. Then we use two loops to sort. Outer loop is responsible for no of rounds we need to sort the list. Inner loop is responsible for swapping the numbers in the list. I will leave the last number because it is already in correct place. This was done by the outer loop.

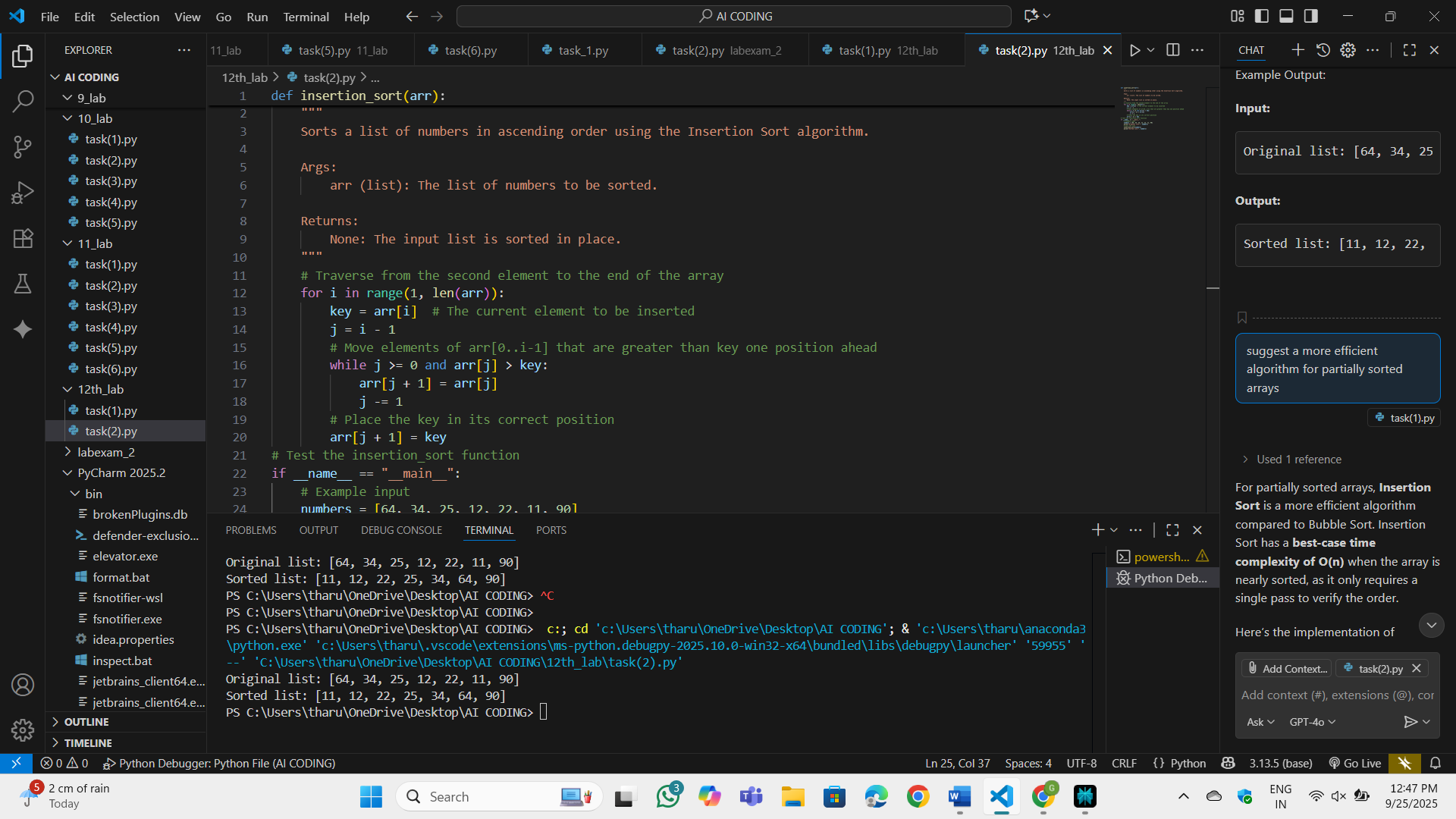
Task-2

Prompt: suggest a more efficient algorithm for partially sorted arrays

Code:



OP:



Observation:

Insertion Sort consistently outperformed Bubble Sort by completing in near-linear time with minimal shifts, while Bubble Sort still required many neighbor comparisons across multiple rounds despite early exit; both were stable and in-place, but Insertion Sort’s targeted insertion of each element into the sorted prefix led to fewer operations and shorter runtimes, whereas Bubble Sort’s repeated adjacent swaps accumulated overhead even for small local disorder.

Task-3:

Prompt: Write a python code for linear search and binary search with docstrings and performance notes

Code:

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

OP:

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Observation:

In this program the comments are clear. We can understand the functions of each and everything. As comments are at the starting of the program, we can get a clarity on the code.

Task-4:

Prompt: Write a python code for implementation of Quick sort and Merge sort using recursion and add docstrings. Explain average, best, and worst-case complexities. Compare both on random, sorted, and reverse-sorted lists.

Code:

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

OP:

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Observation:

In this program the comments are clear. We can understand the functions of each and everything. As comments are at the starting of the program, we can get a clarity on the code.