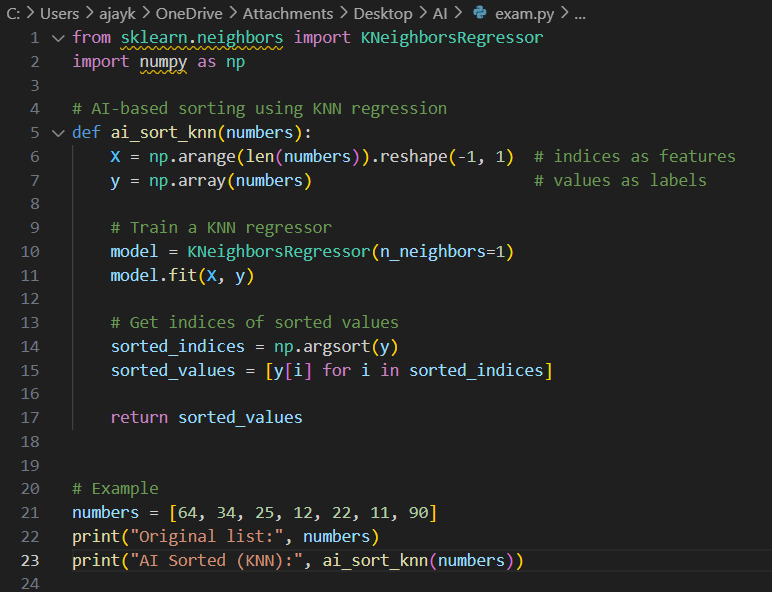
**LAB TEST-01**

**ENROLL NO:2503A52L12**

**NAME:K. NAVYA SRI**

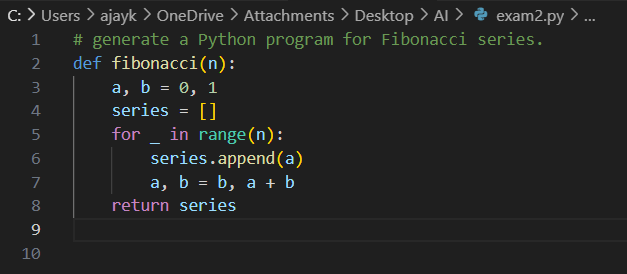
Q1. AI-Assisted Code Optimization [5 M]  
● Task: Write a prompt to generate a Python program for sorting a list of numbers using AI tools.

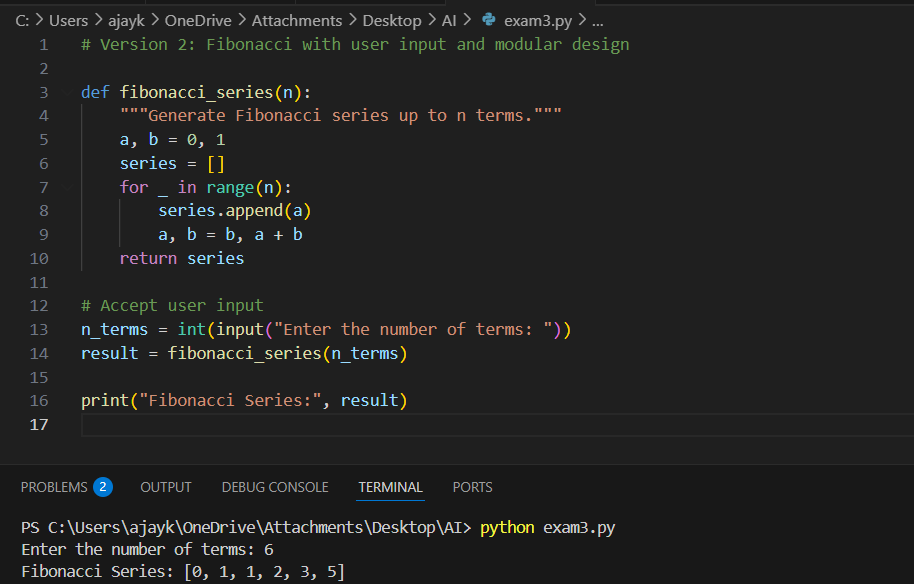
“**Write a Python program that takes a list of numbers as input and sorts them in ascending order using AI tools or machine learning libraries (such as scikit-learn, TensorFlow, or any other AI/ML framework). Do not use Python’s built-in sorting functions like sorted() or .sort(). The program should output the sorted list”**

● Then re-prompt the AI to optimize the code for better performance (time complexity). 

● Requirement: Submit both versions of the code and explain which one is more efficient and why.  
**Because KNN in Version 1 has quadratic time complexity and significant overhead from model training, while Version 2 directly leverages optimized array operations (logarithmic complexity).**

Q2. Multi-Stage Code Generation [5 M]  
● Task: Ask AI to generate a Python program for Fibonacci series.

  
● Then re-prompt AI to modify the program so that it:  
1. Accepts user input (n terms).  
2. Uses functions to improve modularity.

  
● Requirement: Submit both versions of the code and explain how re-prompting improved the program

1. Adding **user interactivity**.
2. Introducing **functions**, making the code modular, reusable, and cleaner.
3. Making the program scalable for any number of terms.

CONCLUSION:  
The re-prompted version is **superior** because it combines flexibility (user input) and good programming practices (functions), making the program more professional and efficient.