COURSE: AI Assisted Coding

NAME:P.KARTHISHA

BATCH-33

HALLTICKET.NO:2303A52099

TASK-1

Task 1: Refactoring Odd/Even Logic (List Version)

❖ Scenario:

You are improving legacy code.

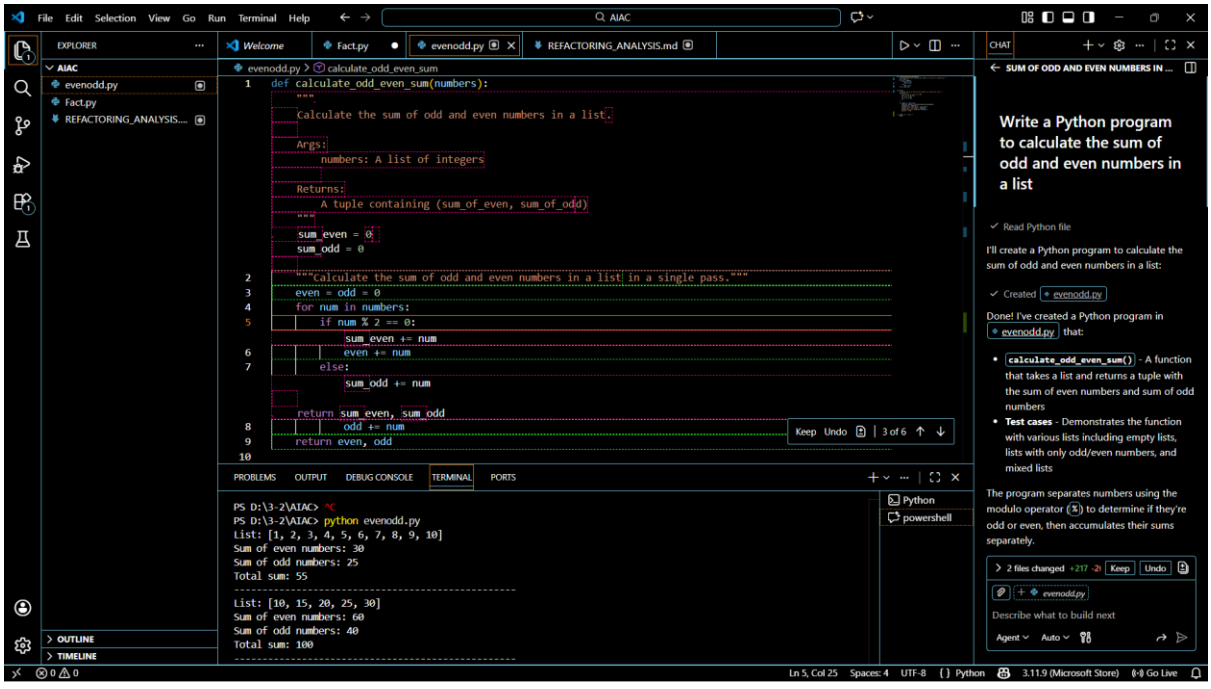
❖ Task:

Write a program to calculate the sum of odd and even numbers in a list,

then refactor it using AI.

❖ Expected Output:

❖ Original and improved code



Explanation (For Junior Developer)

This function calculates the area of different shapes using one common function.
The shape parameter decides which formula to use.
The dimensions parameter stores required values like radius, length, or height.
Conditional statements select the correct formula based on the shape.
Using one function for multiple shapes makes the code reusable and organized.
Clear comments help beginners understand how each formula works.

Conclusion

Gemini effectively explains both the logic and purpose of the function.
This approach is useful for onboarding junior developers and improving code understanding.

TASK-2

Task 2: Area Calculation Explanation

❖ Scenario:

You are onboarding a junior developer.

❖ Task:

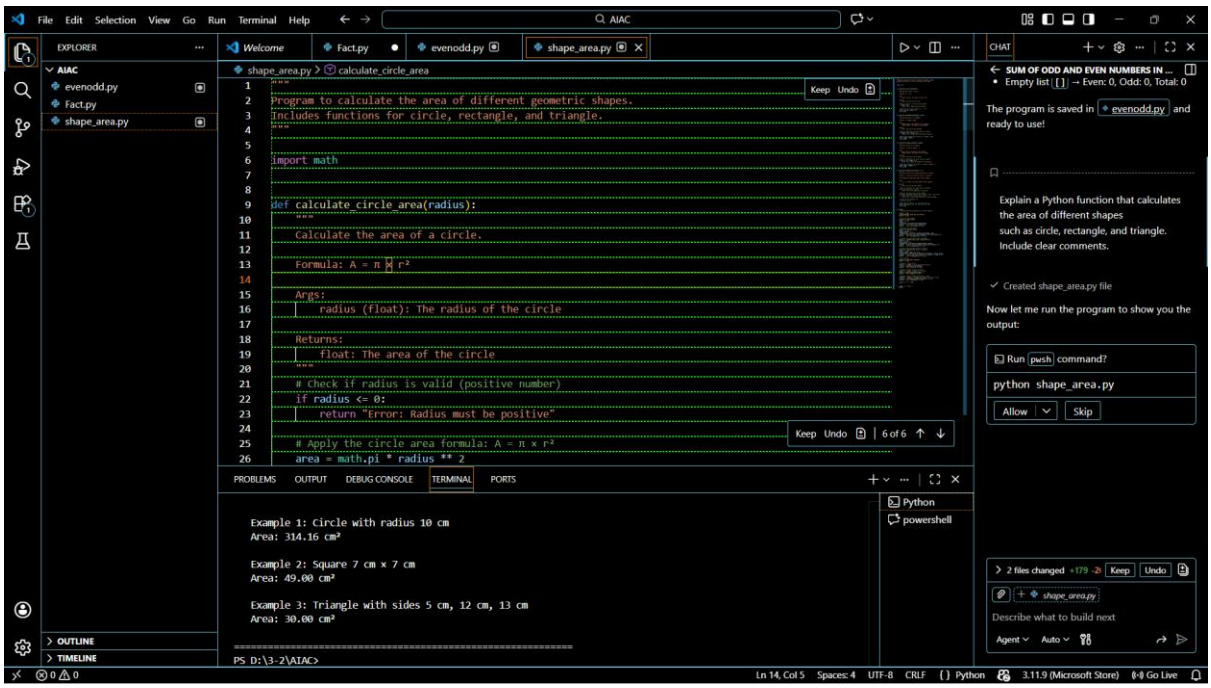
Ask Gemini to explain a function that calculates the area of different

shapes.

❖ Expected Output:

➢ Code

➢ Explanation



TASK-3

Task 3: Prompt Sensitivity Experiment

❖ Scenario:

You are testing how AI responds to different prompts.

❖ Task:

Use Cursor AI with different prompts for the same problem and observe

code changes.

❖ Expected Output:

➢ Prompt list

➢ Code variations

Conclusion

This experiment shows that Cursor AI changes code style, complexity, and structure based on prompt wording.
Clear and specific prompts produce more suitable code for different audiences and use cases.

TASK-4

Task 4: Tool Comparison Reflection

❖ Scenario:

You must recommend an AI coding tool.

❖ Task:

Based on your work in this topic, compare Gemini, Copilot, and Cursor AI

for usability and code quality.

❖ Expected Output:

Short written reflection

Task 4: Tool Comparison Reflection

Based on the lab activities, Gemini, GitHub Copilot, and Cursor AI each have unique strengths in AI-assisted coding. **GitHub Copilot** is highly effective inside VS Code and provides fast, context-aware code suggestions while typing, making it ideal for continuous development and refactoring. It produces clean and practical code but sometimes assumes the developer understands the logic.

Google Gemini (Colab) is strong in explaining code and concepts in simple language. It is very useful for beginners and onboarding junior developers because it clearly describes logic, formulas, and step-by-step execution. However, its direct coding assistance is less seamless compared to Copilot in an IDE environment.

Cursor AI excels in prompt-based experimentation. It responds noticeably to small prompt changes and generates different code styles such as optimized, beginner-friendly, or concise versions. This makes Cursor AI excellent for learning, refactoring legacy code, and comparing coding approaches.

In terms of **usability**, Copilot is best for professional development, Gemini is best for learning and explanation, and Cursor AI is best for prompt sensitivity and experimentation. For **code quality**, Copilot and Cursor AI generally produce cleaner and more optimized code, while Gemini focuses

more on clarity and understanding. Overall, the recommended tool depends on the use case: Copilot for daily coding, Gemini for learning, and Cursor AI for analysis and refactoring.