ASSIGNMENT -3.2

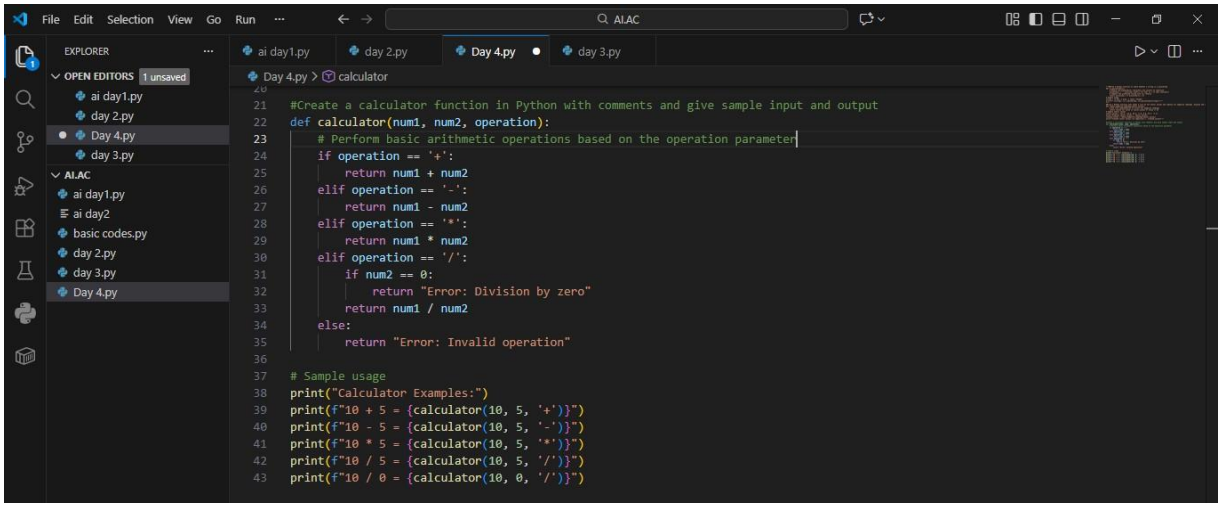
A.NagaKoushik

2403A51L22

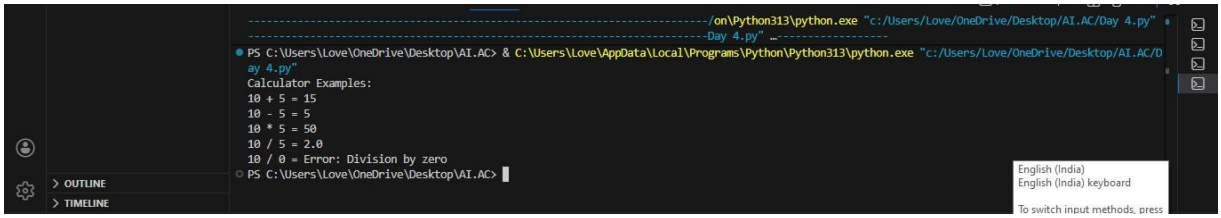
Batch:51

Task 1: Progressive Prompting – Calculator Design

PROMPT: Create a calculator function in Python with comments and give sample input and output.



OUTPUT:



EXPLANATION:

When we give only a function name, the AI generates very basic or incomplete code.

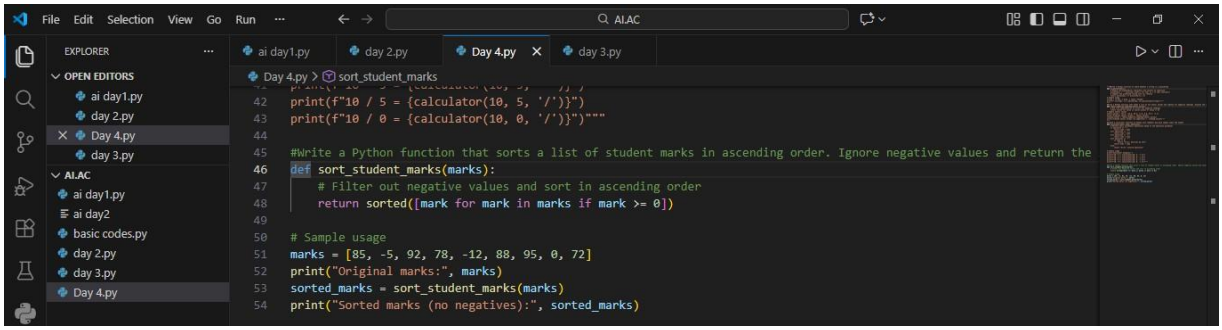
As we gradually add comments, requirements, and examples, the AI understands better and produces:

- Proper logic ,Error handling , Cleaner structure

This shows that well-defined prompts lead to better AI-generated programs.

Task 2: Refining Prompts – Sorting Student Marks

PROMPT: Write a Python function that sorts a list of student marks in ascending order. Ignore negative values and return the sorted list using efficient logic.



OUTPUT:

EXPLANATION:

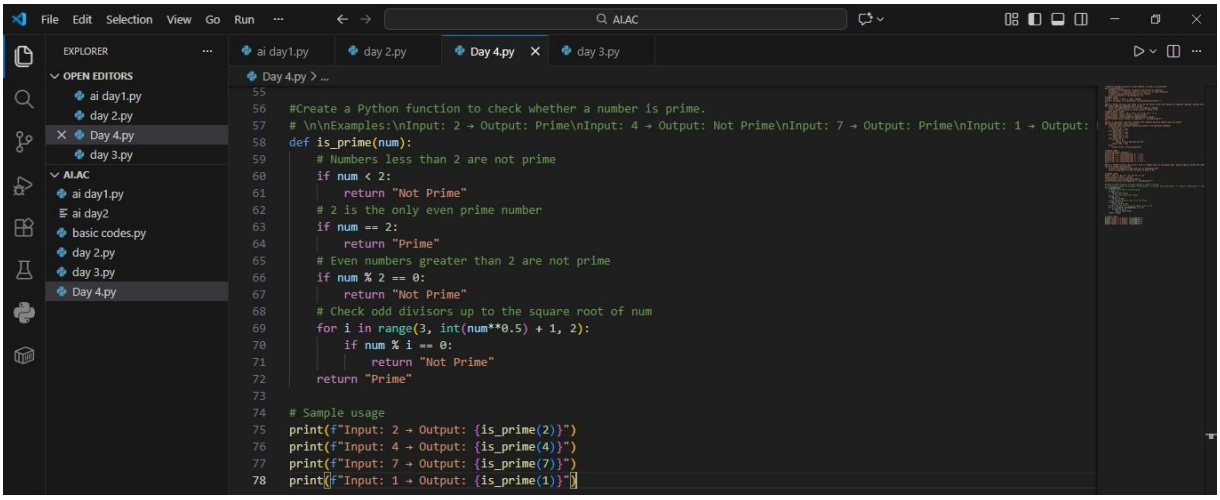
This task demonstrates how **vague prompts cause ambiguous results**. Initially, the AI may not know:

- Sorting order , Data constraints , Output format

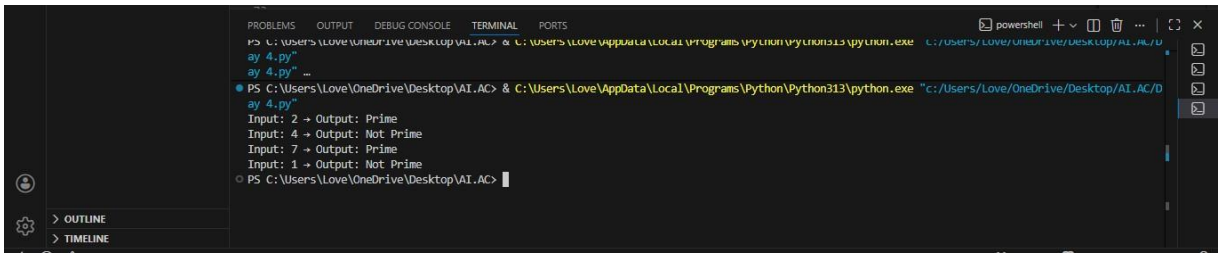
By refining the prompt, we guide the AI to generate **accurate and efficient sorting logic**.
This highlights the importance of **specific instructions in prompt engineering**.

Task 3: Few-Shot Prompting – Prime Number Validation

Prompt: Create a Python function to check whether a number is prime. Examples:\n Input: 2 → Output: Prime\n Input: 4 → Output: Not Prime\n Input: 7 . Use these examples to design the logic



OUTPUT:



EXPLANATION:

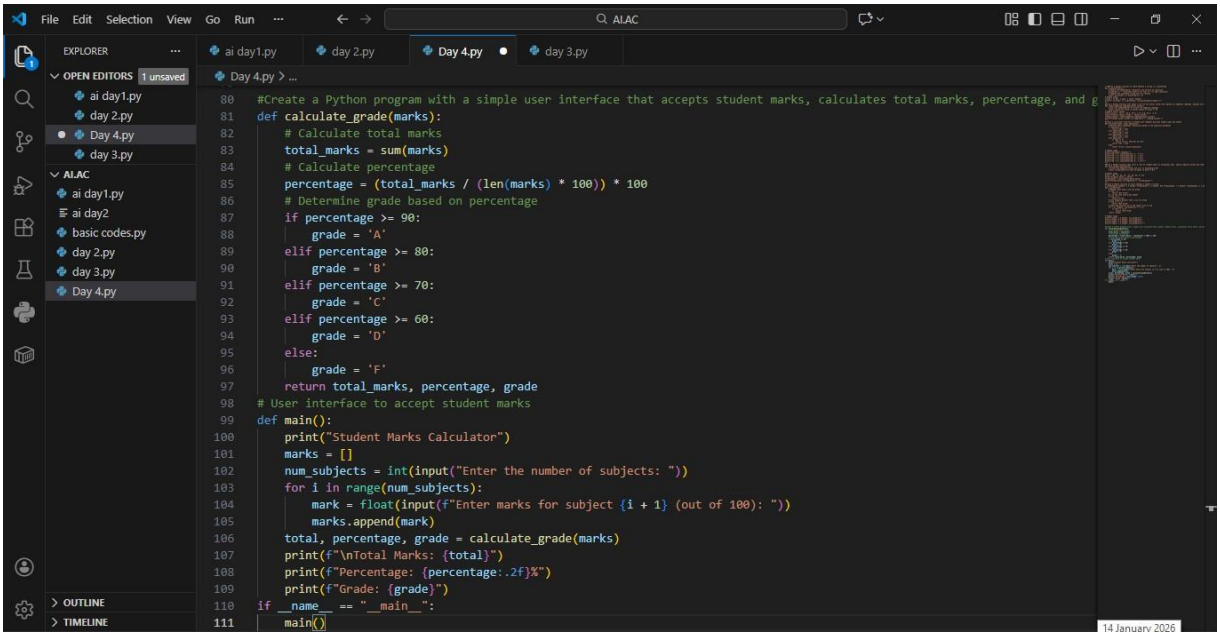
Few-shot prompting means providing **example inputs and outputs** along with the prompt. This helps the AI:

- Understand edge cases , Improve accuracy , Avoid logical mistakes

Compared to a simple prompt, few-shot prompting results in **more reliable prime-checking logic**.

Task 4: Prompt-Guided UI Design – Student Grading System

Prompt : Create a Python program with a simple user interface that accepts student marks, calculates total marks, percentage, and grade, and displays the result.



Output:

Explanation:

This task focuses on using prompts to guide program structure and user interaction.

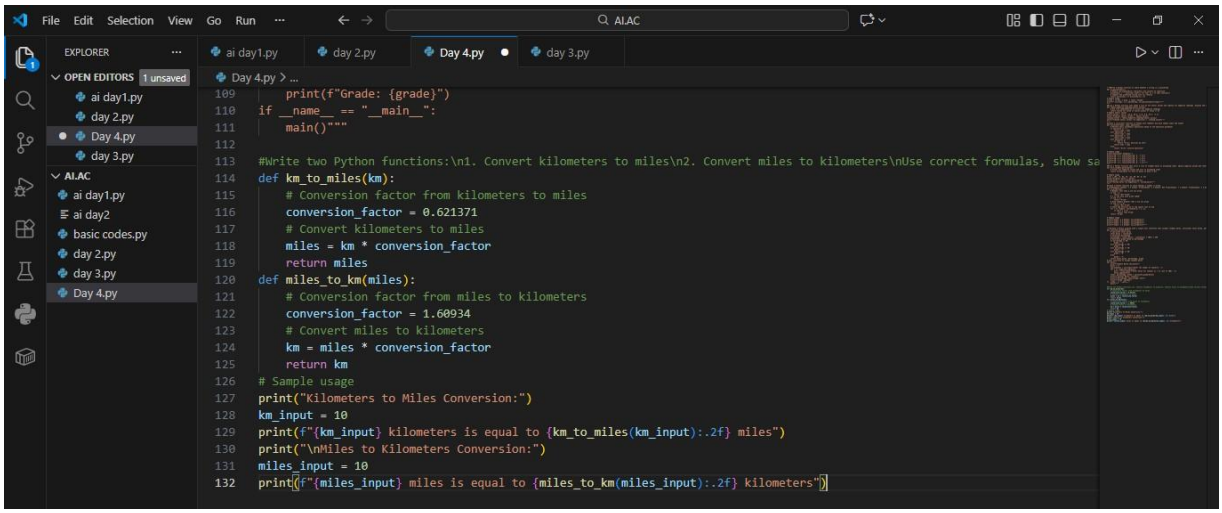
Instead of a graphical UI, a console-based UI is used for:

- Simplicity , Code compatibility , Clear user interaction

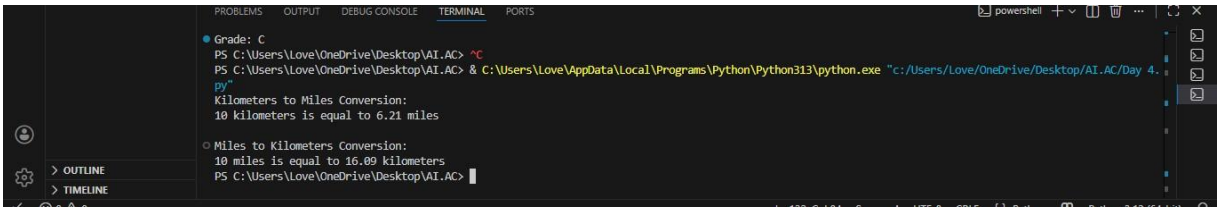
Task 5: Prompt Specificity – Unit Conversion Function

Prompt:

Write two Python functions:\n1. Convert kilometers to miles\n2. Convert miles to kilometers\nUse correct formulas, show sample input/output, and add comments explaining the logic



Output:



Explanation:

This task highlights how clear and specific prompts improve code accuracy.

A vague prompt may produce incomplete or incorrect conversions.

When formulas and requirements are clearly stated, the AI generates:

- Accurate calculations, Reusable functions, Well-documented code

This proves that **prompt specificity directly affects output quality**.