**AI Assisted Coding**

**Assignment – 3.2**

**Name: Kommu Madhupriya  
Batch:22**

**Hallticket:2303A51583**

**Task Description-1**

• Progressive Prompting for Calculator Design: Ask the AI to design a simple calculator

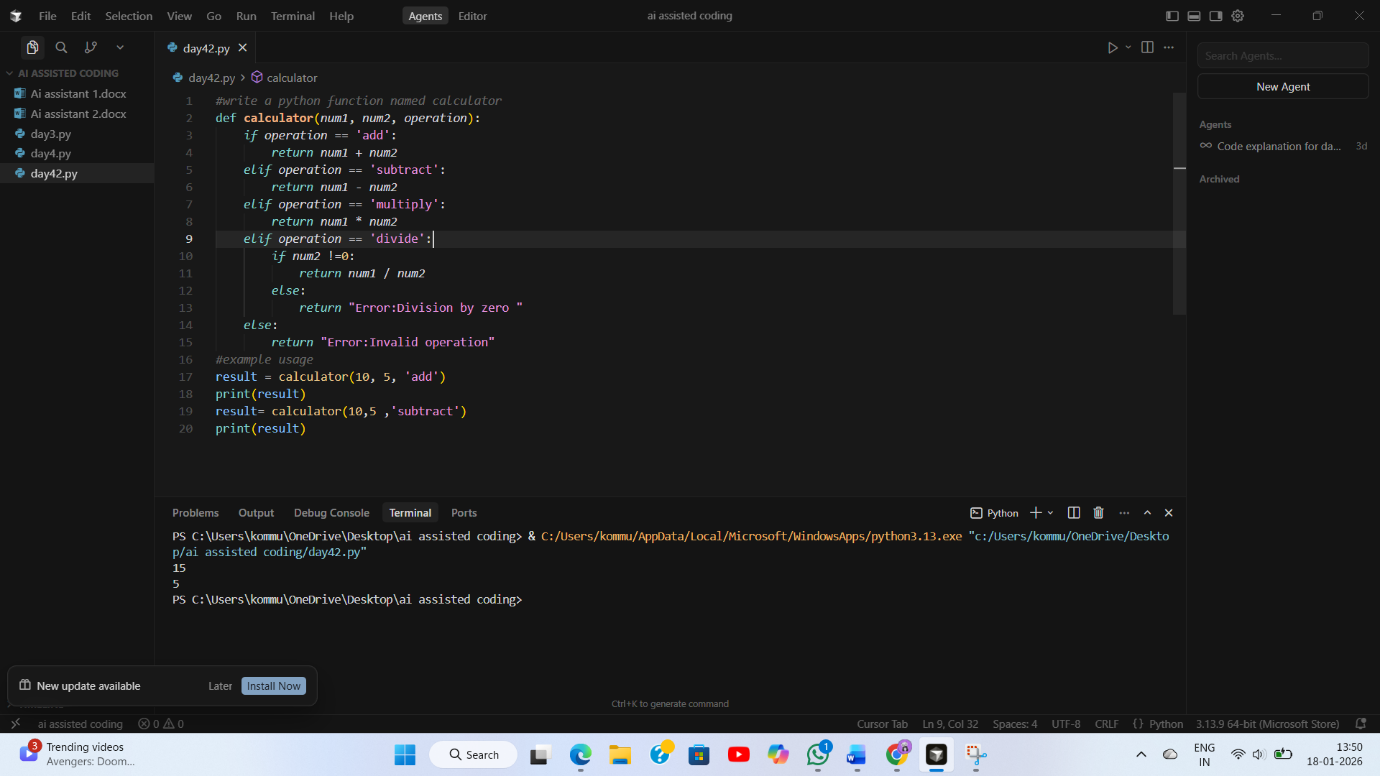
program by initially providing only the function name. Gradually enhance the prompt by

adding comments and usage examples.

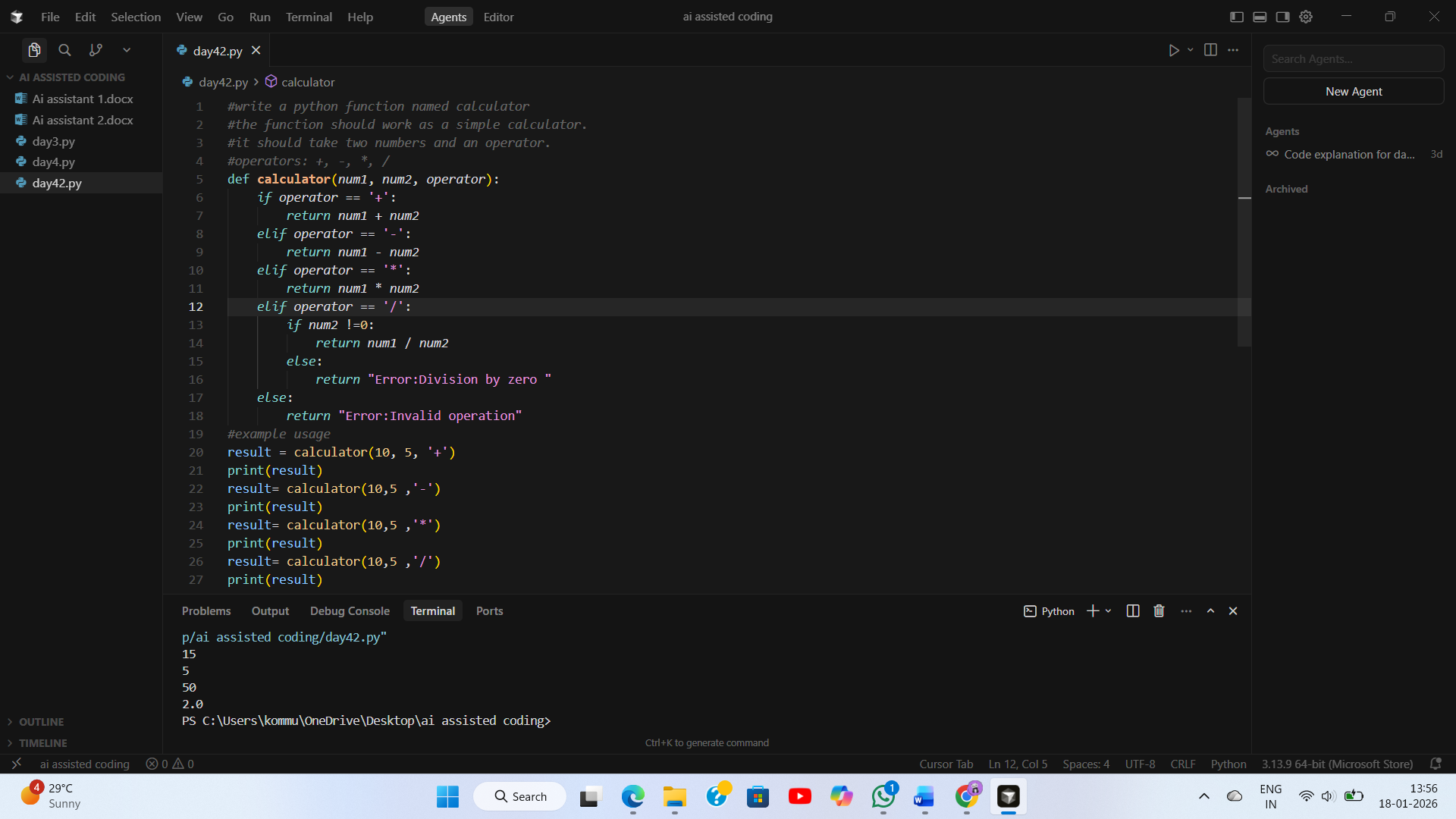
Expected Output-1

• Comparison showing improvement in AI-generated calculator logic and structure.

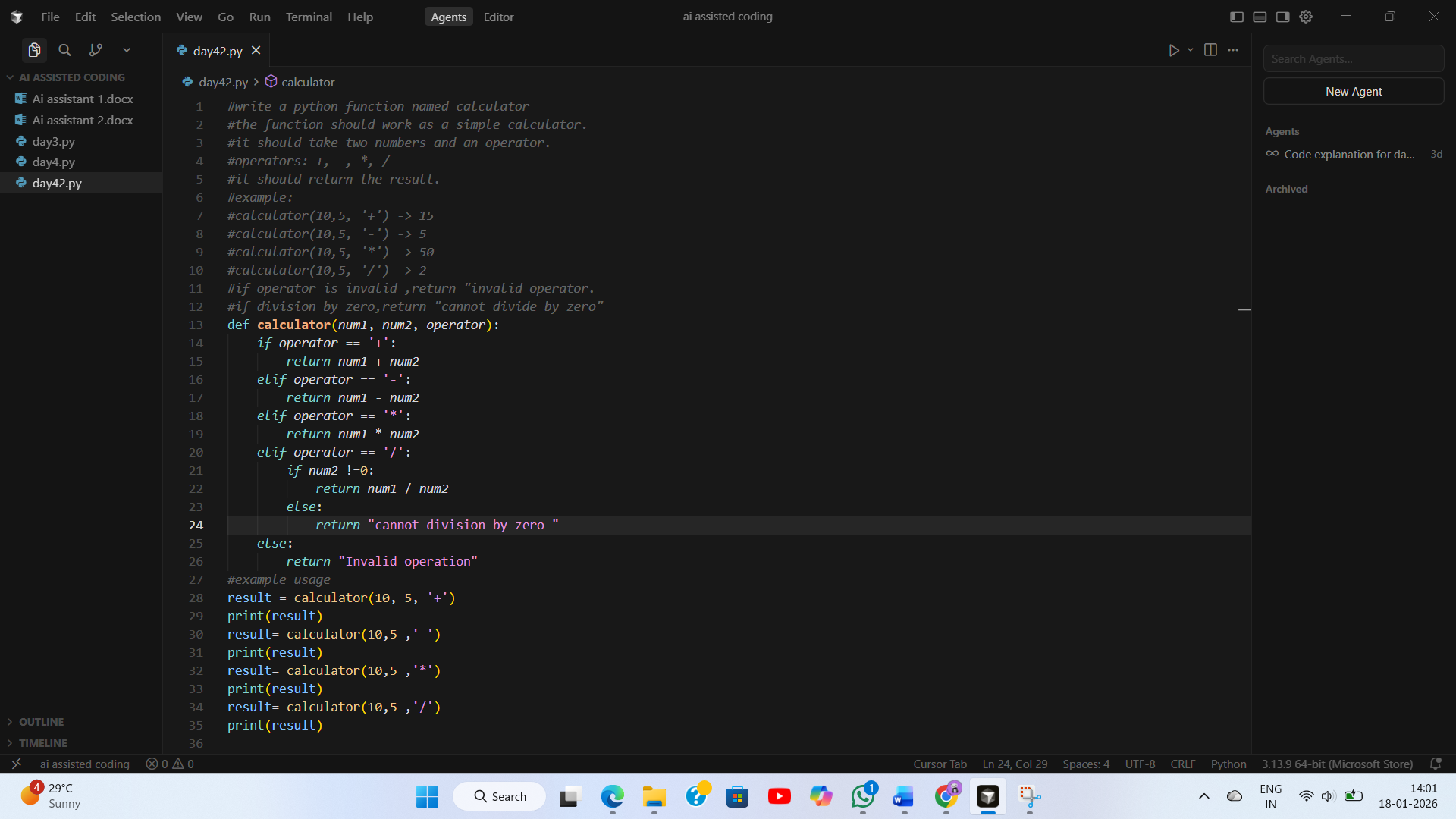
**Stage 1:**



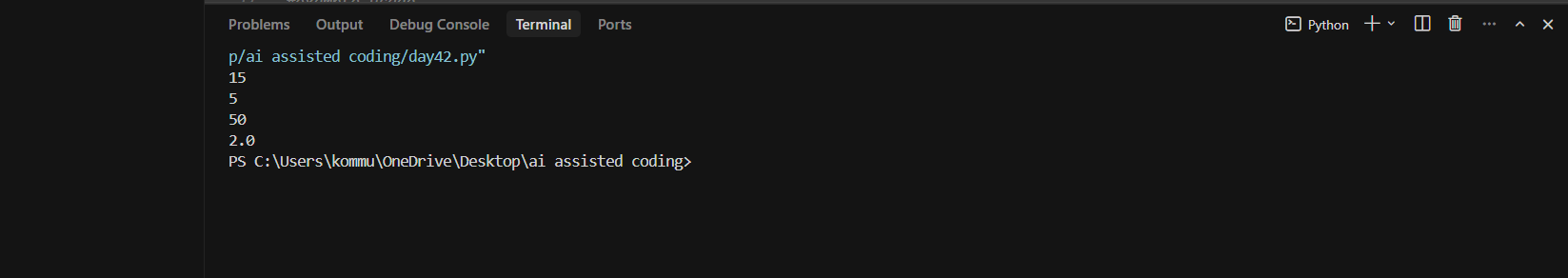
**Stage 2:**

****

**Stage3:**

****

**Output:**

****

**Own Experience or observation:**

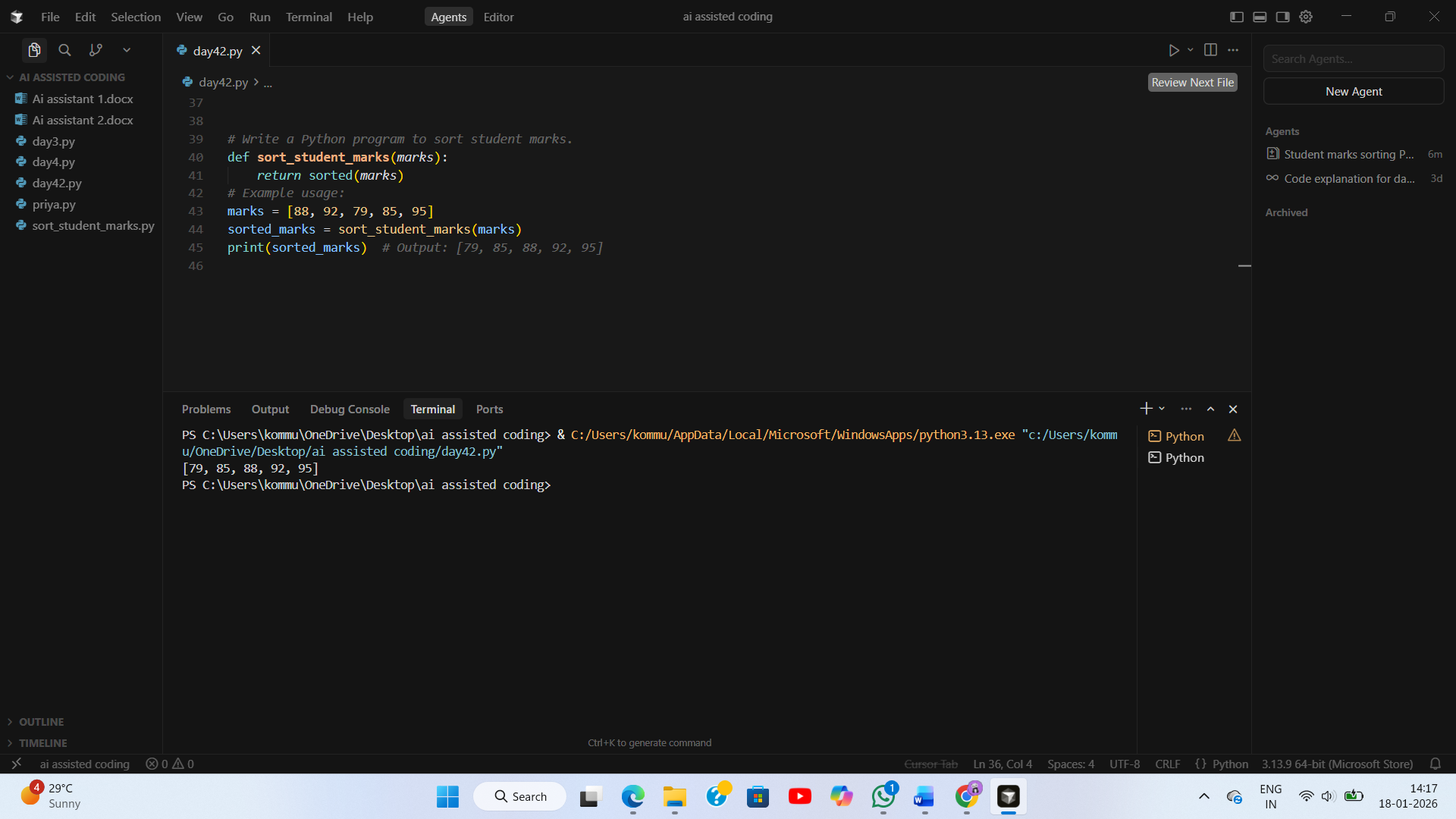
At first, when only the function name was given, the AI generated a very basic and incomplete calculator function with little or no logic. After adding comments, the AI started including parameters and arithmetic operations. When usage examples were finally added, the AI produced a complete and well-structured calculator program with proper conditions and error handling. This clearly shows that progressive prompting improves both the logic and structure of the generated code.

**Question 2:**

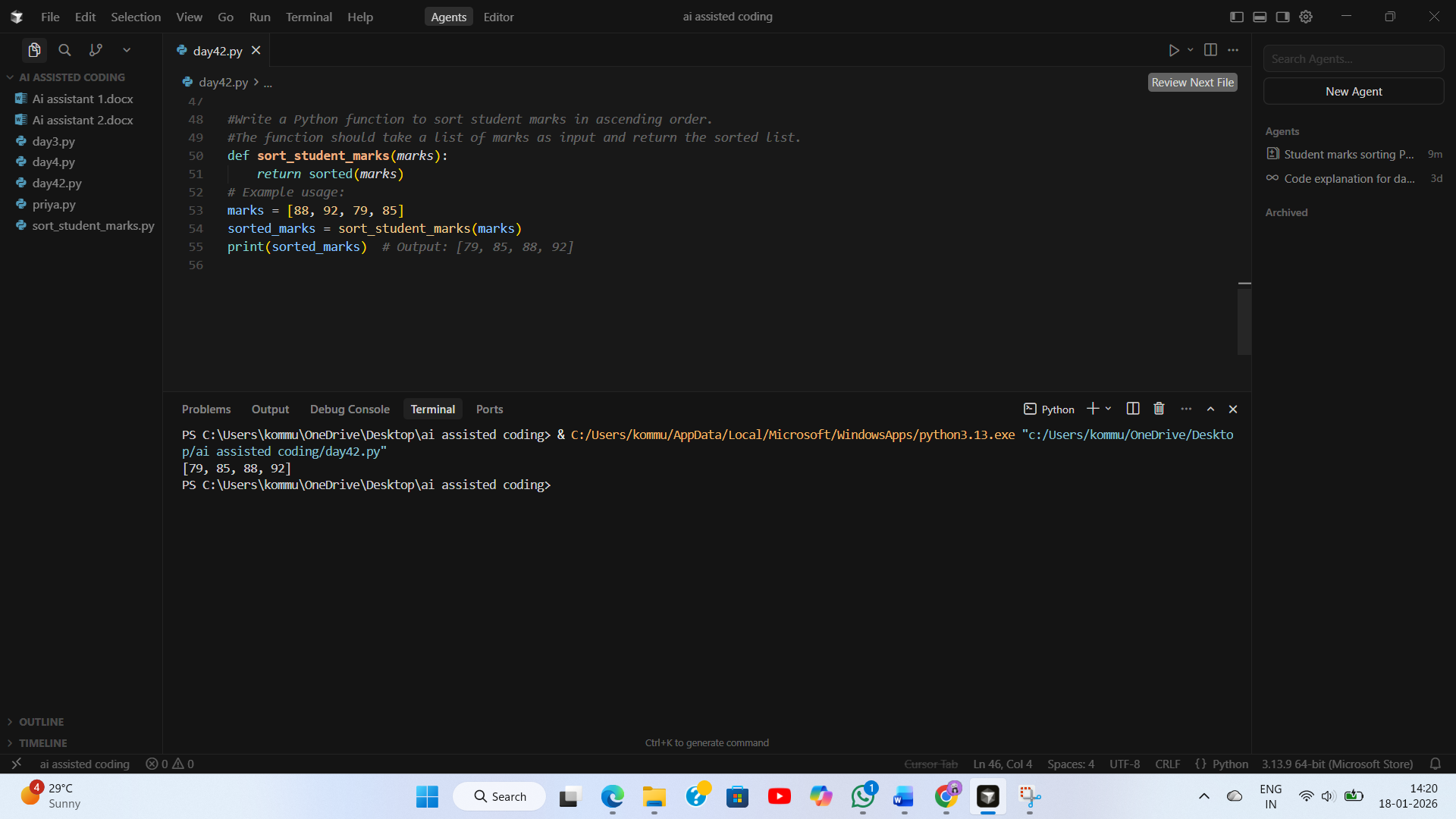
**Task Description-2** Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints.

**Expected Output-2** AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.

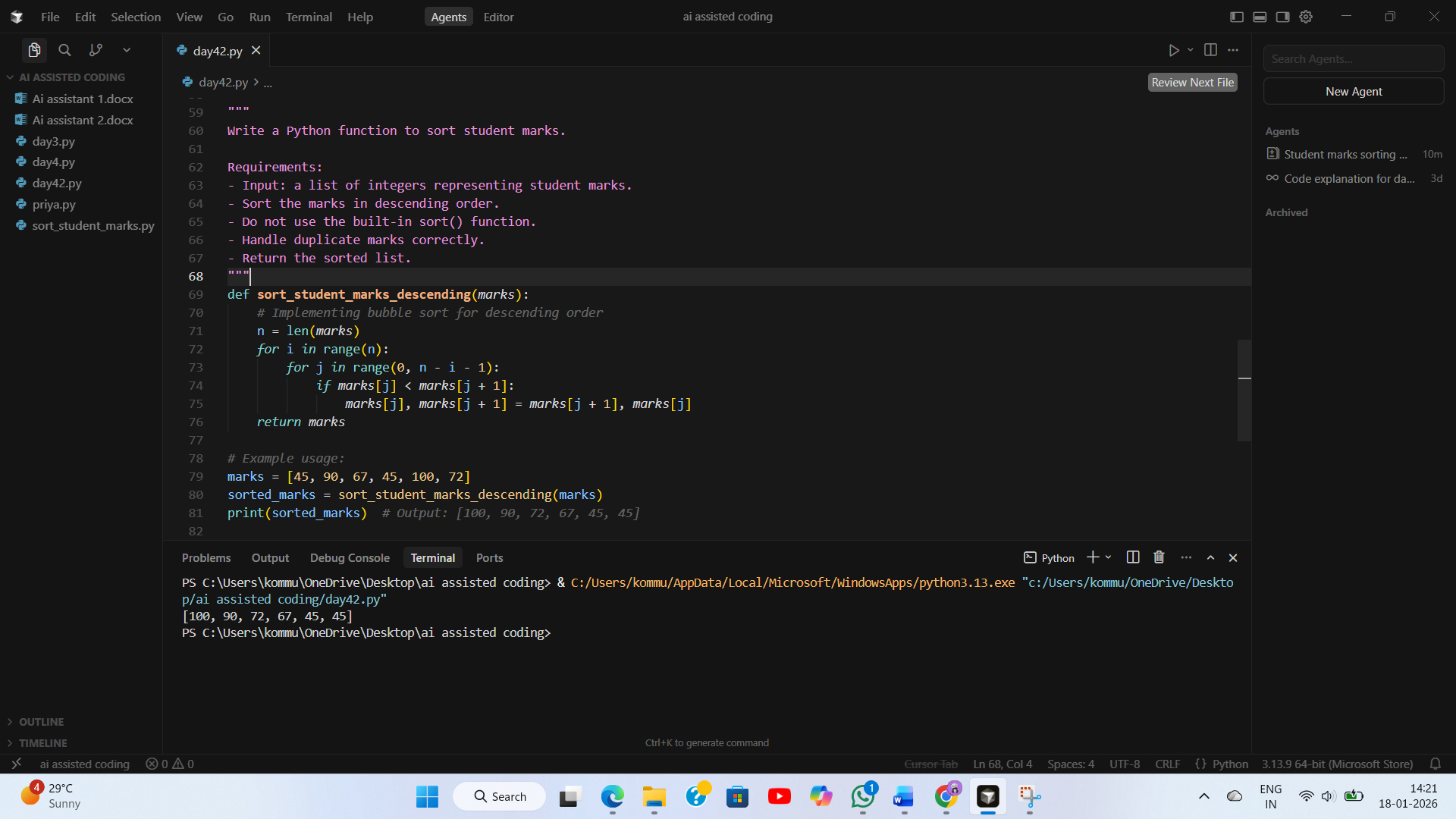
**Stage 1:**

****

**Stage2:**

****

**Stage3:**

****

**Observation:**

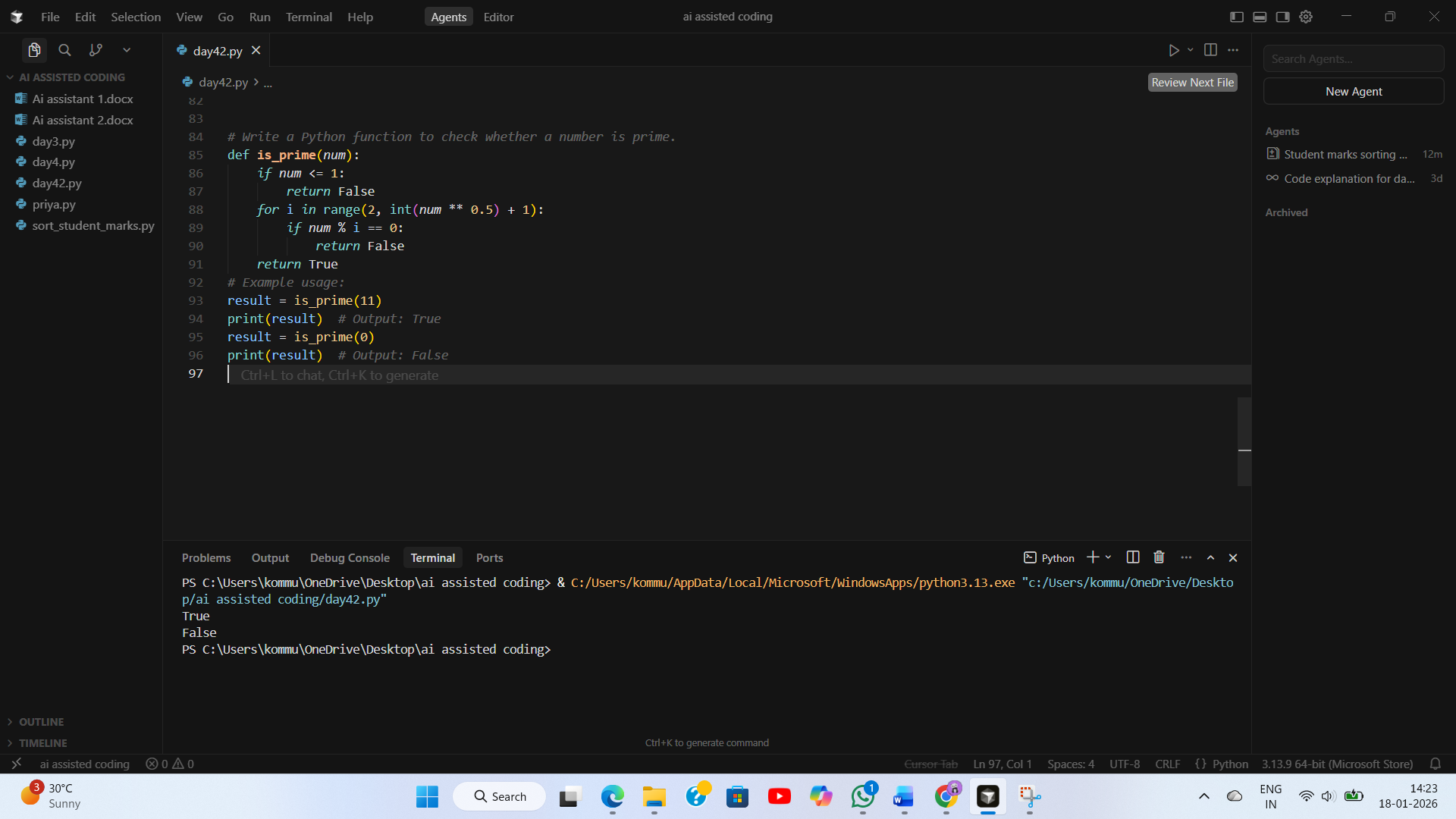
With a vague prompt, the AI produced a simple sorting solution without clear direction or constraints. After refining the prompt to specify sorting order, the output became more accurate and meaningful. When clear constraints and examples were added, the AI generated a more structured and efficient sorting function. This demonstrates that refining prompts helps the AI move from ambiguous logic to a correct and reliable implementation.

**Question 3:**

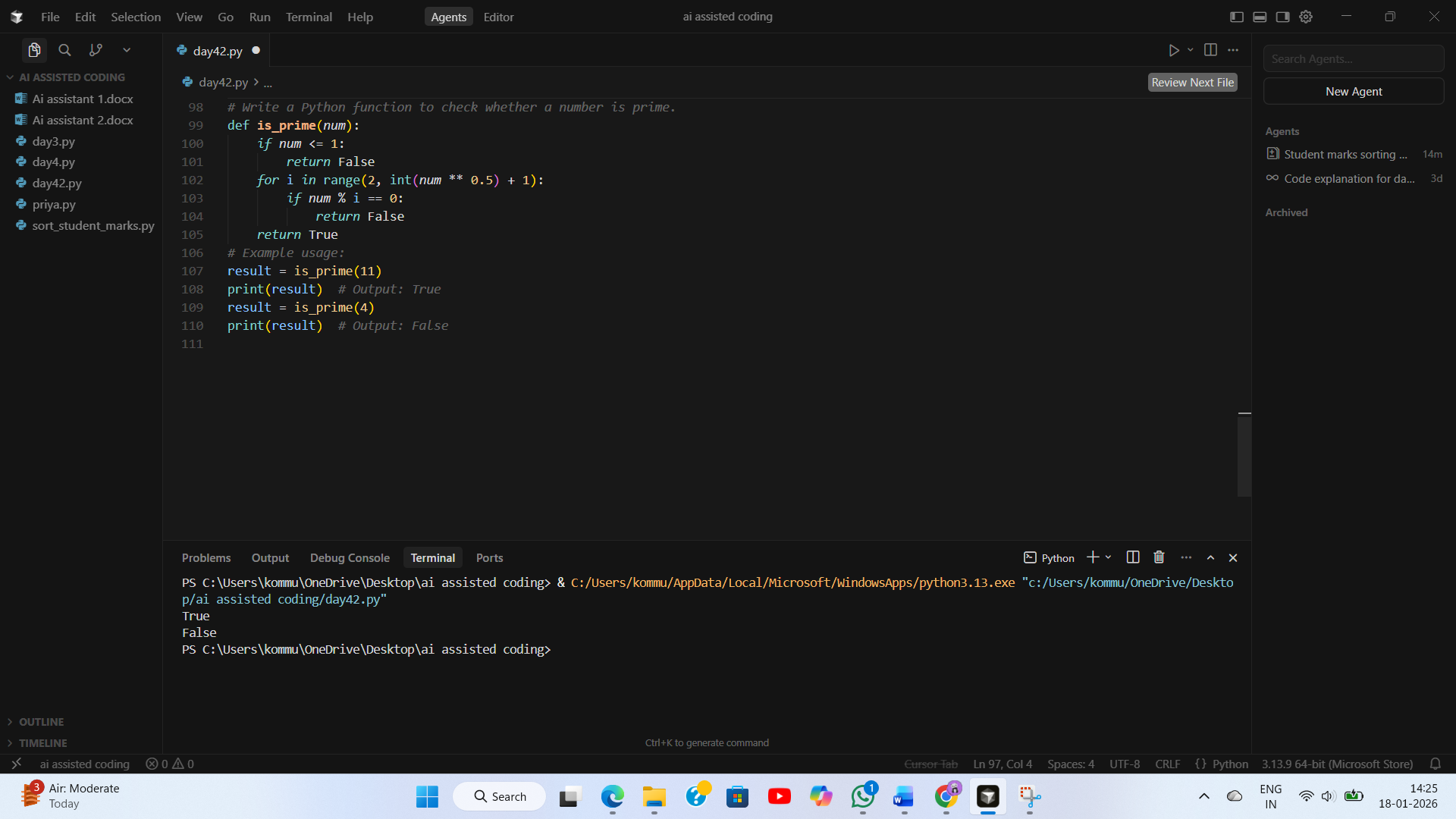
**Task Description-3** • Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness.

**Expected Output-3 •** Improved prime-checking function with better edge-case handling.

**Stage 1:**

****

**Stage2:**

****

**Stage3:**

****

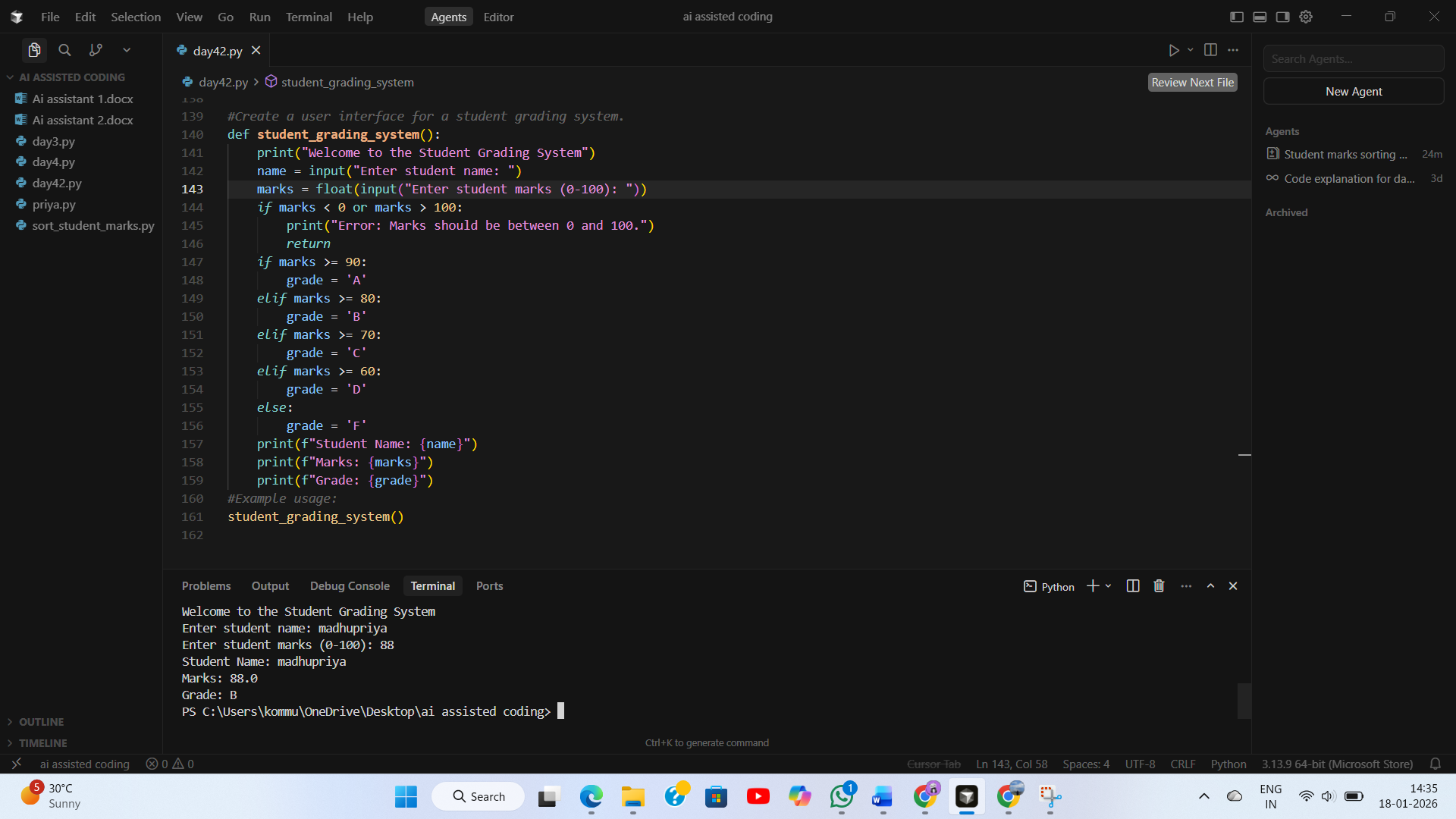
**Observation**: In the initial prompt without examples, the AI generated a basic prime-checking function that could miss important edge cases. When one example was provided, the result improved slightly. After giving multiple input-output examples (few-shot prompting), the AI clearly handled cases like 0, 1, and negative numbers and produced a more accurate and robust prime-checking function. This shows that few-shot prompting improves correctness and edge-case handling.

**Question 4:**

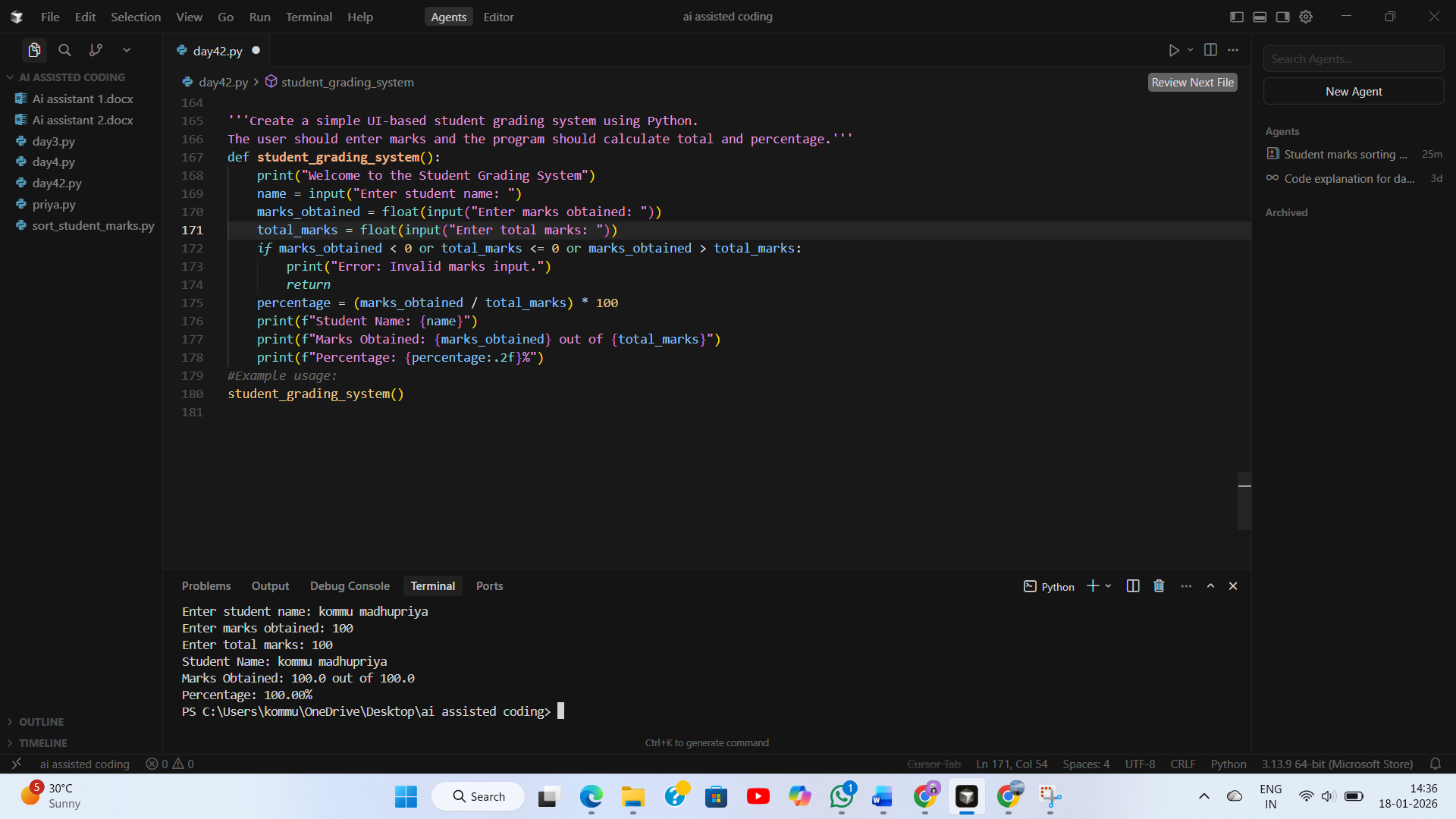
**Task Description-4 •** Prompt-Guided UI Design for Student Grading System: Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.

**Expected Output-4 •** Well-structured UI code with accurate calculations and clear output display.

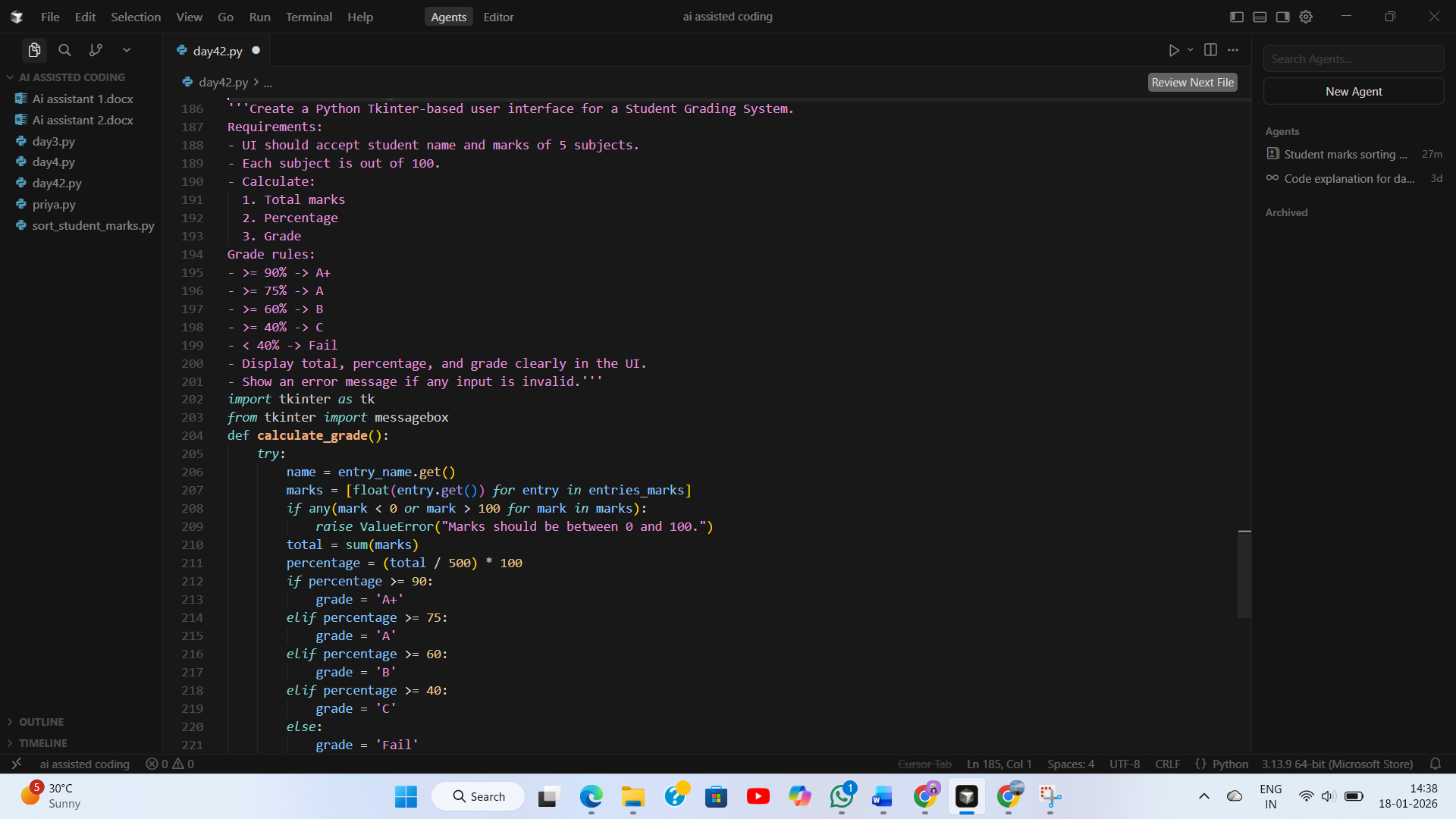
**Stage1:**

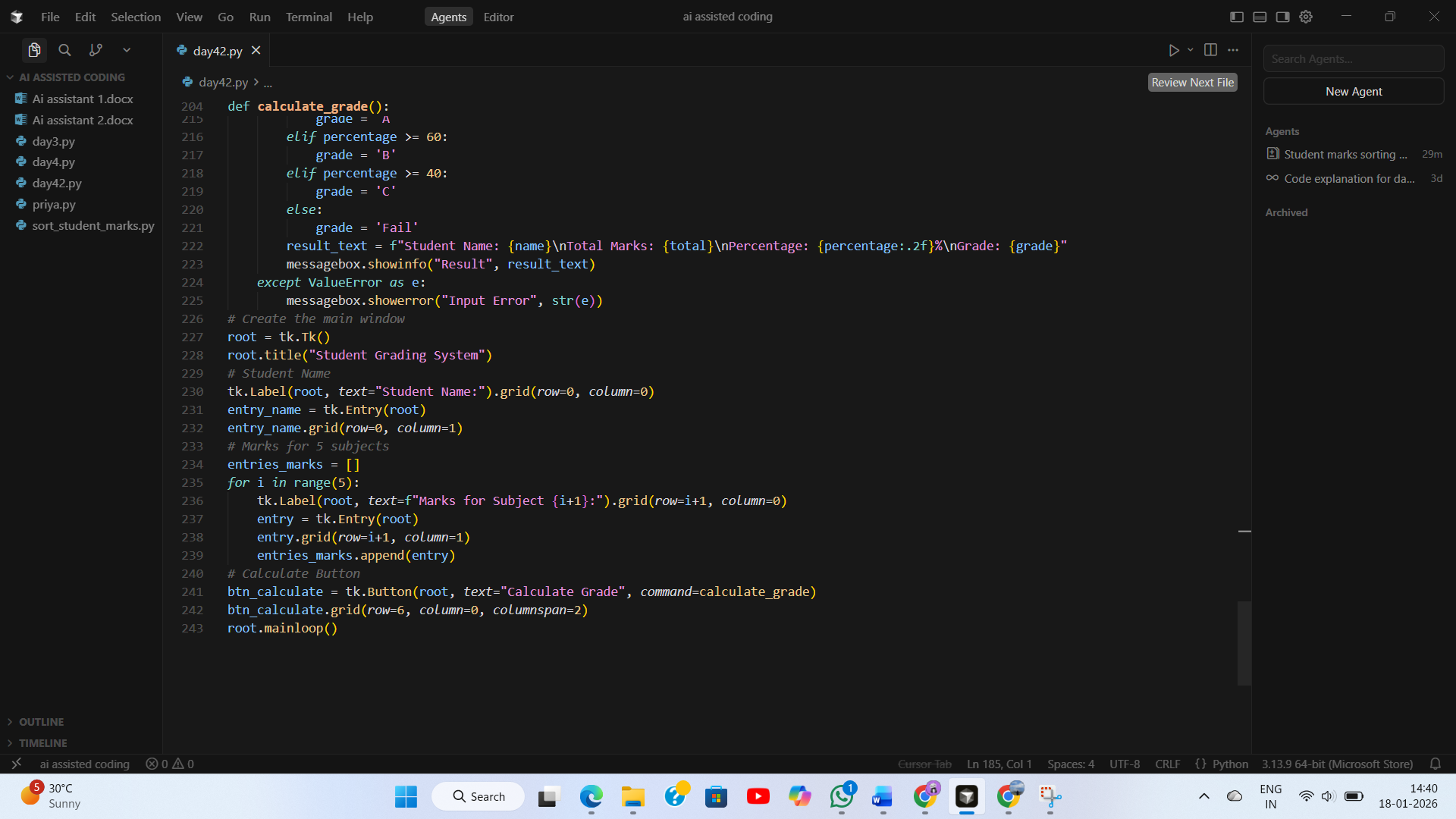
****

**Stage2:**

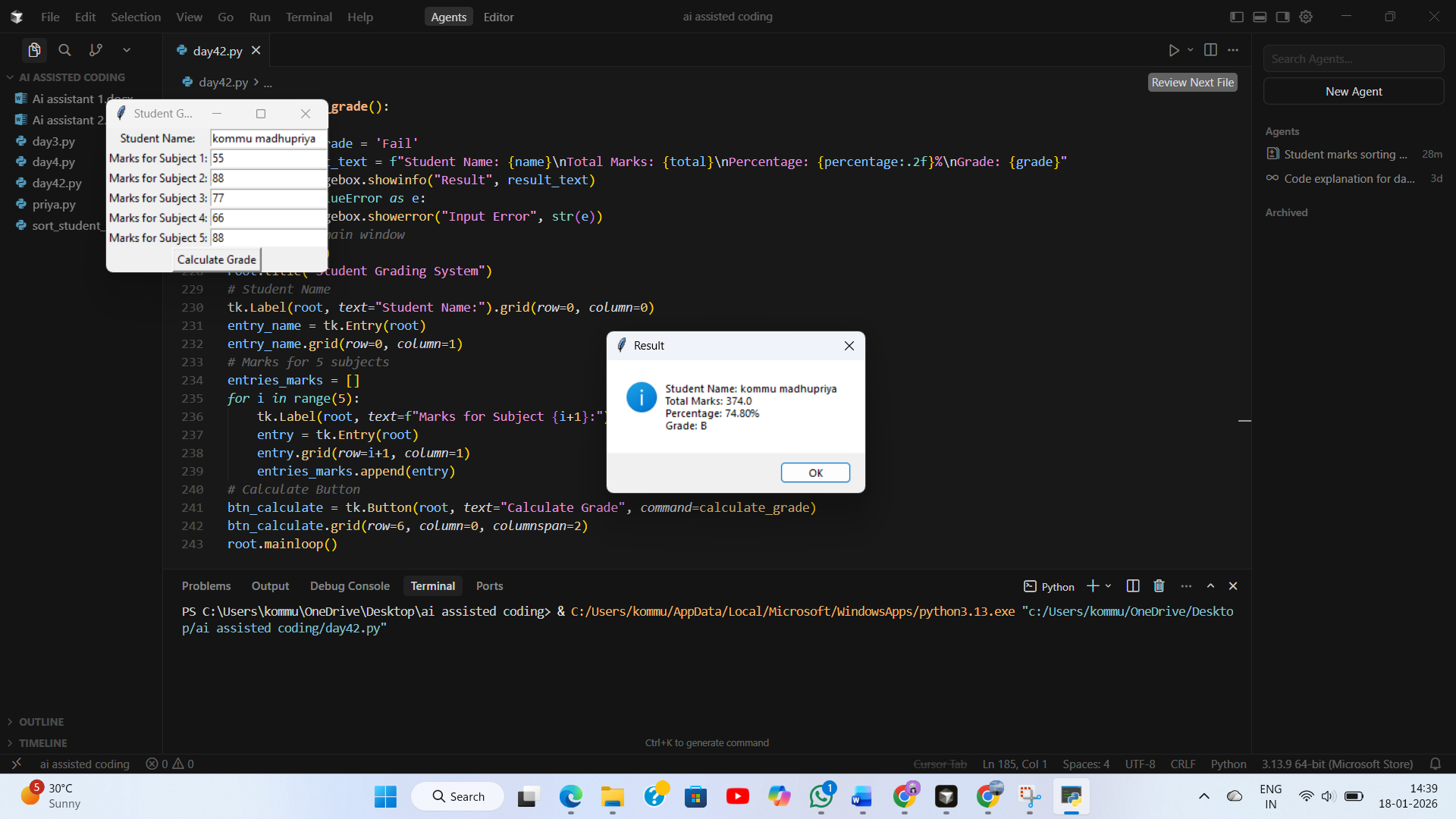
****

**Stage3:**

****

****

**Output:**

****

**Observation:**

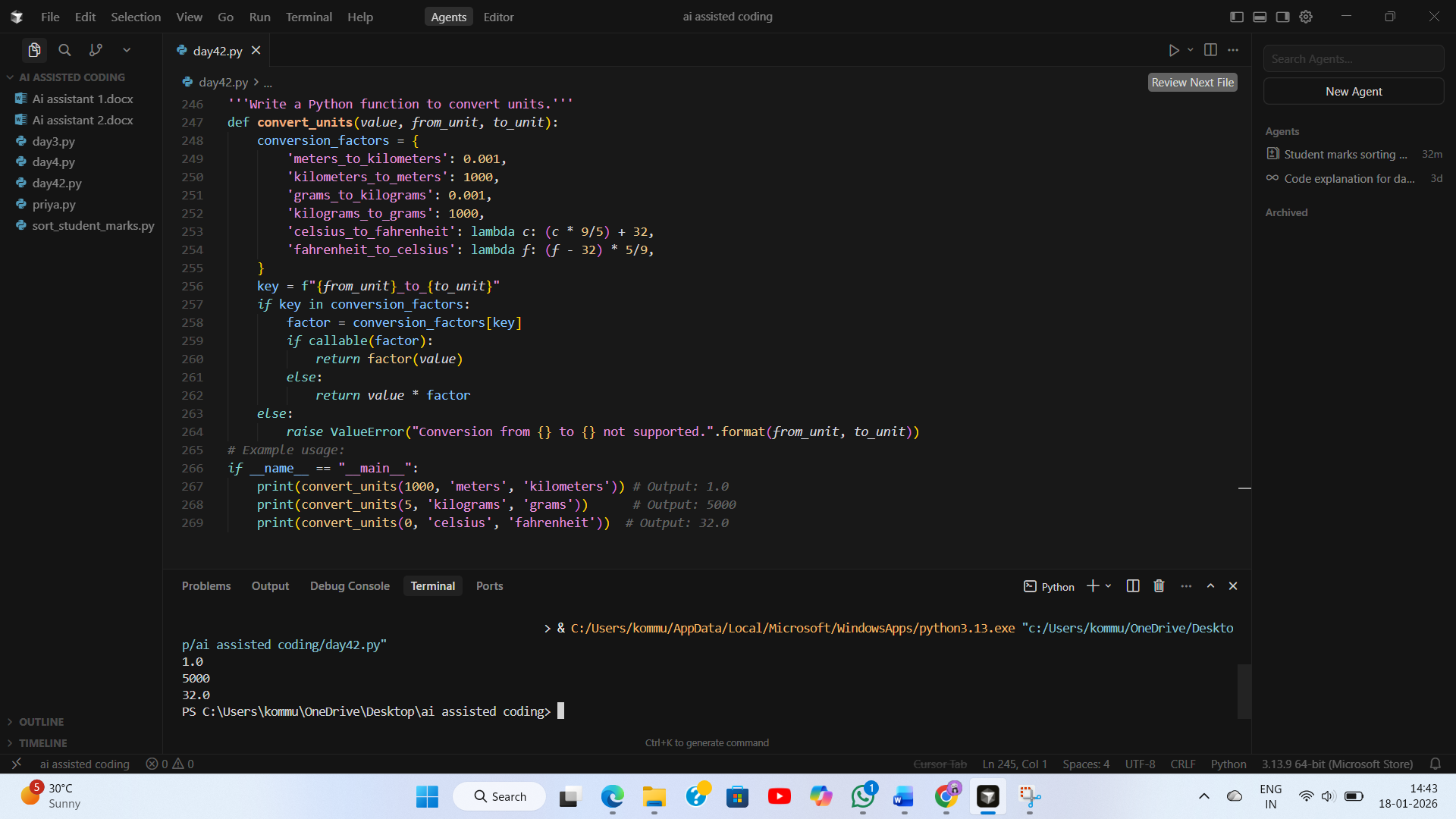
With a vague UI prompt, the AI produced only a simple or unclear interface idea. As the prompt was refined to include calculation requirements, the UI output became more meaningful. When full instructions were given (inputs, calculations, grade rules, and display), the AI generated a well-structured user interface with correct total, percentage, and grade calculation along with clear result display. This shows that prompt guidance greatly improves UI structure and usability.

**Question 5:**

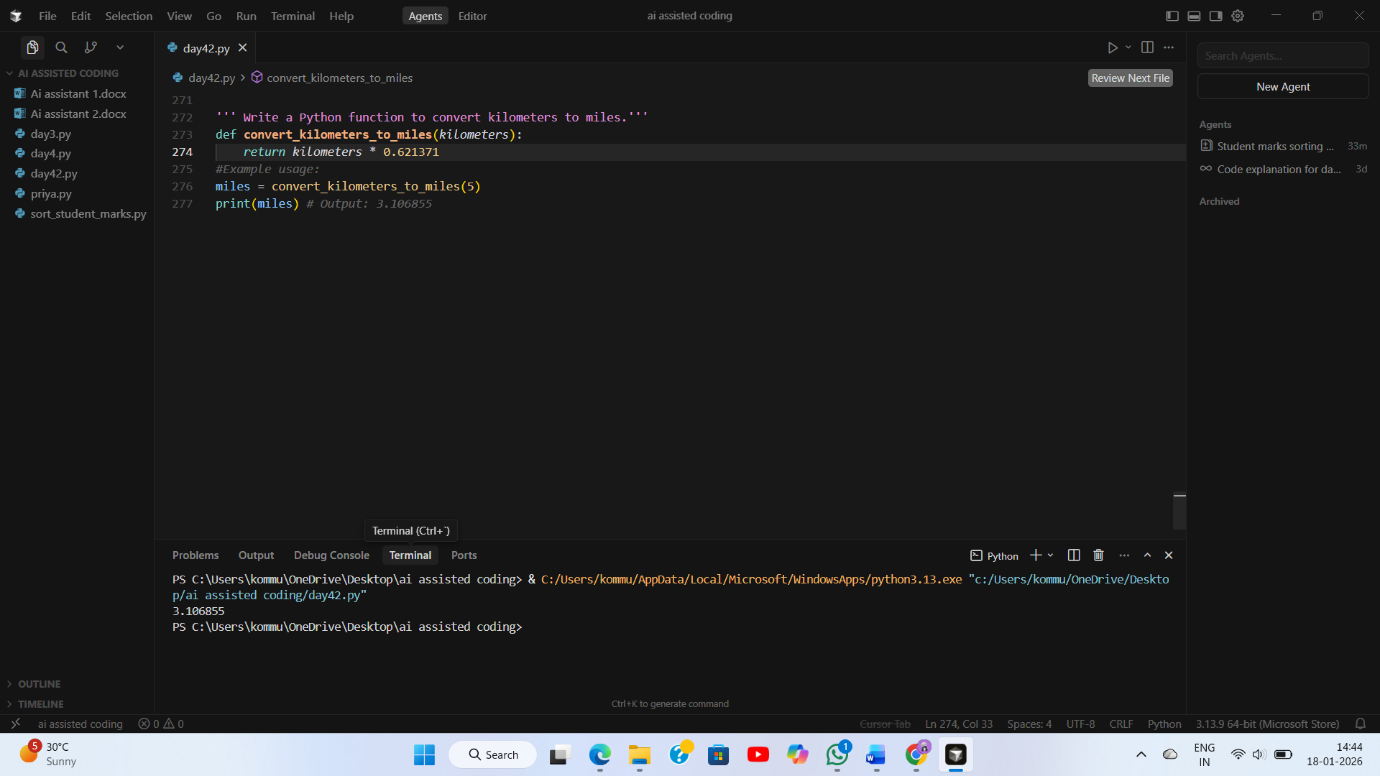
**Task Description-5** • Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.

**Expected Output-5** • Analysis of code quality and accuracy differences across multiple prompt variations.

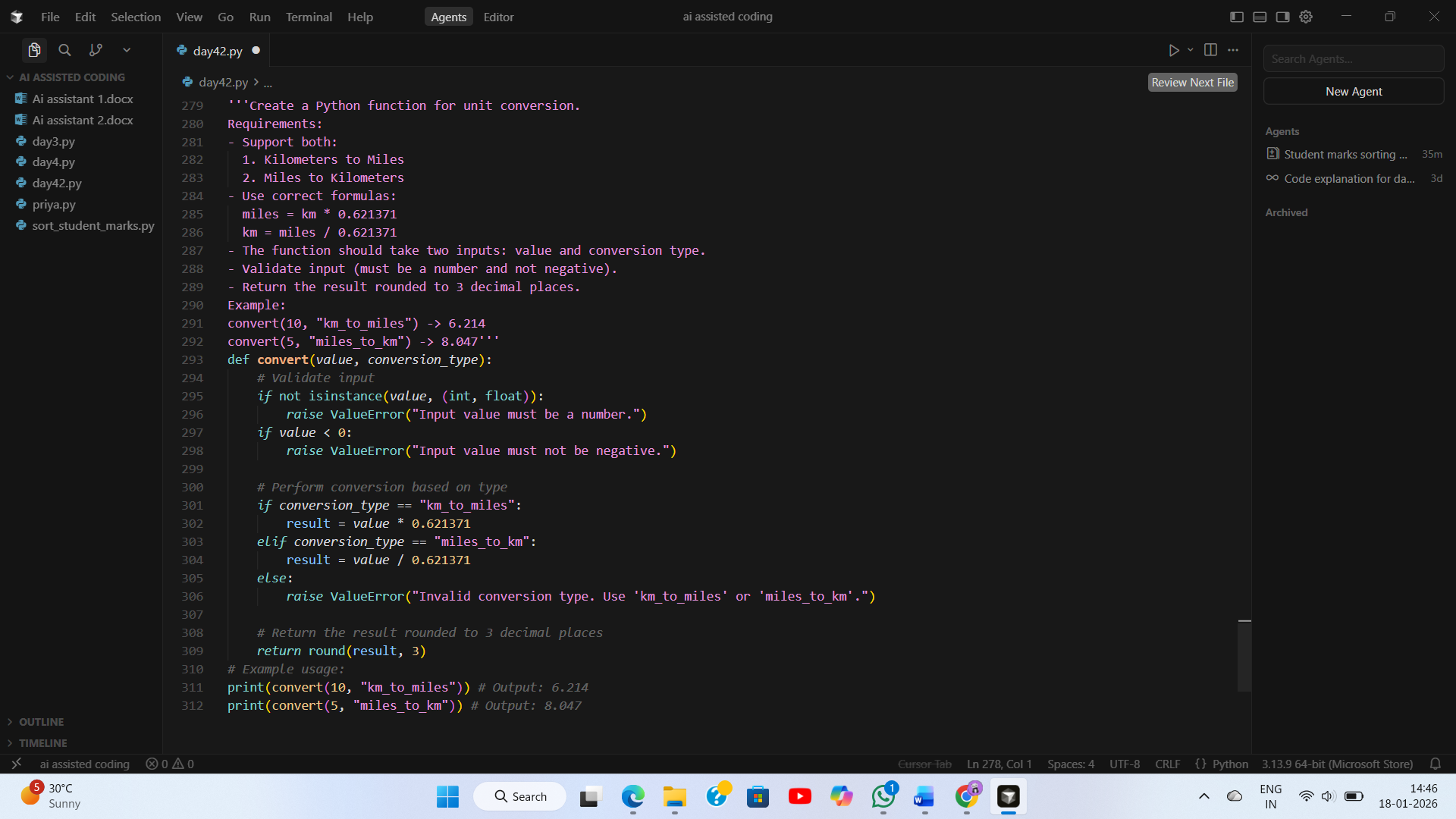
**Stage1:**

****

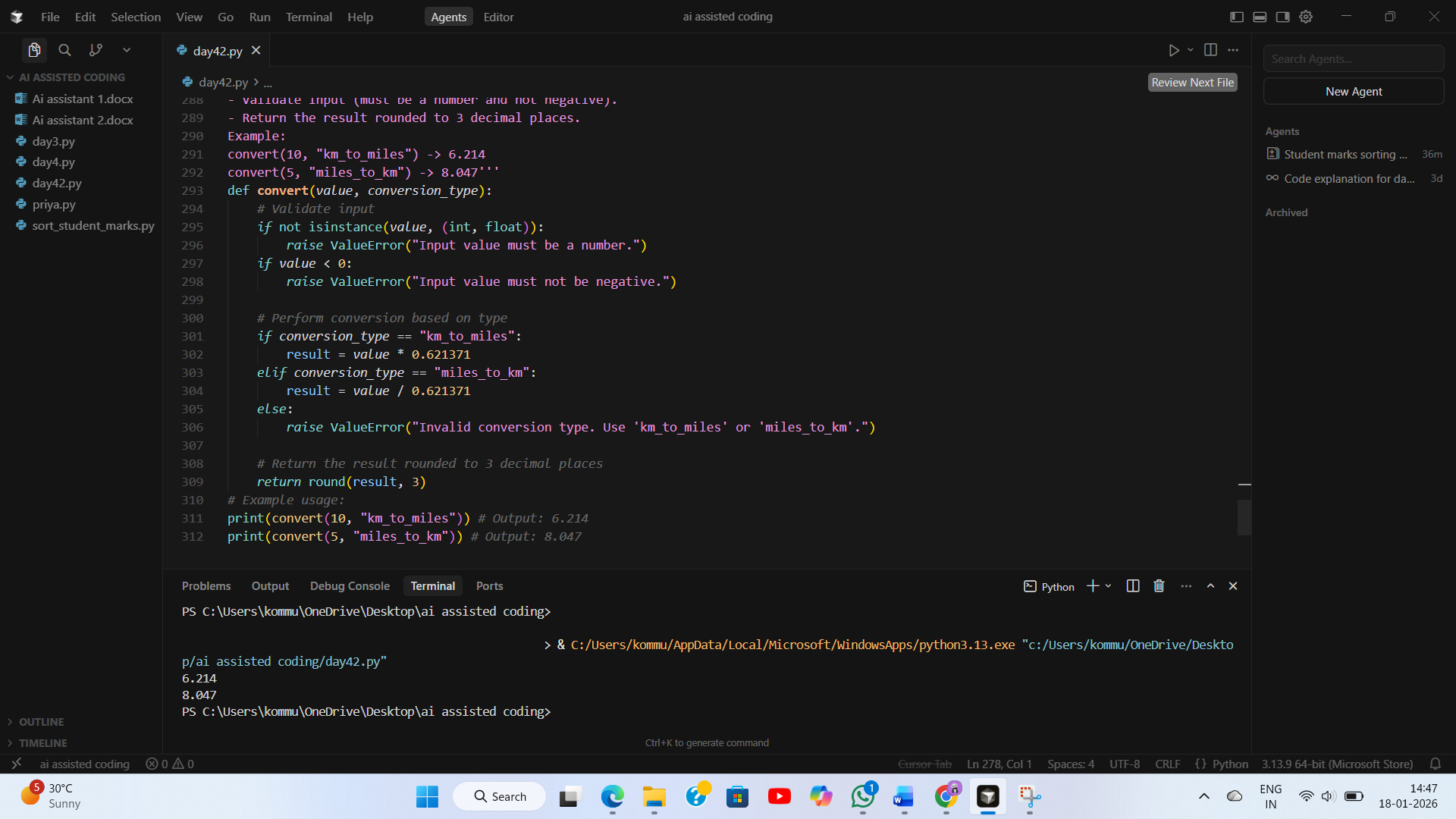
**Stage2:**



**Stage3:**

****

**Output:**

****

**Observation:**

When a vague prompt was used, the AI generated unclear or very general conversion code. After specifying the type of conversion, the AI produced a basic one-way converter. When detailed instructions, formulas, and validation rules were added, the AI generated an accurate, well-structured, and reusable unit conversion function. This proves that higher prompt specificity leads to better code quality, accuracy, and reliability.