

AI Assisted Coding

Assignment – 3.2

Name: Bandi Akiranandini

Batch:22

Hallticket:2303a51516

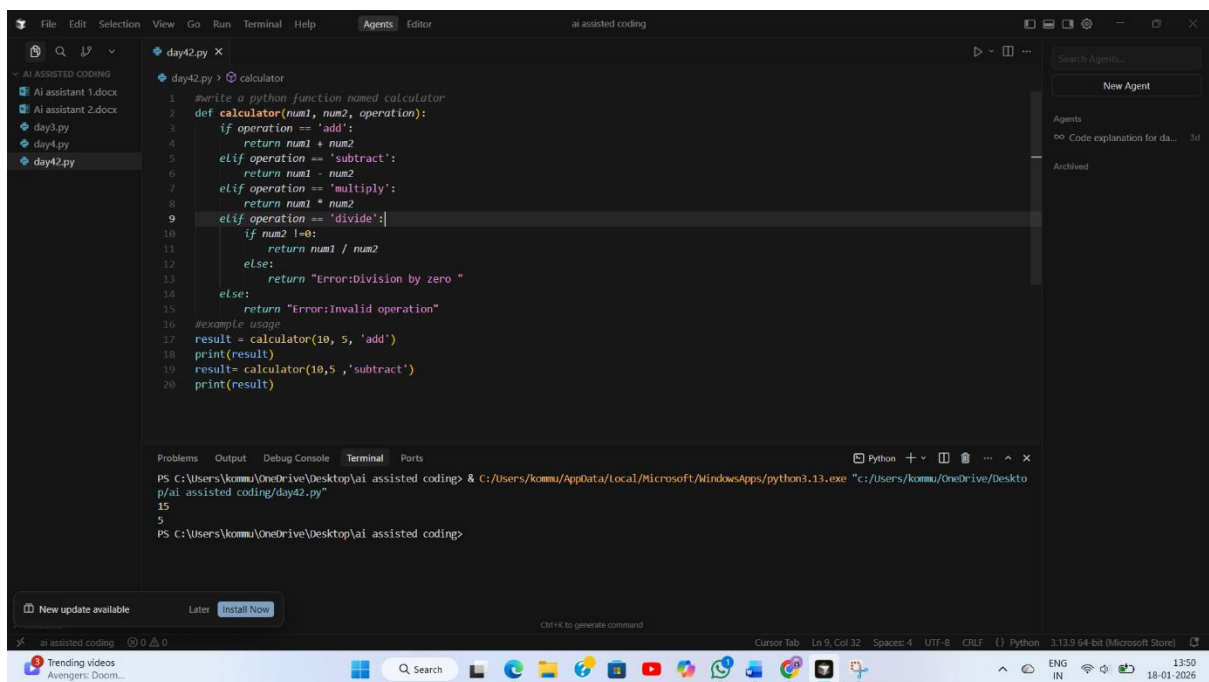
Task Description-1

- 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:



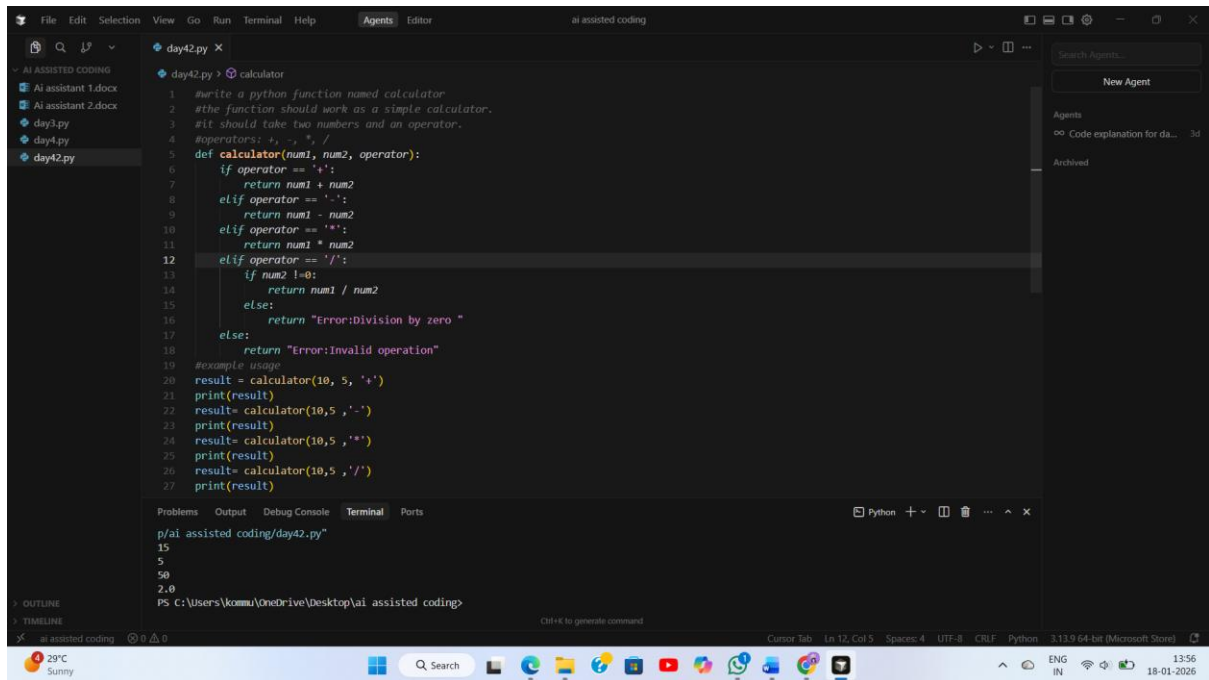
The screenshot displays a code editor with a Python file named `day42.py`. The code defines a `calculator` function that takes three arguments: `num1`, `num2`, and `operation`. It uses a series of `if` and `elif` statements to perform addition, subtraction, multiplication, and division. It also includes error handling for division by zero and invalid operations. Example usage is provided at the bottom of the code block.

```
1 #write a python function named calculator
2 def calculator(num1, num2, operation):
3     if operation == 'add':
4         return num1 + num2
5     elif operation == 'subtract':
6         return num1 - num2
7     elif operation == 'multiply':
8         return num1 * num2
9     elif operation == 'divide':
10        if num2 != 0:
11            return num1 / num2
12        else:
13            return "Error:Division by zero "
14    else:
15        return "Error:Invalid operation"
16
17 #example usage
18 result = calculator(10, 5, 'add')
19 print(result)
20 result= calculator(10,5 ,'subtract')
21 print(result)
```

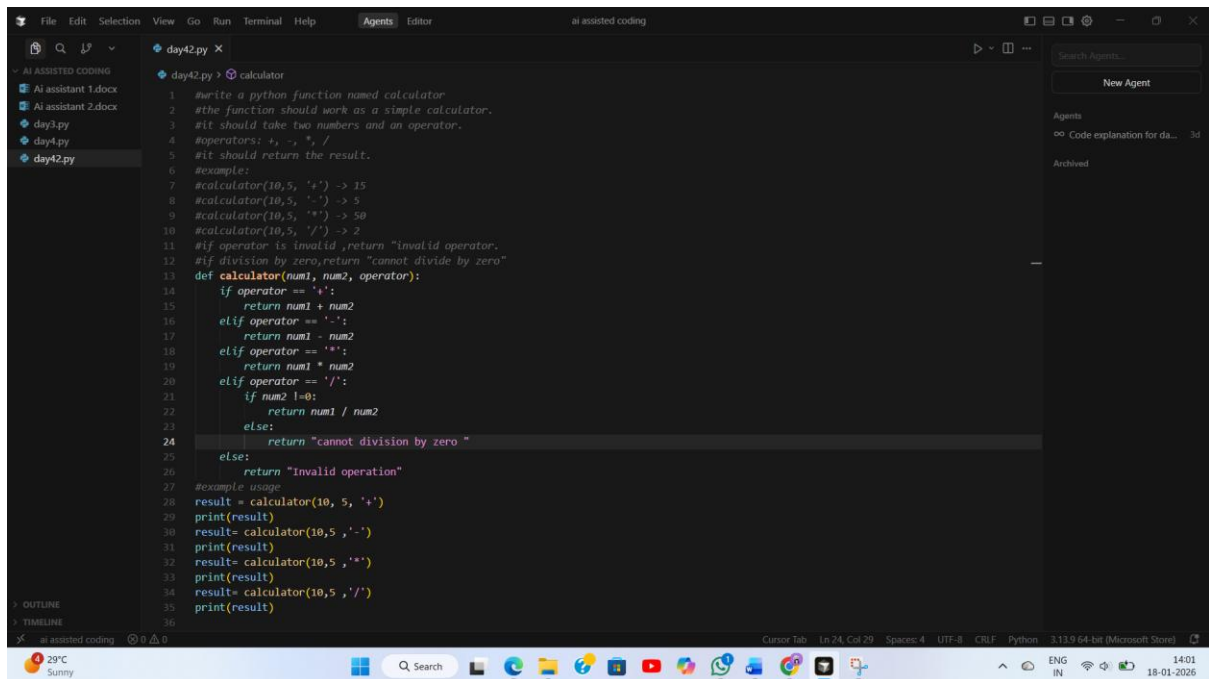
The terminal at the bottom shows the command to run the script and the output:

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/kommu/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/kommu/OneDrive/Desktop/p/ai assisted coding/day42.py"
15
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

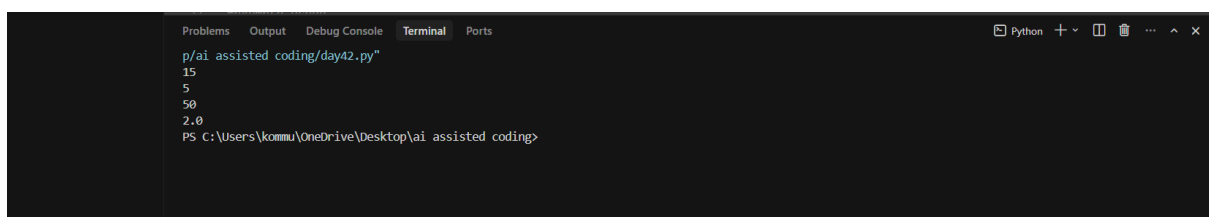
Stage 2:



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:

```
File Edit Selection View Go Run Terminal Help Agents Editor ai assisted coding
day42.py x
AI ASSISTED CODING
AI assistant 1.docx
AI assistant 2.docx
day3.py
day4.py
day42.py
priya.py
sort_student_marks.py
day42.py > ...
37
38
39 # Write a Python program to sort student marks.
40 def sort_student_marks(marks):
41     return sorted(marks)
42 # Example usage:
43 marks = [88, 92, 79, 85, 95]
44 sorted_marks = sort_student_marks(marks)
45 print(sorted_marks) # output: [79, 85, 88, 92, 95]
46

Problems Output Debug Console Terminal Ports
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:\Users\kommu\AppData\Local\Microsoft\WindowsApps\python3.13.exe "C:\Users\kommu\OneDrive\Desktop\ai assisted coding\day42.py"
[79, 85, 88, 92, 95]
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>

OUTLINE
TIMELINE
ai assisted coding 0 0 0
Ctrl+K to generate command
Python 3.13.9 64-bit (Microsoft Store)
```

Stage2:

```
47
48 #Write a Python function to sort student marks in ascending order.
49 #The function should take a list of marks as input and return the sorted list.
50 def sort_student_marks(marks):
51     return sorted(marks)
52 # Example usage:
53 marks = [88, 92, 79, 85]
54 sorted_marks = sort_student_marks(marks)
55 print(sorted_marks) # Output: [79, 85, 88, 92]
56
```

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/kommu/AppData/Local/Microsoft/WindowsApps/python3.13.exe "C:/Users/kommu/OneDrive/Desktop/p/ai assisted coding/day42.py"
[79, 85, 88, 92]
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

Stage3:

```
59 """
60 Write a Python function to sort student marks.
61
62 Requirements:
63 - Input: a list of integers representing student marks.
64 - Sort the marks in descending order.
65 - Do not use the built-in sort() function.
66 - Handle duplicate marks correctly.
67 - Return the sorted list.
68 """
69 def sort_student_marks_descending(marks):
70     # Implementing bubble sort for descending order
71     n = len(marks)
72     for i in range(n):
73         for j in range(0, n - i - 1):
74             if marks[j] < marks[j + 1]:
75                 marks[j], marks[j + 1] = marks[j + 1], marks[j]
76     return marks
77
78 # Example usage:
79 marks = [45, 90, 67, 45, 100, 72]
80 sorted_marks = sort_student_marks_descending(marks)
81 print(sorted_marks) # Output: [100, 90, 72, 67, 45, 45]
82
```

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/kommu/AppData/Local/Microsoft/WindowsApps/python3.13.exe "C:/Users/kommu/OneDrive/Desktop/p/ai assisted coding/day42.py"
[100, 90, 72, 67, 45, 45]
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

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

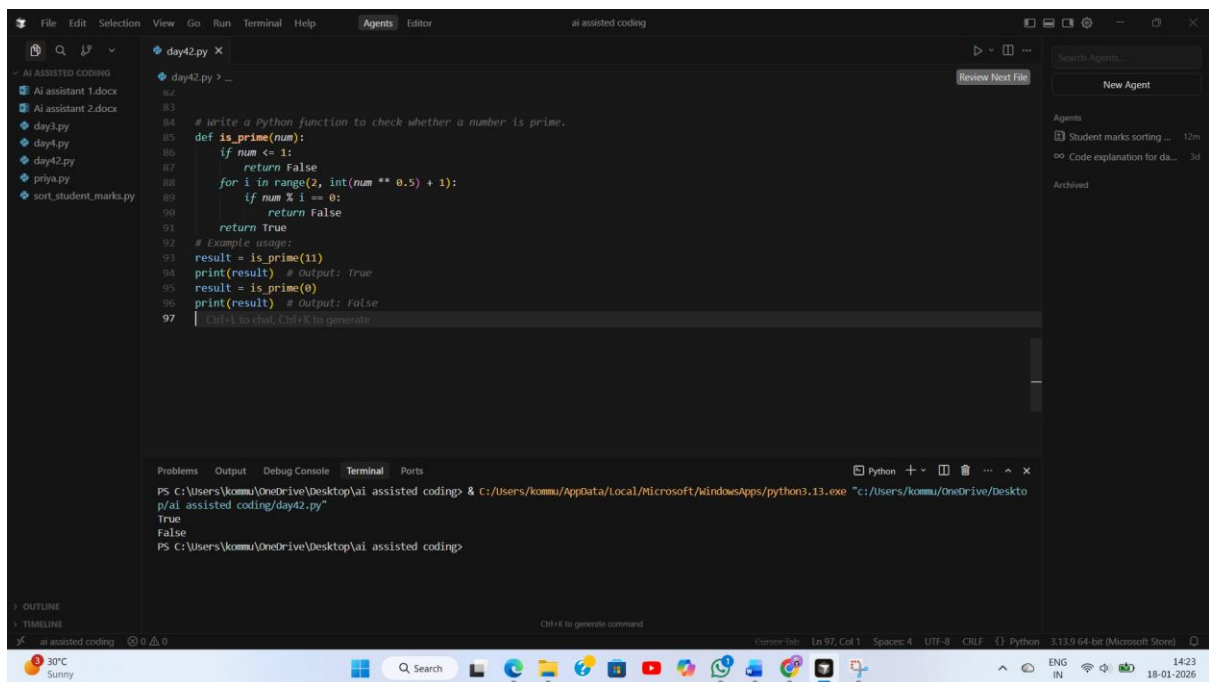
and efficient sorting function. This demonstrates that refining prompts helps the AI move from ambiguous logic to a correct and reliable implementation.

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:



The screenshot shows a Visual Studio Code editor window with a file named `day42.py`. The code is a Python function `is_prime(num)` designed to check if a number is prime. It includes a few-shot prompt as a comment: `# write a Python function to check whether a number is prime.` The function logic is as follows:

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True
```

Below the function definition, there are example usages:

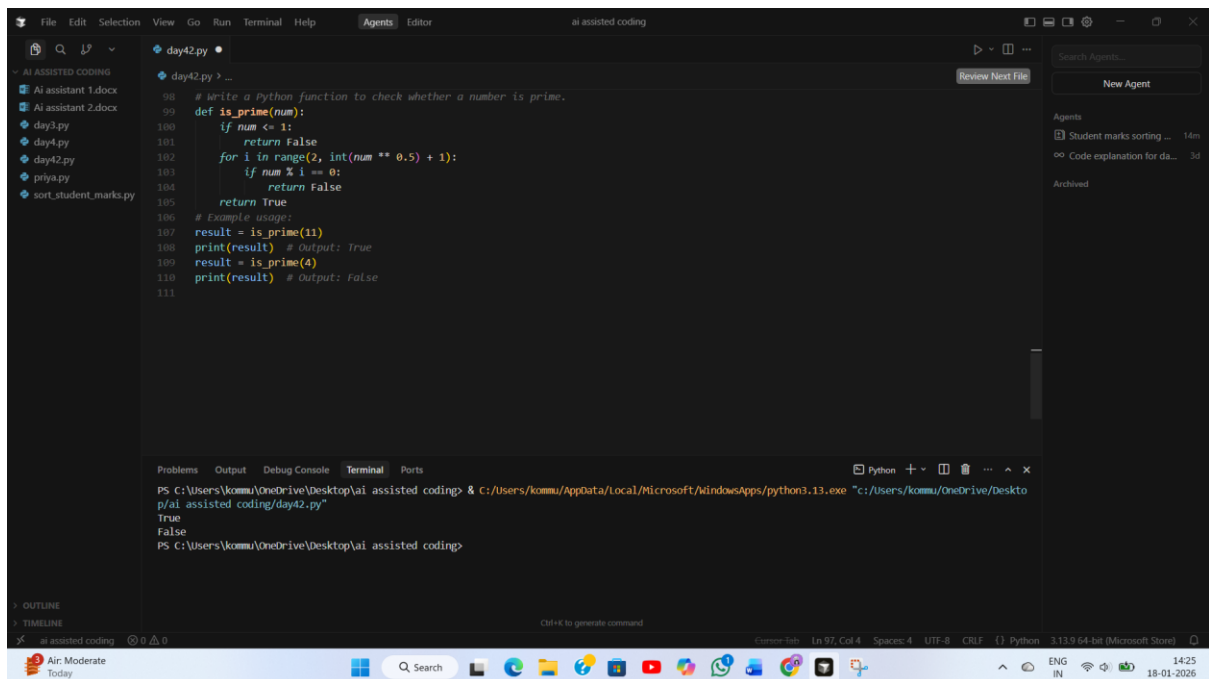
```
# Example usage:
result = is_prime(11)
print(result) # output: True
result = is_prime(0)
print(result) # output: False
```

The terminal at the bottom shows the command to run the script and its output:

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/komm/OneDrive/Desktop/ai assisted coding/day42.py
True
False
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

The status bar at the bottom indicates the file is `day42.py`, line 97, column 1, with 4 spaces, using UTF-8 encoding and CRLF line endings. The Python version is 3.11.9 64-bit (Microsoft Store).

Stage2:



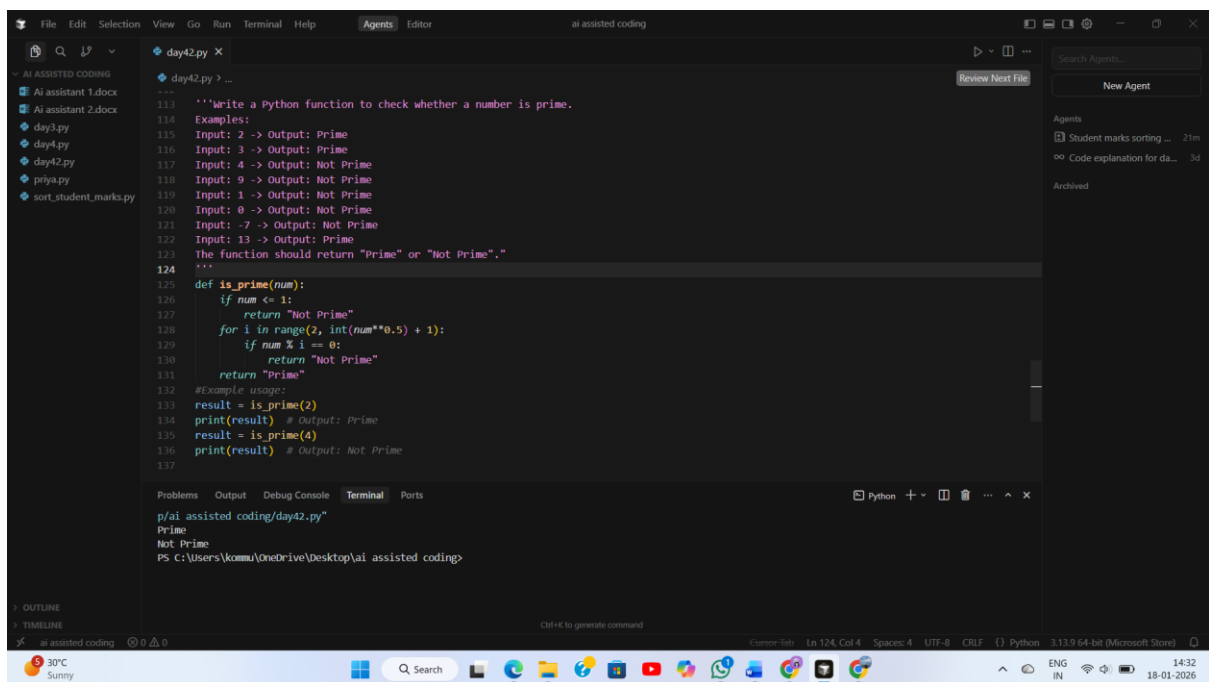
The screenshot shows the VS Code interface with a file explorer on the left containing files like `day3.py`, `day4.py`, `day42.py`, `priya.py`, and `sort_student_marks.py`. The main editor displays the `day42.py` file with the following Python code:

```
'''Write a Python function to check whether a number is prime.'''
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
# Example usage:
result = is_prime(11)
print(result) # Output: True
result = is_prime(4)
print(result) # Output: False
```

The terminal at the bottom shows the command `python p/ai assisted coding/day42.py` being executed, with the output:

```
True
False
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

Stage3:



The screenshot shows the VS Code interface with the `day42.py` file updated to include examples and a more robust prime-checking function. The code is as follows:

```
'''Write a Python function to check whether a number is prime.'''
Examples:
Input: 2 -> Output: Prime
Input: 3 -> Output: Prime
Input: 4 -> Output: Not Prime
Input: 9 -> Output: Not Prime
Input: 1 -> Output: Not Prime
Input: 0 -> Output: Not Prime
Input: -7 -> Output: Not Prime
Input: 13 -> Output: Prime
The function should return "Prime" or "Not Prime".'''
def is_prime(num):
    if num <= 1:
        return "Not Prime"
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return "Not Prime"
    return "Prime"
#Example usage:
result = is_prime(2)
print(result) # Output: Prime
result = is_prime(4)
print(result) # Output: Not Prime
```

The terminal shows the command `python p/ai assisted coding/day42.py` being executed, with the output:

```
Prime
Not Prime
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

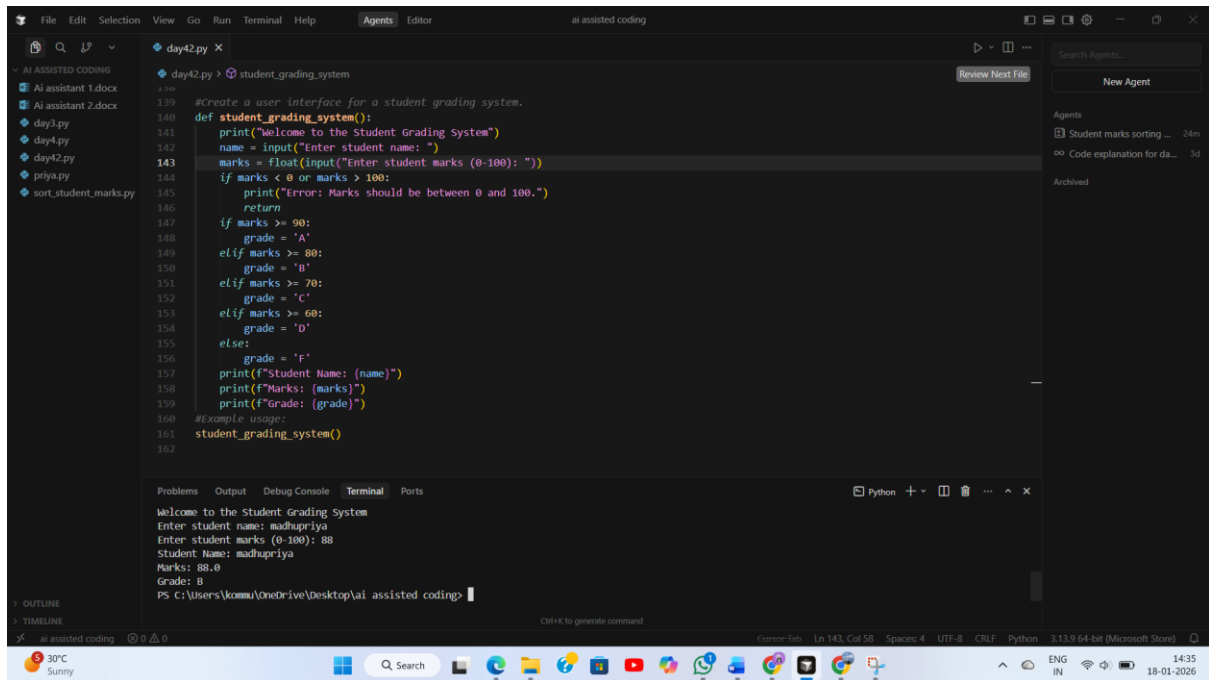
Observation: In the initial prompt without examples, the AI generated a basic prime-checking function that could miss important edge cases. When one example was provided, the result improved slightly. After giving multiple input-output examples (few-shot prompting), the AI clearly handled cases like 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.

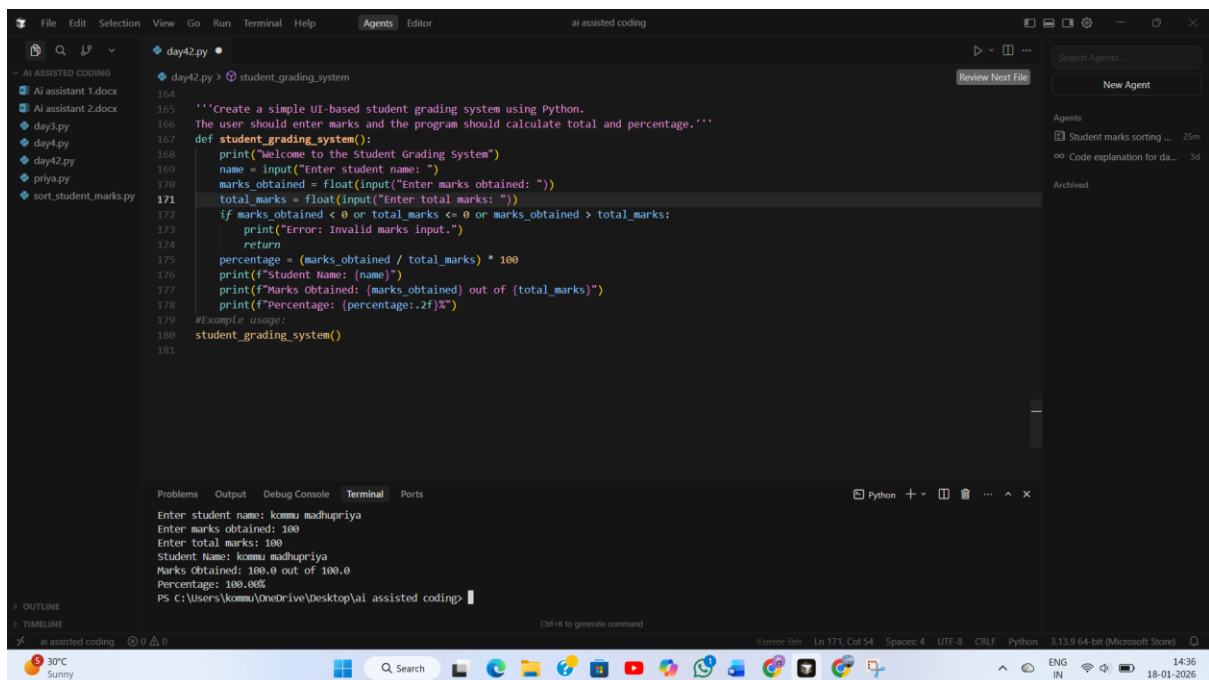
Expected Output-4 • Well-structured UI code with accurate calculations and clear output display.

Stage1:



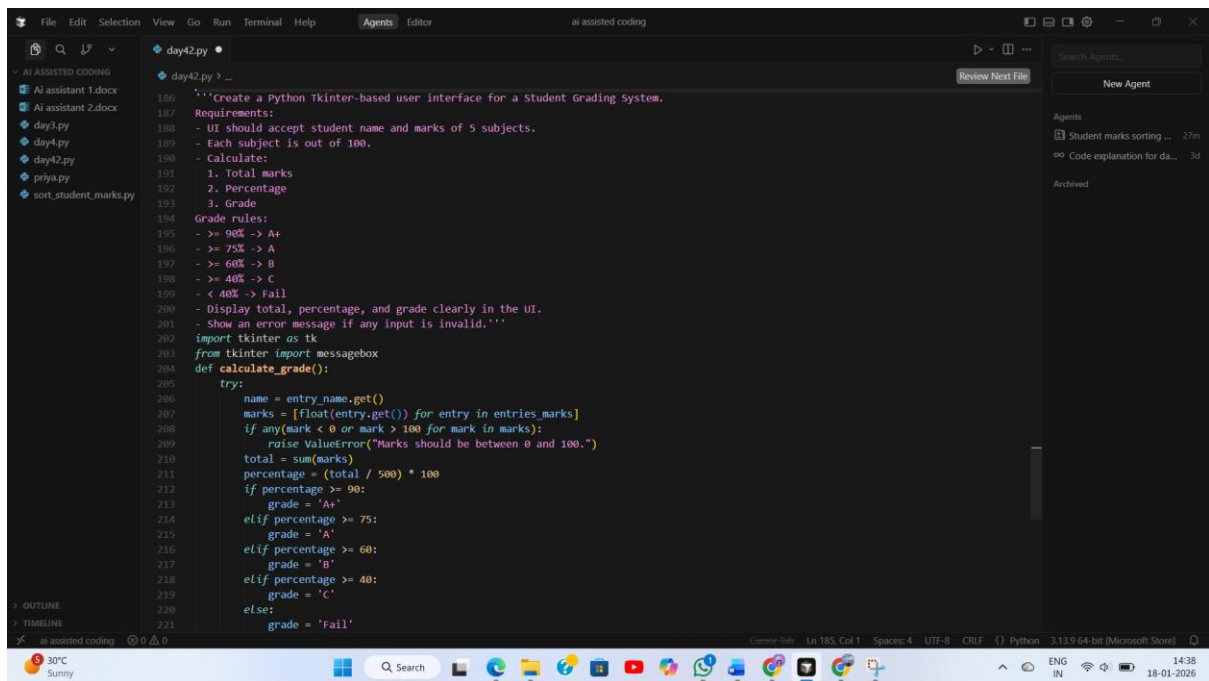
```
File Edit Selection View Go Run Terminal Help Agents Editor ai assisted coding
day42.py x
AI ASSISTED CODING
AI assistant 1.docx
AI assistant 2.docx
day3.py
day4.py
day42.py
priya.py
sort_student_marks.py
day42.py > student_grading_system
# Create a user interface for a student grading system.
139 def student_grading_system():
140     print("Welcome to the Student Grading System")
141     name = input("Enter student name: ")
142     marks = float(input("Enter student marks (0-100): "))
143     if marks < 0 or marks > 100:
144         print("Error: Marks should be between 0 and 100.")
145         return
146     if marks >= 90:
147         grade = 'A'
148     elif marks >= 80:
149         grade = 'B'
150     elif marks >= 70:
151         grade = 'C'
152     elif marks >= 60:
153         grade = 'D'
154     else:
155         grade = 'F'
156     print(f"Student Name: {name}")
157     print(f"Marks: {marks}")
158     print(f"Grade: {grade}")
159     # Example usage:
160     student_grading_system()
161
Problems Output Debug Console Terminal Ports
Welcome to the Student Grading System
Enter student name: madhupriya
Enter student marks (0-100): 88
Student Name: madhupriya
Marks: 88.0
Grade: B
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

Stage2:



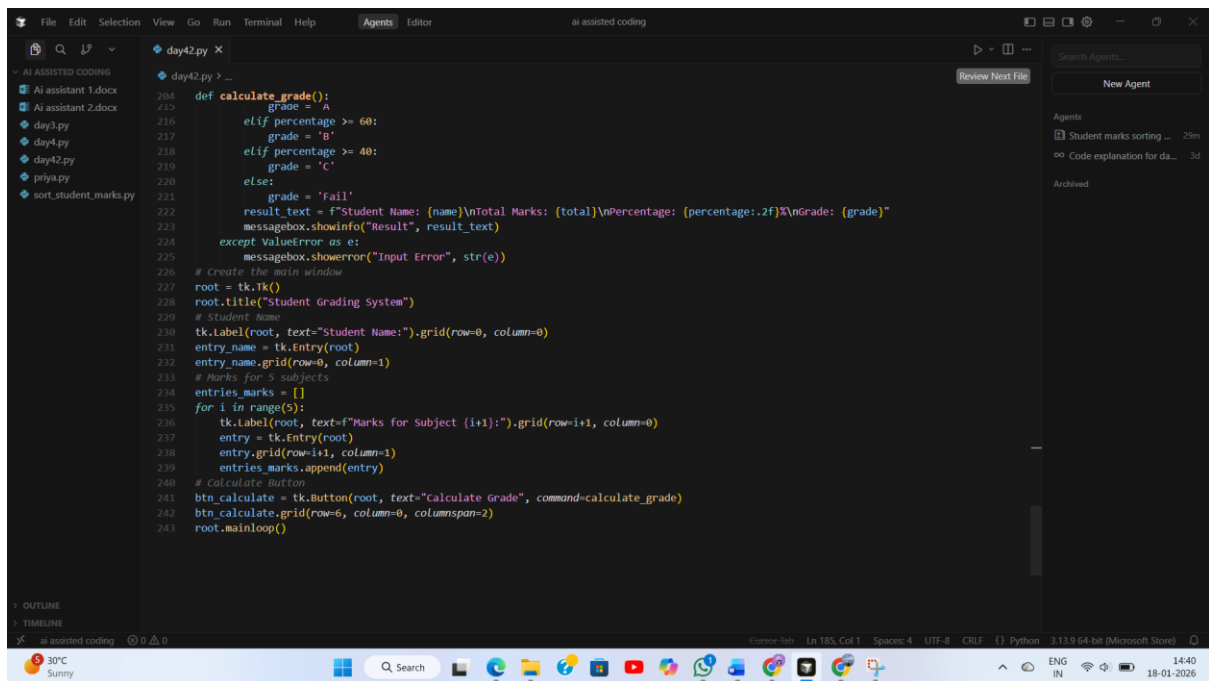
```
File Edit Selection View Go Run Terminal Help Agents Editor ai assisted coding
day42.py •
AI ASSISTED CODING
AI assistant 1.docx
AI assistant 2.docx
day3.py
day4.py
day42.py
priya.py
sort_student_marks.py
day42.py > student_grading_system
'''Create a simple UI-based student grading system using Python.
The user should enter marks and the program should calculate total and percentage.'''
164 def student_grading_system():
165     print("Welcome to the Student Grading System")
166     name = input("Enter student name: ")
167     marks_obtained = float(input("Enter marks obtained: "))
168     total_marks = float(input("Enter total marks: "))
169     if marks_obtained < 0 or total_marks <= 0 or marks_obtained > total_marks:
170         print("Error: Invalid marks input.")
171         return
172     percentage = (marks_obtained / total_marks) * 100
173     print(f"Student Name: {name}")
174     print(f"Marks Obtained: {marks_obtained} out of {total_marks}")
175     print(f"Percentage: {percentage:.2f}%")
176     # Example usage:
177     student_grading_system()
178
Problems Output Debug Console Terminal Ports
Enter student name: kommu madhupriya
Enter marks obtained: 100
Enter total marks: 100
Student Name: kommu madhupriya
Marks Obtained: 100.0 out of 100.0
Percentage: 100.00%
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

Stage3:



The screenshot shows the VS Code editor with the file `day42.py` open. The code implements a function `calculate_grade()` that calculates the percentage and grade based on 5 subjects' marks. The requirements are: UI should accept student name and marks of 5 subjects; Each subject is out of 100; Calculate: 1. Total marks, 2. Percentage, 3. Grade. Grade rules: >= 90% -> A+, >= 75% -> A, >= 60% -> B, >= 40% -> C, < 40% -> Fail. The function uses `tkinter` for the UI and `messagebox` for displaying results and errors.

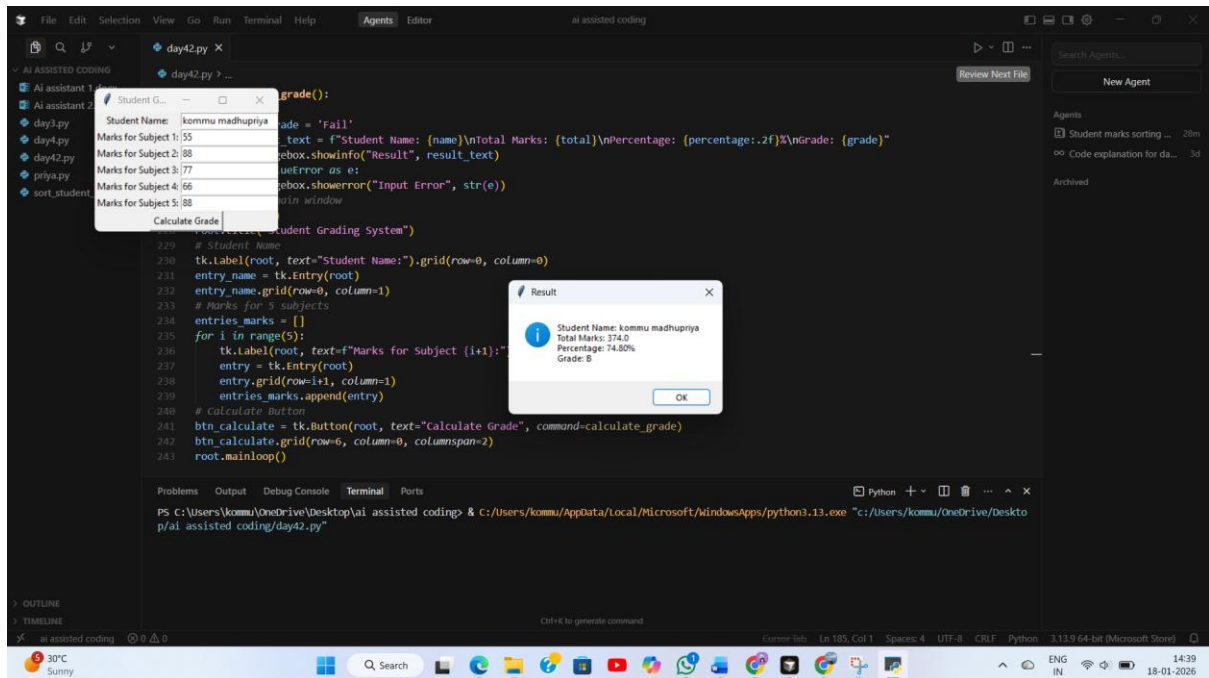
```
186 '''create a Python Tkinter-based user interface for a Student Grading System.
187 Requirements:
188 - UI should accept student name and marks of 5 subjects.
189 - Each subject is out of 100.
190 - Calculate:
191   1. Total marks
192   2. Percentage
193   3. Grade
194 Grade rules:
195 - >= 90% -> A+
196 - >= 75% -> A
197 - >= 60% -> B
198 - >= 40% -> C
199 - < 40% -> Fail
200 - Display total, percentage, and grade clearly in the UI.
201 - Show an error message if any input is invalid.'''
202 import tkinter as tk
203 from tkinter import messagebox
204 def calculate_grade():
205     try:
206         name = entry_name.get()
207         marks = [float(entry.get()) for entry in entries_marks]
208         if any(mark < 0 or mark > 100 for mark in marks):
209             raise ValueError("Marks should be between 0 and 100.")
210         total = sum(marks)
211         percentage = (total / 500) * 100
212         if percentage >= 90:
213             grade = 'A+'
214         elif percentage >= 75:
215             grade = 'A'
216         elif percentage >= 60:
217             grade = 'B'
218         elif percentage >= 40:
219             grade = 'C'
220         else:
221             grade = 'Fail'
```



The screenshot shows the VS Code editor with the file `day42.py` open. The code implements the main window for the Student Grading System. It uses `tkinter` to create a window titled "Student Grading System". It includes a label for "Student Name", an entry field for the name, and a grid of labels and entry fields for marks of 5 subjects. A "calculate" button is added, which calls the `calculate_grade` function. The window is configured with a grid layout and a main loop.

```
204 def calculate_grade():
205     grade = 'A'
206     elif percentage >= 60:
207         grade = 'B'
208     elif percentage >= 40:
209         grade = 'C'
210     else:
211         grade = 'Fail'
212     result_text = f"Student Name: {name}\nTotal Marks: {total}\nPercentage: {percentage:.2f}%\nGrade: {grade}"
213     messagebox.showinfo("Result", result_text)
214 except ValueError as e:
215     messagebox.showerror("Input Error", str(e))
216
217 # Create the main window
218 root = tk.Tk()
219 root.title("Student Grading System")
220 # Student Name
221 tk.Label(root, text="Student Name:").grid(row=0, column=0)
222 entry_name = tk.Entry(root)
223 entry_name.grid(row=0, column=1)
224 # Marks for 5 subjects
225 entries_marks = []
226 for i in range(5):
227     tk.Label(root, text=f"Marks for Subject {i+1}:").grid(row=i+1, column=0)
228     entry = tk.Entry(root)
229     entry.grid(row=i+1, column=1)
230     entries_marks.append(entry)
231 # calculate Button
232 btn_calculate = tk.Button(root, text="calculate Grade", command=calculate_grade)
233 btn_calculate.grid(row=6, column=0, columnspan=2)
234 root.mainloop()
```


Output:



Observation:

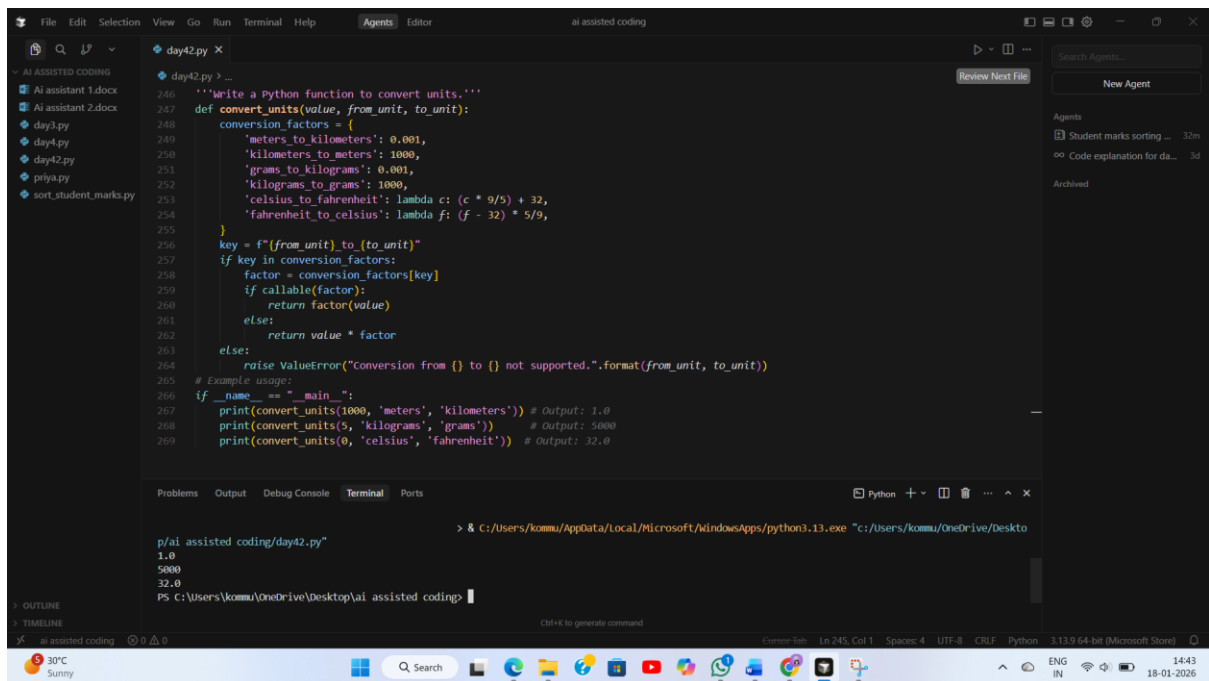
With a vague UI prompt, the AI produced only a simple or unclear interface idea. As the prompt was refined to include calculation requirements, the UI output became more meaningful. When full instructions were given (inputs, calculations, grade rules, and display), the AI generated a well-structured 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:



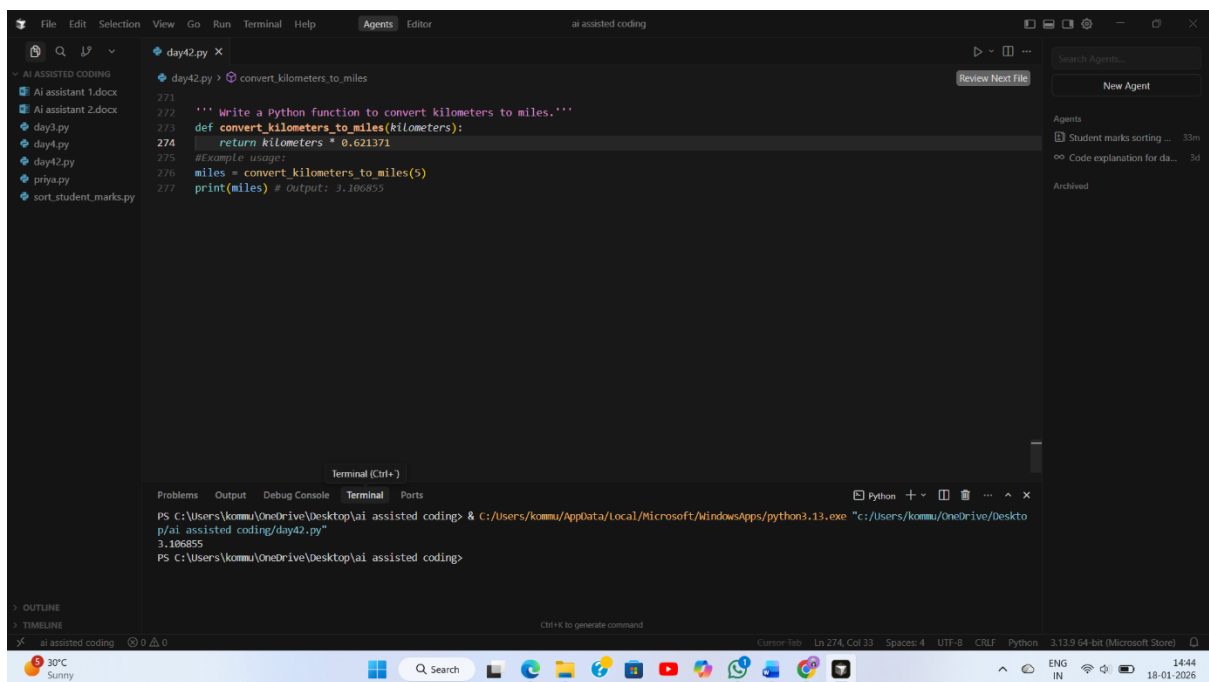
```
'''Write a Python function to convert units.'''
def convert_units(value, from_unit, to_unit):
    conversion_factors = {
        'meters_to_kilometers': 0.001,
        'kilometers_to_meters': 1000,
        'grams_to_kilograms': 0.001,
        'kilograms_to_grams': 1000,
        'celsius_to_fahrenheit': lambda c: (c * 9/5) + 32,
        'fahrenheit_to_celsius': lambda f: (f - 32) * 5/9,
    }
    key = f"{from_unit}_to_{to_unit}"
    if key in conversion_factors:
        factor = conversion_factors[key]
        if callable(factor):
            return factor(value)
        else:
            return value * factor
    else:
        raise ValueError("Conversion from {} to {} not supported.".format(from_unit, to_unit))
# Example usage:
if __name__ == "__main__":
    print(convert_units(1000, 'meters', 'kilometers')) # Output: 1.0
    print(convert_units(5, 'kilograms', 'grams')) # Output: 5000
    print(convert_units(0, 'celsius', 'fahrenheit')) # Output: 32.0
```

p/ai assisted coding/day42.py

1.0
5000
32.0

PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>

Stage2:



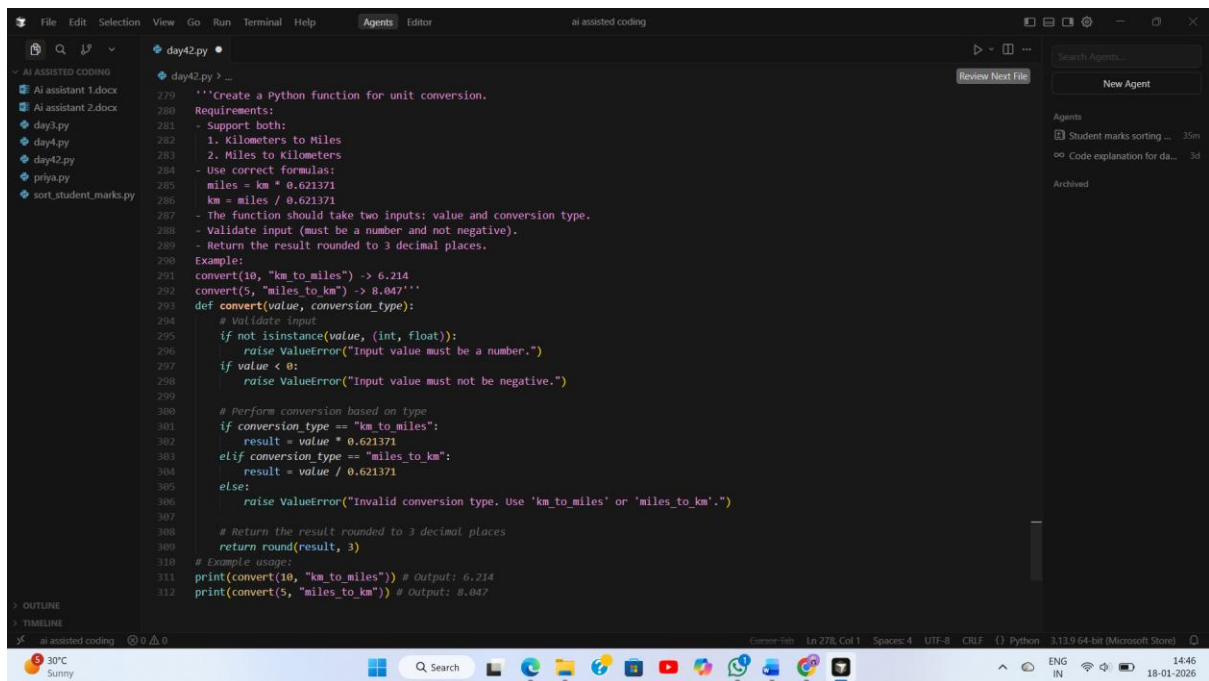
```
'''Write a Python function to convert kilometers to miles.'''
def convert_kilometers_to_miles(kilometers):
    return kilometers * 0.621371
#Example usage:
miles = convert_kilometers_to_miles(5)
print(miles) # Output: 3.106855
```

PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/kommu/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/kommu/OneDrive/Desktop/p/ai assisted coding/day42.py"

3.106855

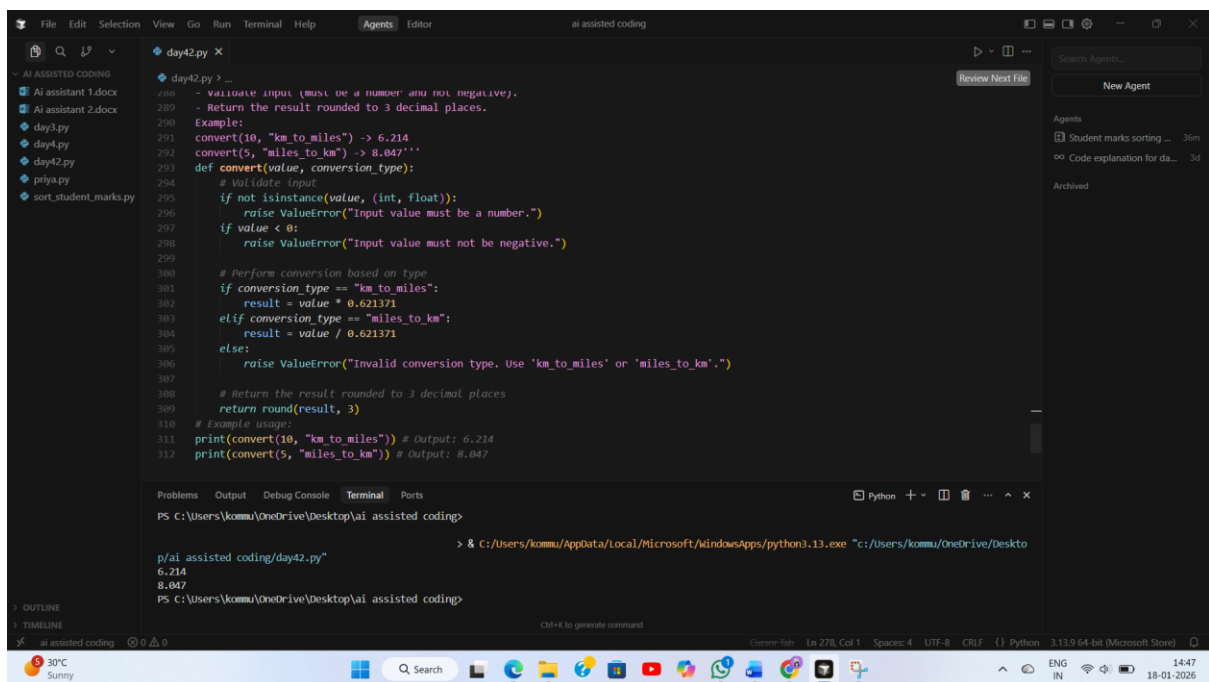
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>

Stage3:



```
279 """Create a Python function for unit conversion.
280 Requirements:
281 - Support both:
282   1. Kilometers to Miles
283   2. Miles to Kilometers
284 - Use correct formulas:
285   miles = km * 0.621371
286   km = miles / 0.621371
287 - The function should take two inputs: value and conversion type.
288 - Validate input (must be a number and not negative).
289 - Return the result rounded to 3 decimal places.
290 Example:
291 convert(10, "km_to_miles") -> 6.214
292 convert(5, "miles_to_km") -> 8.047"""
293 def convert(value, conversion_type):
294     # Validate input
295     if not isinstance(value, (int, float)):
296         raise ValueError("Input value must be a number.")
297     if value < 0:
298         raise ValueError("Input value must not be negative.")
299
300     # Perform conversion based on type
301     if conversion_type == "km_to_miles":
302         result = value * 0.621371
303     elif conversion_type == "miles_to_km":
304         result = value / 0.621371
305     else:
306         raise ValueError("Invalid conversion type. Use 'km_to_miles' or 'miles_to_km'.")
307
308     # Return the result rounded to 3 decimal places
309     return round(result, 3)
310
311 # Example usage:
312 print(convert(10, "km_to_miles")) # Output: 6.214
313 print(convert(5, "miles_to_km")) # Output: 8.047
```

Output:



```
288 - Validate input (must be a number and not negative).
289 - Return the result rounded to 3 decimal places.
290 Example:
291 convert(10, "km_to_miles") -> 6.214
292 convert(5, "miles_to_km") -> 8.047"""
293 def convert(value, conversion_type):
294     # Validate input
295     if not isinstance(value, (int, float)):
296         raise ValueError("Input value must be a number.")
297     if value < 0:
298         raise ValueError("Input value must not be negative.")
299
300     # Perform conversion based on type
301     if conversion_type == "km_to_miles":
302         result = value * 0.621371
303     elif conversion_type == "miles_to_km":
304         result = value / 0.621371
305     else:
306         raise ValueError("Invalid conversion type. Use 'km_to_miles' or 'miles_to_km'.")
307
308     # Return the result rounded to 3 decimal places
309     return round(result, 3)
310
311 # Example usage:
312 print(convert(10, "km_to_miles")) # Output: 6.214
313 print(convert(5, "miles_to_km")) # Output: 8.047
```

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
> & C:/Users/kommu/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/kommu/OneDrive/Desкто
6.214
8.047
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

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.