

# AI Assisted coding Assignment-3.2

Name: Kavati Chaitanya

Ht.no: 2303A51677

Batch. No: 22

---

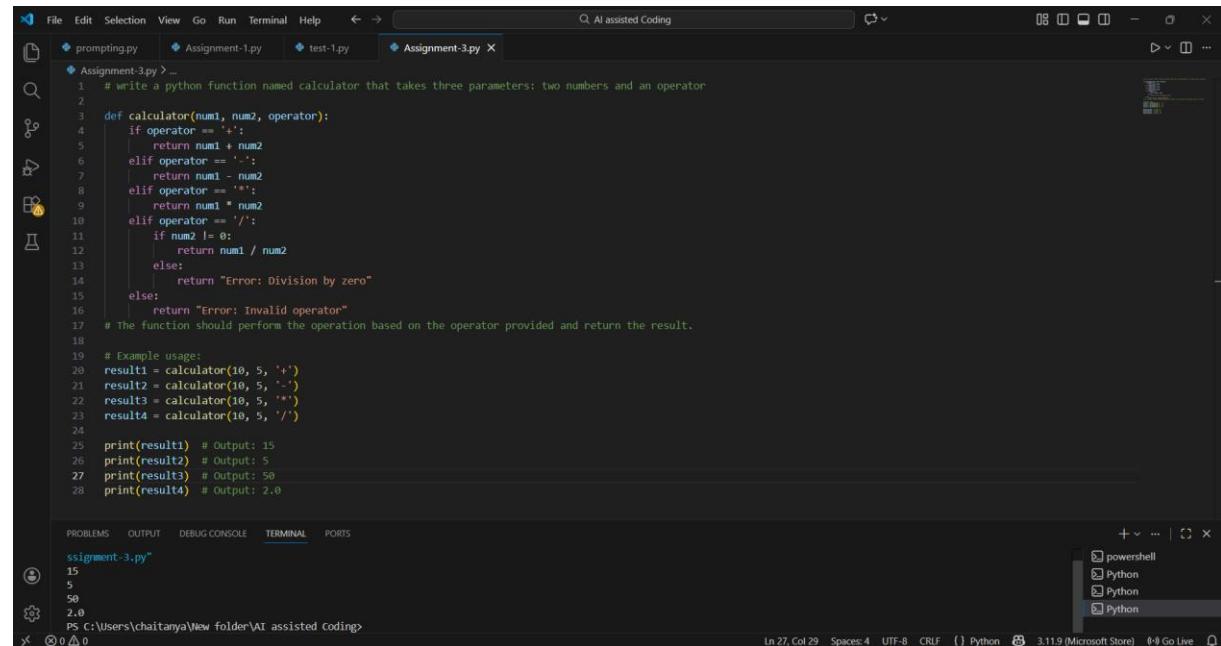
## Task Description

- 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.

### Prompt:

write a python function named calculator that takes three parameters: two numbers and an operator

### Code:



```
File Edit Selection View Go Run Terminal Help ← → C:\AI assisted Coding

Assignment-3.py > ...
prompting.py Assignment-1.py test-1.py Assignment-3.py ×

Assignment-3.py
1 # write a python function named calculator that takes three parameters: two numbers and an operator
2
3 def calculator(num1, num2, operator):
4     if operator == '+':
5         return num1 + num2
6     elif operator == '-':
7         return num1 - num2
8     elif operator == '**':
9         return num1 * num2
10    elif operator == '/':
11        if num2 != 0:
12            return num1 / num2
13        else:
14            return "Error: Division by zero"
15    else:
16        return "Error: Invalid operator"
17 # The function should perform the operation based on the operator provided and return the result.
18
19 # Example usage:
20 result1 = calculator(10, 5, '+')
21 result2 = calculator(10, 5, '-')
22 result3 = calculator(10, 5, '**')
23 result4 = calculator(10, 5, '/')
24
25 print(result1) # Output: 15
26 print(result2) # Output: 5
27 print(result3) # Output: 50
28 print(result4) # Output: 2.0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Assignment-3.py*
15
5
50
2.0
PS C:\Users\chaitanya\New folder\AI assisted Coding>
In 27, Col 29 Spaces: 4 UTF-8 CRLF Python 3.11.9 (Microsoft Store) ⓘ Go Live
```

**Output-1:**

**15**

**5**

**50**

**2.0**

**Prompt used-2:**

write a python function named calculator that takes three parameters: two numbers and an operator (as a string: '+', '-', '\*', '/').

requirements:

Take 2 numbers and operator as input

Handle division by zero

Handle invalid operator inputs

example input: 10, 5, "+" output:15

## Code-2:

The screenshot shows the VS Code interface with the following details:

- Editor:** The main editor window displays the Python file `Assignment-3.py`. The code defines a `calculator` function that takes three parameters: two numbers and an operator. It performs arithmetic operations based on the operator and handles division by zero and invalid operators.
- Terminal:** Below the editor, the terminal tab is active, showing the following session:

```
2.0
Enter the first number: 24
Enter the second number: 2
Enter the operator (+, -, *, /): /
The result is: 12.0
PS C:\Users\chaitanya\New folder\AI assisted Coding>
```
- Output:** The output tab shows the result of the calculation: `12.0`.
- Debug Console:** The debug console tab shows the following session:

```
powershell
Python
Python
Python
```
- Bottom Status Bar:** The status bar indicates the current line (Ln 50), column (Col 32), spaces (Spaces: 4), encoding (UTF-8), and file type (CRLF). It also shows the Python extension version (3.11.9) and the Go Live button.

## Output:

The terminal window shows the following session:

```
2.0
Enter the first number: 24
Enter the second number: 2
Enter the operator (+, -, *, /): /
The result is: 12.0
PS C:\Users\chaitanya\New folder\AI assisted Coding>
```

Aspect	Initial Calculator	Improvised Calculator
Suitability	Works fine for simple calculations	Better suited for real-world, user-facing use
Readability	Easy to understand but minimal clarity	Cleaner and easier to read with comments
Input Handling	Assumes inputs are already numbers	Safely converts and validates user input
User Feedback	Very limited error messages	Clear and helpful error messages
Code Structure	Basic logic inside the function	Well-structured with proper validation
Robustness	Can break with invalid input	More stable and user-friendly
Error Handling	Handles only basic errors	Covers invalid input, operators, and division by zero

## **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.

## Prompt:

# Write a program to sort student marks

## **Code-1:**

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "File Edit Selection View Go Run Terminal Help < > Q AI assisted Coding". The left sidebar has icons for file operations like Open, Save, Find, and Run. The main editor area contains Python code for sorting student marks. The code includes a docstring, a function definition for sorting marks, and several print statements demonstrating its usage. The status bar at the bottom shows "In 68 Col 99 Spaces 4 UTF-8 CR LF () Python" and "3.11.9 (Microsoft Store)".

```
prompting.py Assignment-1.py test-1.py Assignment-3.py

Assignment-3.py > ...
52 # Write a program to sort student marks in ascending order using a function with edge cases
53
54 def sort_marks(marks):
55     if not marks:
56         return "Error: The list of marks is empty"
57     for mark in marks:
58         if not isinstance(mark, (int, float)):
59             return "Error: All marks must be numbers"
60     return sorted(marks)
61 # Example usage
62 marks_list = [85, 92, 78, 98, 88]
63 sorted_marks = sort_marks(marks_list)
64 print("Sorted marks:", sorted_marks) # Output: [78, 85, 88, 90, 92] 5
65 # Edge case: empty list
66 empty_marks_list = []
67 sorted_empty_marks = sort_marks(empty_marks_list)
68 print("Sorted empty marks list:", sorted_empty_marks) # output: Error: The list of marks is empty
```

## Output:

The result is: 12.0

PS C:\Users\chaitanya\New folder\AI assisted Coding> & c:/Users/chaitanya\Assignment-3.py"

Sorted marks: [78, 85, 88, 90, 92]

Sorted empty marks list: Error: The list of marks is empty

PS C:\Users\chaitanya\New folder\AI assisted Coding> [

0 0 △ 0

A screenshot of a code editor window titled "Assignment-3.py". The code is as follows:

```
File Edit Selection View Go Run Terminal Help ← → C:\Users\chaitanya\New folder\AI assisted Coding
```

```
prompting.py Assignment-1.py test1.py Assignment-3.py X
```

```
Assignment-3.py > sort_marks
```

```
69
70 # Write a Python function that takes a list of student marks as input and sorts them in ascending order.
71 # Handle duplicate marks correctly.
72 # use an efficient sorting approach.
73 # Output sorted list clearly.
74 def sort_marks(marks):
75     if not marks:
76         return "Error: The list of marks is empty"
77     for mark in marks:
78         if not isinstance(mark, (int, float)):
79             return "Error: All marks must be numbers"
80     return sorted(marks)
81 # Example usage:
82 marks_list = [85, 92, 78, 90, 88, 85, 92]
83 sorted_marks = sort_marks(marks_list)
84 print("Sorted marks with duplicates:", sorted_marks) # Output: [78, 85, 85, 88, 90, 92, 92]
85
```

At the bottom of the code editor, there is a terminal window showing the command and its output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\chaitanya\New folder\AI assisted Coding> & c:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/chaitanya/New folder/AI assisted Coding/AI assignment-3.py"
Assignment-3.py
Sorted marks: [78, 85, 88, 90, 92]
Sorted empty marks list: Error: The list of marks is empty
Sorted marks with duplicates: [78, 85, 85, 88, 90, 92, 92]
PS C:\Users\chaitanya\New folder\AI assisted Coding> [
```

The terminal also shows a dropdown menu for "Python" options.

## **Output-2:**

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    FORKS

ssignment-3.py"

Sorted marks: [78, 85, 88, 90, 92]

Sorted empty marks list: Error: The list of marks is empty

Sorted marks with duplicates: [78, 85, 85, 88, 90, 92, 92]

Sorted invalid marks list: Error: All marks must be numbers

PS C:\Users\chaitanya\New folder\AI assisted Coding> □

## **Justification:**

By clearly specifying the sorting order and input constraints, the refined prompt improved the correctness and organization of the AI-generated logic. It guided the AI to better handle numeric values, duplicates, and edge cases such as empty inputs. This demonstrates that clearer prompts result in more reliable and efficient AI-generated solutions.

# 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.

## Prompt:

```
# Write a python function that checks whether a given number is prime or prime.
```

## Code:

The screenshot shows a code editor window with several tabs at the top: 'prompting.py', 'Assignment-1.py', 'test-1.py', and 'Assignment-3.py'. The 'Assignment-3.py' tab is active, displaying the following Python code:

```
80
81     # Write a python function that checks whether a given number is prime or prime not and take user input to call the function and display the result.
82
83     def is_prime(number):
84         if number <= 1:
85             return False
86         for i in range(2, int(number**0.5) + 1):
87             if number % i == 0:
88                 return False
89         return True
90
91     # Take user input
92     num = int(input("Enter a number to check if it is prime: "))
93     if is_prime(num):
94         print(f"{num} is a prime number.")
95     else:
96         print(f"{num} is not a prime number.")
97
98
99
100
101
102
103
104
```

Below the code editor, there is a 'PROBLEMS' panel showing an error message: 'Sorted invalid marks list: Error: All marks must be numbers'. The 'OUTPUT' panel shows the terminal output of running the script with the number 89 as input, resulting in the message '89 is a prime number.' The status bar at the bottom indicates the file is 'Assignment-3.py', line 104, column 1, and the Python version is 3.11.9 (Microsoft Store).

## OUTPUT:

The screenshot shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Sorted invalid marks list: Error: All marks must be numbers
PS C:\Users\chaitanya\New folder\AI assisted Coding> & C:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:\Users\chaitanya\New folder\AI assisted Coding\Assignment-3.py"
Enter a number to check if it is prime: 89
89 is a prime number.
PS C:\Users\chaitanya\New folder\AI assisted Coding> 0 △ 0
Ln 104
```

## **Justification:**

By using few-shot prompting, the AI learns prime-checking behavior from example inputs rather than assumptions.

Seeing both valid primes and non-primes helps it avoid common mistakes with values like 0, 1, and negative numbers.

This leads to a prime-checking function that is more correct, consistent, and dependable overall.

## **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.

### **Prompt Used:**

Design a simple and user-friendly student grading system UI

### Requirements:

1. Create input fields to enter marks for multiple subjects.
2. Calculate total marks and percentage based on the entered values.
3. Assign grades based on percentage (e.g., A, B, C, Fail).
4. Display total marks, percentage, and grade clearly on the screen.
5. Use a clean and well-structured layout.
6. Validate inputs to ensure only numeric values are accepted.
7. Use HTML and CSS for the user interface and JavaScript for calculations. create all files in one file

### **Code:**

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "AI assisted Coding". The left sidebar shows icons for files: prompting.py, Assignment-1.py, test-1.py, Assignment-3.py, and Untitled-1. The main editor area contains the following code:

```
1 # Design a simple and user-friendly student grading system.
2 # Requirements:
3 # 1. Create input fields to enter marks for multiple subjects.
4 # 2. Calculate total marks and percentage based on the entered values.
5 # 3. Assign grades based on percentage (e.g., A, B, C, Fail).
6 # 4. Display total marks, percentage, and grade clearly on the screen.
7 # 5. Use a clean and well-structured layout.
8 # 6. Validate inputs to ensure only numeric values are accepted.
9 <html lang="en">
10    <head>
11        <meta charset="UTF-8">
12        <meta name="viewport" content="width=device-width, initial-scale=1.0">
13        <title>Student Grading System</title>
14        <style>
15            body {
16                font-family: Arial, sans-serif;
17                margin: 20px;
18                padding: 20px;
19                background-color: #f4f4f4;
20            }
21            .container {
22                max-width: 500px;
23                margin: auto;
24                background: # white;
25                padding: 20px;
26                border-radius: 8px;
27                box-shadow: 0 0 10px #rgba(0, 0, 0, 0.1);
28            }
29            input[type="number"] {
30                width: 100%;
```

The terminal at the bottom shows:

```
Sorted invalid marks list: Error: All marks must be numbers
PS C:\Users\chaitanya\New folder\AI assisted Coding & C:/Users/chaitanya/AppData/Local/Microsoft/Windows/apps/python3.11.exe "c:/Users/chaitanya/New folder/AI assisted Coding/AI assignment-3.py"
Enter a number to check if it is prime: 89
89 is a prime number.
PS C:\Users\chaitanya\New folder\AI assisted Coding> []
```

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "AI assisted Coding". The left sidebar shows icons for files: prompting.py, Assignment-1.py, test-1.py, Assignment-3.py, and Untitled-1. The main editor area contains the following CSS code:

```
9     <html lang="en">
11         <head>
15             <style>
30                 input[type="number"] {
31                     width: 100%;
32                     padding: 10px;
33                     margin: 10px 0;
34                     border: 1px solid #ccc;
35                     border-radius: 4px;
36                 }
37                 button {
38                     padding: 10px 15px;
39                     background-color: #28a745;
40                     color: #white;
41                     border: none;
42                     border-radius: 4px;
43                     cursor: pointer;
44                 }
45                 button:hover {
46                     background-color: #218838;
47                 }
48                 .result {
49                     margin-top: 20px;
50                 }
51             </style>
52         </head>
53         <body>
54             <div class="container">
55                 <h2>Student Grading System</h2>
56                 <form id="gradingForm">
57                     <label for="subject1">Subject 1 Marks:</label>
```

# Student Grading System

Enter Marks for 5 Subjects:

Calculate Grade

Total Marks: 420 / 500

Percentage: 84.00%

Grade: B

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons. A search bar at the top right says "AI assisted Coding". The left sidebar has sections for RUN AND DEBUG, RUN, and a "Run and Debug" button which is highlighted in blue. Below these are links to other files: prompting.py, Assignment-1.py, test-1.py, Assignment-3.py, and student.html. The main workspace displays the content of student.html:

```
<html lang="en">
<body>
<script>
function calculateGrade() {
    const subjects = [
        parseFloat(document.getElementById('subject1').value) || 0,
        parseFloat(document.getElementById('subject2').value) || 0,
        parseFloat(document.getElementById('subject3').value) || 0,
        parseFloat(document.getElementById('subject4').value) || 0,
        parseFloat(document.getElementById('subject5').value) || 0
    ];

    const totalMarks = subjects.reduce((a, b) => a + b, 0);
    const percentage = (totalMarks / (subjects.length * 100)) * 100;
    let grade;

    if (percentage >= 90) {
        grade = 'A';
    } else if (percentage >= 75) {
        grade = 'B';
    } else if (percentage >= 50) {
        grade = 'C';
    } else {
        grade = 'Fail';
    }

    document.getElementById('result').innerHTML =
        `<h3>Results:</h3>
        <p>Total Marks: ${totalMarks}</p>
        <p>Percentage: ${percentage.toFixed(2)}%</p>
        <p>Grade: ${grade}</p>`;
}
</script>
</body>
</html>
```

At the bottom of the editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The DEBUG CONSOLE tab is currently selected.

**Output:**

# Student Grading System

Enter Marks for 5 Subjects:

**Calculate Grade**

Total Marks: 125 / 500

Percentage: 25.00%

Grade: Fail

## Justification:

A well-defined prompt helps the UI accurately calculate total marks, percentage, and grades from user input.

Proper input validation increases reliability by effectively managing missing or invalid values.

A structured and organized UI layout improves readability and delivers clear results for both students and evaluators.

## Task Description-5

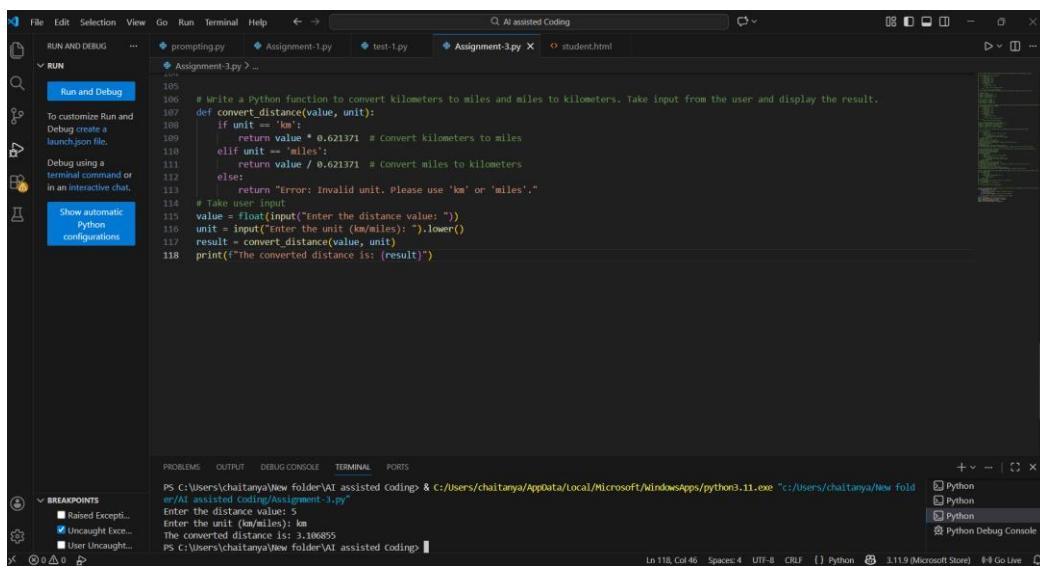
- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.

### Prompt used-1:

Write a Python function to convert kilometers to miles and miles to kilometers.

Take input from the user and display the result.

### Code-1:



The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows files: prompting.py, Assignment-1.py, test-1.py, Assignment-3.py (the active file), and studenthtml.
- Run and Debug View:** Shows a "Run and Debug" button and a tooltip: "To customize Run and Debug create a launch.json file. Debug using a terminal command or in an interactive chat." It also has a "Show automatic Python configurations" button.
- Code Editor:** Displays the Python code for Assignment-3.py. The code defines a function convert\_distance that takes value and unit as parameters. It checks if unit is 'km' or 'miles' and performs the conversion. If unit is invalid, it returns an error message. It then prompts the user for distance and unit, calls the function, and prints the result.
- Terminal:** Shows the command PS C:\Users\chaitanya\New Folder\AI assisted Coding & C:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/chaitanya/New folder/AI assisted Coding/Assignment-3.py". The user enters "Enter the distance value: 5" and "Enter the unit (km/miles): km". The output shows "The converted distance is: 3.106855".
- Bottom Status Bar:** Shows "Ln 118, Col 46" and other system information.

### Output-1:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\chaitanya\New folder\AI assisted Coding> & C:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/chaitanya/New folder/AI assisted Coding/Assignment-3.py"
Enter the distance value: 5
Enter the unit (km/miles): km
The converted distance is: 3.106855
PS C:\Users\chaitanya\New folder\AI assisted Coding>

```

## Prompt used-2:

Write a Python program using functions to convert:

- Kilometers to miles
- Miles to kilometers

Requirements:

- Take numeric input from the user.
- Allow the user to choose the conversion type.
- Use correct conversion formulas.
- Validate input and handle invalid values.
- Display clear and formatted output.

## Code-2:

```

File Edit Selection View Go Run Terminal Help ⏎ → Q: AI assisted Coding
RUN AND DEBUG ... prompting.py Assignment-1.py test-1.py Assignment-3.py student.html
RUN
Run and Debug
To customize Run and
Debug create a
launch.json file.
Debug using a
terminal command or
in an interactive chat.
Show automatic
Python
configurations
120 # Write a Python program using functions to convert:
121 # - Kilometers to miles
122 # - Miles to Kilometers Requirements:
123 # - Take numeric input from the user,
124 # - Allow the user to choose the conversion type.
125 # - Use correct conversion formulas.
126 # - Validate input and handle invalid values.
127 # - Display clear and formatted output.
128
129 def convert_distance(value, unit):
130     if unit == 'km':
131         return value * 0.621371 # convert kilometers to miles
132     elif unit == 'miles':
133         return value / 0.621371 # convert miles to kilometers
134     else:
135         return "Error: Invalid unit. Please use 'km' or 'miles'."
136     # Take user input
137     value = float(input("Enter the distance value: "))
138     unit = input("Enter the unit (km/miles): ").lower()
139     result = convert_distance(value, unit)
140     print(f"The converted distance is: {result}")
141

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

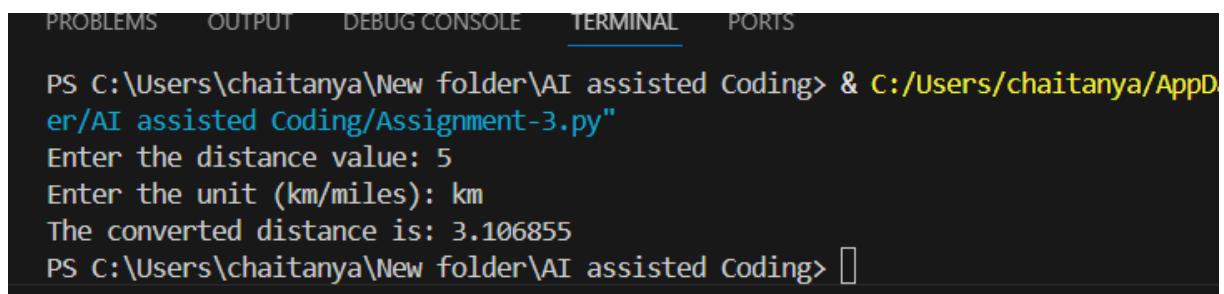
```

PS C:\Users\chaitanya\New folder\AI assisted Coding> & C:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/chaitanya/New folder/AI assisted Coding/Assignment-3.py"
Enter the distance value: 5
Enter the unit (km/miles): km
The converted distance is: 3.106855
PS C:\Users\chaitanya\New folder\AI assisted Coding>

```

In 141, Col 1 Spaces: 4 UFT-8 CRLF { } Python 3.11.9 (Microsoft Store) ⇧ Go Live

## Output-2:



A screenshot of a terminal window from a code editor. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal output shows the following:

```
PS C:\Users\chaitanya\New folder\AI assisted Coding> & C:/Users/chaitanya/AppD  
er/AI assisted Coding/Assignment-3.py"  
Enter the distance value: 5  
Enter the unit (km/miles): km  
The converted distance is: 3.106855  
PS C:\Users\chaitanya\New folder\AI assisted Coding> □
```

## Justification:

In the initial prompt, the unit conversion logic gives correct results but allows negative distance values due to weak validation.

With a more detailed prompt, the program improves by enforcing input checks and presenting clearer, well-formatted output.

Overall, greater prompt specificity results in more accurate, robust, and user-friendly unit conversion code

