

# AI Assisted coding Assignment-3.2

**Name:** Kavati Chaitanya

**Ht.no:** 2303A51677

**Batch. No:** 22

---

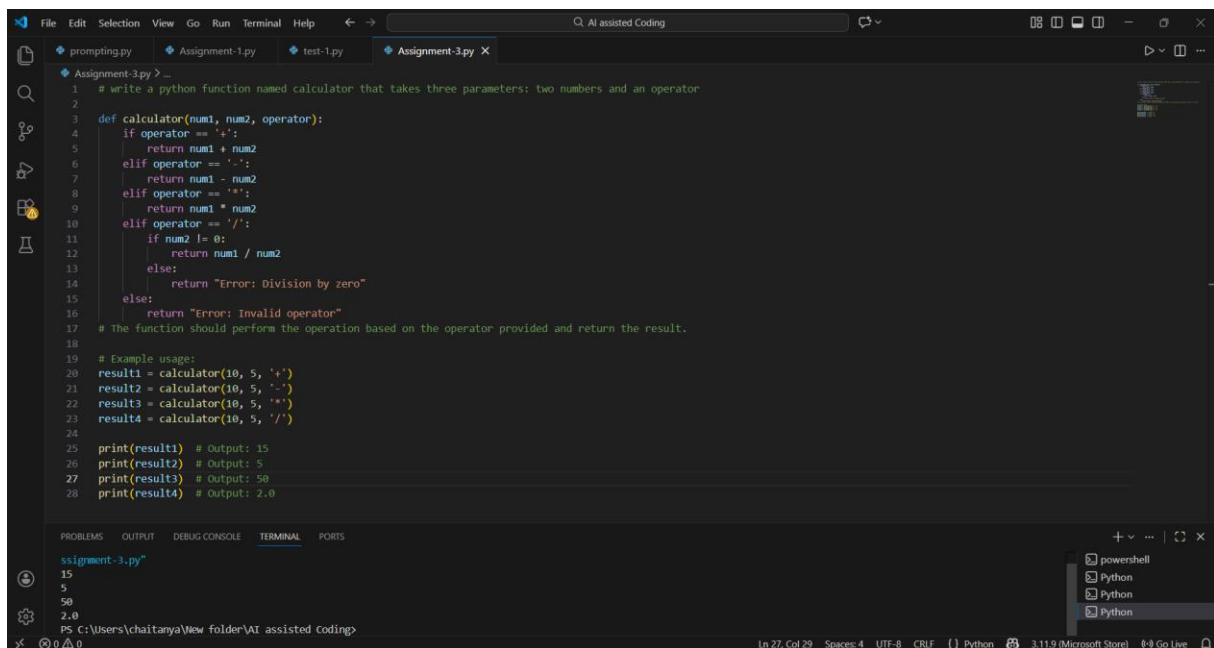
## Task Description

- 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
Assignment-3.py X
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>

Ln 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:

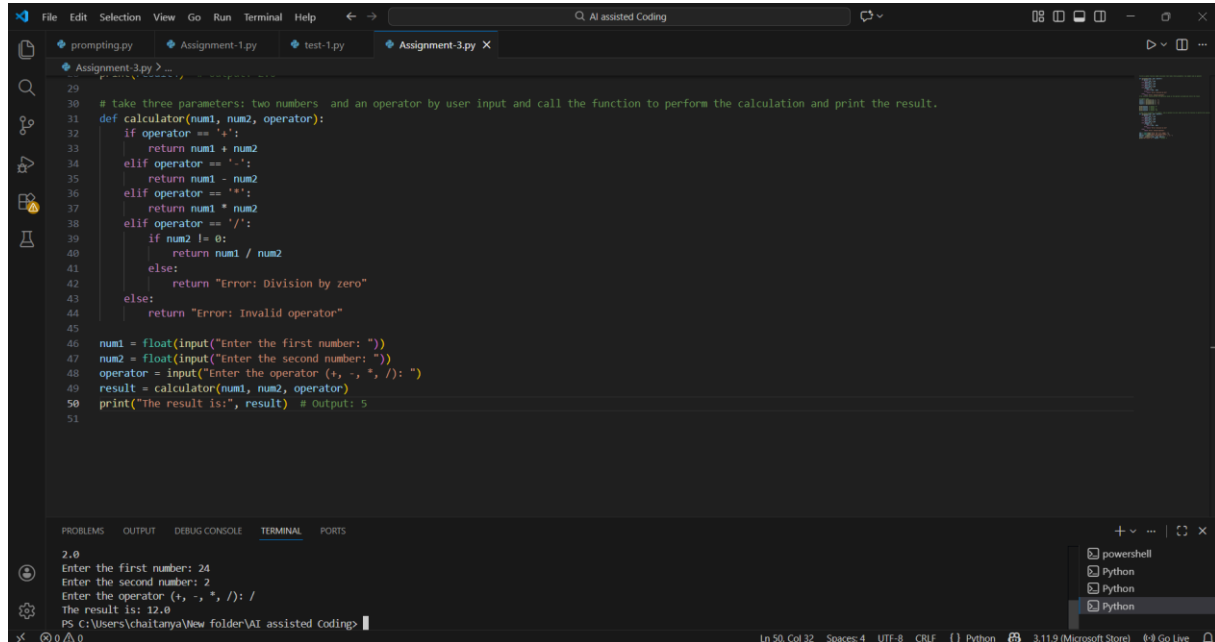
Take 2 numbers and operator as input

Handle division by zero

Handle invalid operator inputs

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

## Code-2:

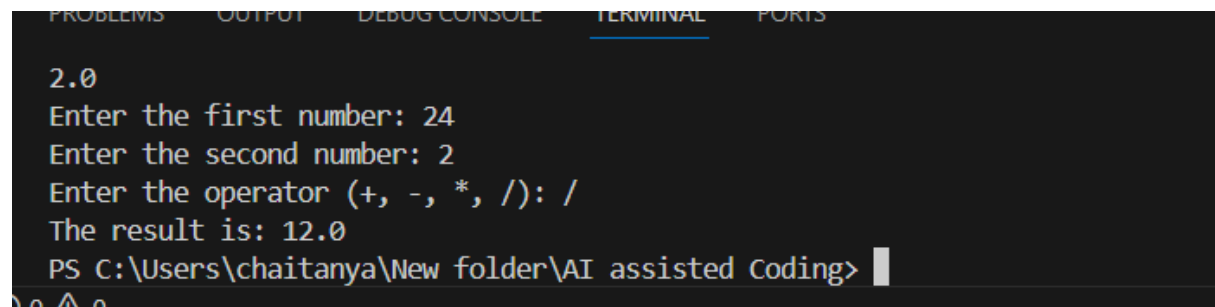


The screenshot shows a Visual Studio Code editor with a Python file named 'Assignment-3.py'. The code defines a 'calculator' function that takes two numbers and an operator as input. It uses conditional logic to perform addition, subtraction, multiplication, division, or return an error message for invalid operators or division by zero. The main part of the code prompts the user for input and prints the result. The terminal window at the bottom shows the execution of the program with inputs 24, 2, and '/', resulting in 'The result is: 12.0'.

```
29
30 # take three parameters: two numbers and an operator by user input and call the function to perform the calculation and print the result.
31 def calculator(num1, num2, operator):
32     if operator == '+':
33         return num1 + num2
34     elif operator == '-':
35         return num1 - num2
36     elif operator == '*':
37         return num1 * num2
38     elif operator == '/':
39         if num2 != 0:
40             return num1 / num2
41         else:
42             return "Error: Division by zero"
43     else:
44         return "Error: Invalid operator"
45
46 num1 = float(input("Enter the first number: "))
47 num2 = float(input("Enter the second number: "))
48 operator = input("Enter the operator (+, -, *, /): ")
49 result = calculator(num1, num2, operator)
50 print("The result is:", result) # Output: 5
51
```

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:



This is a close-up of the terminal window from the previous image, showing the user's input and the program's output.

```
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:

```

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
62 # Example usage:
63 marks_list = [85, 92, 78, 90, 88]
64 sorted_marks = sort_marks(marks_list)
65 print("Sorted marks:", sorted_marks) # Output: [78, 85, 88, 90, 92] 5
66 # Edge cases: empty list
67 empty_marks_list = []
68 sorted_empty_marks = sort_marks(empty_marks_list)
69 print("Sorted empty marks list:", sorted_empty_marks) # Output: Error: The list of marks is empty

```

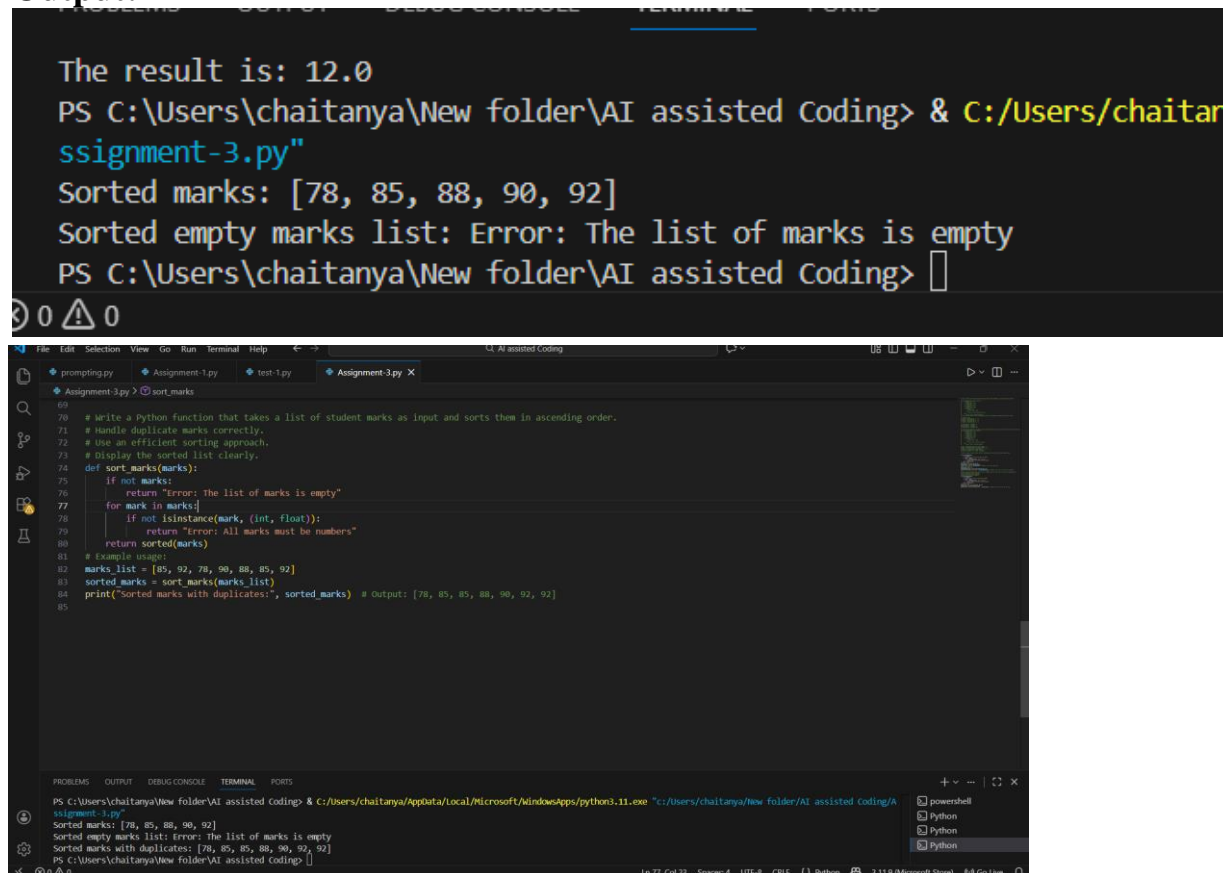
Terminal Output:

```

The result is: 12.8
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"
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>

```

## Output:

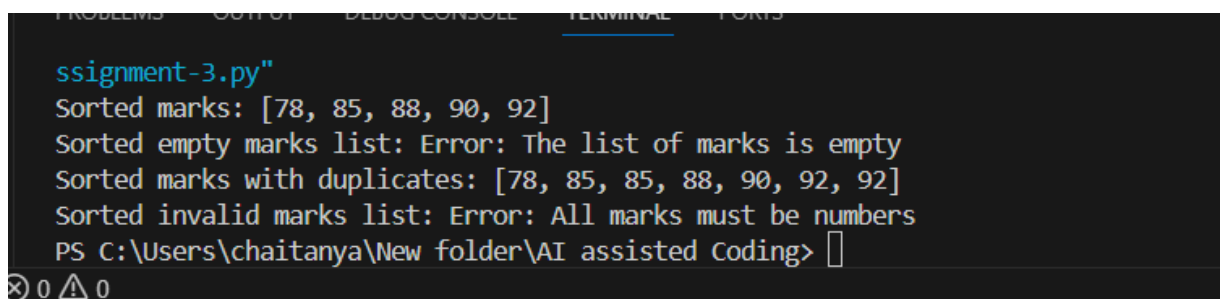


The screenshot displays a terminal window at the top and a code editor at the bottom. The terminal shows the execution of a Python script named 'Assignment-3.py'. The output of the script is as follows:

```
The result is: 12.0
PS C:\Users\chaitanya\New folder\AI assisted Coding> & C:/Users/chaitanya/AppData/Local/Microsoft/WindowsApps/python11.exe "C:/Users/chaitanya/New folder\AI assisted Coding\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> 
```

The code editor below shows the source code of 'Assignment-3.py'. The code defines a function 'sort\_marks' that takes a list of marks and returns a sorted list. It includes comments for each step: writing the function, handling duplicates, using an efficient sorting approach, and displaying the sorted list. The function also handles edge cases like an empty list or non-numeric values. The main part of the script defines a 'marks\_list' with duplicates, calls 'sort\_marks', and prints the result. It also includes a comment for the output: '# Output: [78, 85, 85, 88, 90, 92, 92]'.

## Output-2:



The screenshot shows a terminal window with the following output:

```
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]
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.

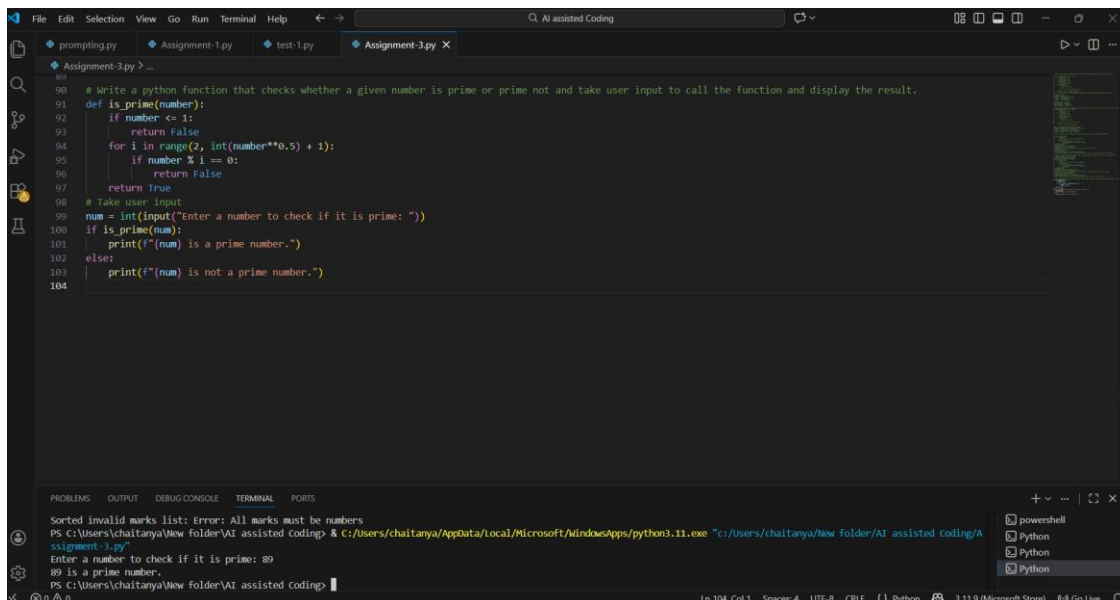
## Task Description-3

- 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.

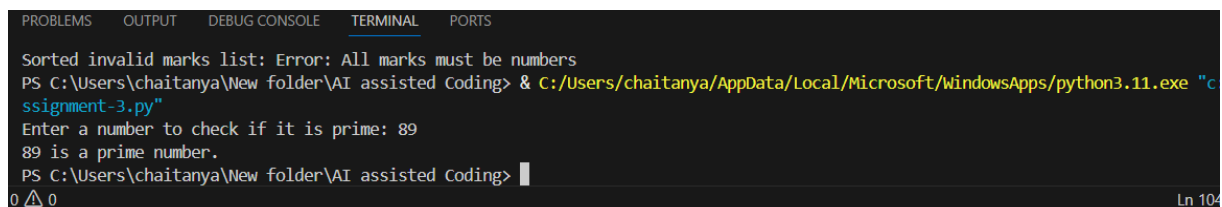
## Code:



```
90 # 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.
91 def is_prime(number):
92     if number <= 1:
93         return False
94     for i in range(2, int(number**0.5) + 1):
95         if number % i == 0:
96             return False
97     return True
98 # Take user input
99 num = int(input("Enter a number to check if it is prime: "))
100 if is_prime(num):
101     print(f"{num} is a prime number.")
102 else:
103     print(f"{num} is not a prime number.")
104
```

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>

## OUTPUT:



```
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>
```

## **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:**



```
File Edit Selection View Go Run Terminal Help
AI assisted Coding

prompting.py Assignment-1.py test-1.py Assignment-3.py # Design a simple and user-friendly stud Untitled-1

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
11 <head>
12   <meta charset="UTF-8">
13   <meta name="viewport" content="width=device-width, initial-scale=1.0">
14   <title>Student Grading System</title>
15   <style>
16     body {
17       font-family: Arial, sans-serif;
18       margin: 20px;
19       padding: 20px;
20       background-color: #f4f4f4;
21     }
22     .container {
23       max-width: 500px;
24       margin: auto;
25       background: #fff;
26       padding: 20px;
27       border-radius: 8px;
28       box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
29     }
30     input[type="number"] {
31       width: 100%;
32     }
33   </style>
34 </head>
35 <body>
36   <div class="container">
37     <h2>Student Grading System</h2>
38     <form id="gradingForm">
39       <label for="subject1">Subject 1 Marks:</label>
40       <input type="number" value="0">
41       <button type="button" value="Calculate">Calculate</button>
42     </form>
43     <div class="result">
44       <div>Total Marks: 0</div>
45       <div>Percentage: 0%</div>
46       <div>Grade: <div></div>
47     </div>
48   </div>
49 </body>
50 </html>
```

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/A
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>
```

Ln 111, Col 8 Spaces: 4 UTF-8 CRLF HTML Go Live

```
prompting.py Assignment-1.py test-1.py Assignment-3.py # Design a simple and user-friendly stud Untitled-1

9 <html lang="en">
10 <head>
11   <style>
12     input[type="number"] {
13       width: 100%;
14       padding: 10px;
15       margin: 10px 0;
16       border: 1px solid #ccc;
17       border-radius: 4px;
18     }
19     button {
20       padding: 10px 15px;
21       background-color: #28a745;
22       color: #fff;
23       border: none;
24       border-radius: 4px;
25       cursor: pointer;
26     }
27     button:hover {
28       background-color: #218838;
29     }
30     .result {
31       margin-top: 20px;
32     }
33   </style>
34 </head>
35 <body>
36   <div class="container">
37     <h2>Student Grading System</h2>
38     <form id="gradingForm">
39       <label for="subject1">Subject 1 Marks:</label>
```

# Student Grading System

Enter Marks for 5 Subjects:

95

85

90

80

70

Calculate Grade

Total Marks: 420 / 500

Percentage: 84.00%

Grade: B

```
1  <html lang="en">
45 </body>
69
70
71   <script>
72     function calculateGrade() {
73       const subjects = [
74         parseFloat(document.getElementById('subject1').value) || 0,
75         parseFloat(document.getElementById('subject2').value) || 0,
76         parseFloat(document.getElementById('subject3').value) || 0,
77         parseFloat(document.getElementById('subject4').value) || 0,
78         parseFloat(document.getElementById('subject5').value) || 0
79       ];
80
81       const totalMarks = subjects.reduce((a, b) => a + b, 0);
82       const percentage = (totalMarks / (subjects.length * 100)) * 100;
83       let grade;
84
85       if (percentage >= 90) {
86         grade = 'A';
87       } else if (percentage >= 75) {
88         grade = 'B';
89       } else if (percentage >= 50) {
90         grade = 'C';
91       } else {
92         grade = 'Fail';
93       }
94
95       document.getElementById('result').innerHTML = `
96         <h3>Results:</h3>
97         <p>Total Marks: ${totalMarks}</p>
98         <p>Percentage: ${percentage.toFixed(2)}%</p>
99         <p>Grade: ${grade}</p>
100       `;
101     }
102   </script>
103 </body>
104 </html>
```

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

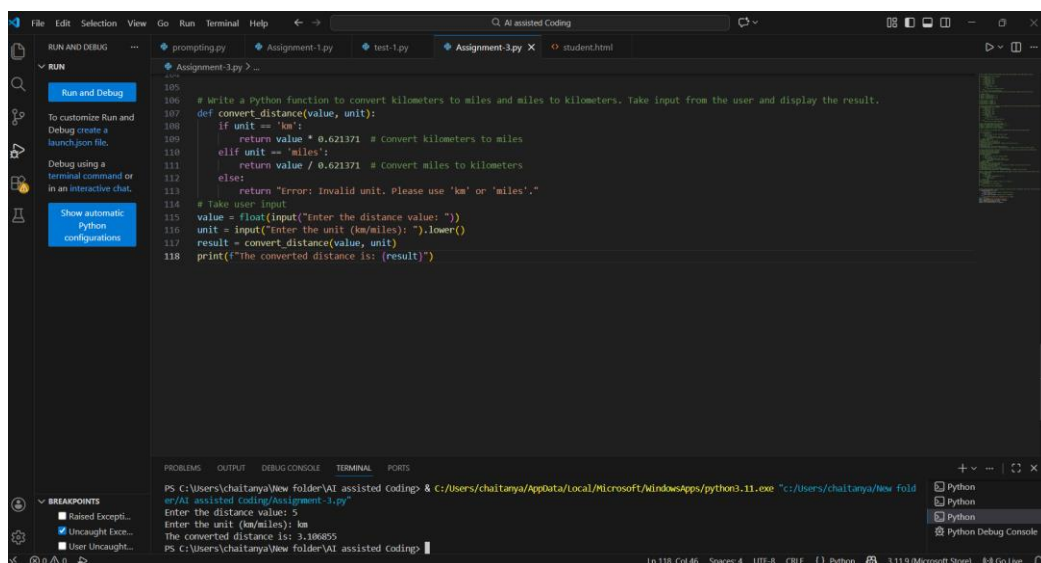
- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.

### 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:



```
105
106 # write a Python function to convert kilometers to miles and miles to kilometers. Take input from the user and display the result.
107 def convert_distance(value, unit):
108     if unit == 'km':
109         return value * 0.621371 # Convert kilometers to miles
110     elif unit == 'miles':
111         return value / 0.621371 # Convert miles to kilometers
112     else:
113         return "Error: Invalid unit. Please use 'km' or 'miles'."
114
115 # Take user input
116 value = float(input("Enter the distance value: "))
117 unit = input("Enter the unit (km/miles): ").lower()
118 result = convert_distance(value, unit)
119 print("The converted distance is: (result)")
```

### 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:

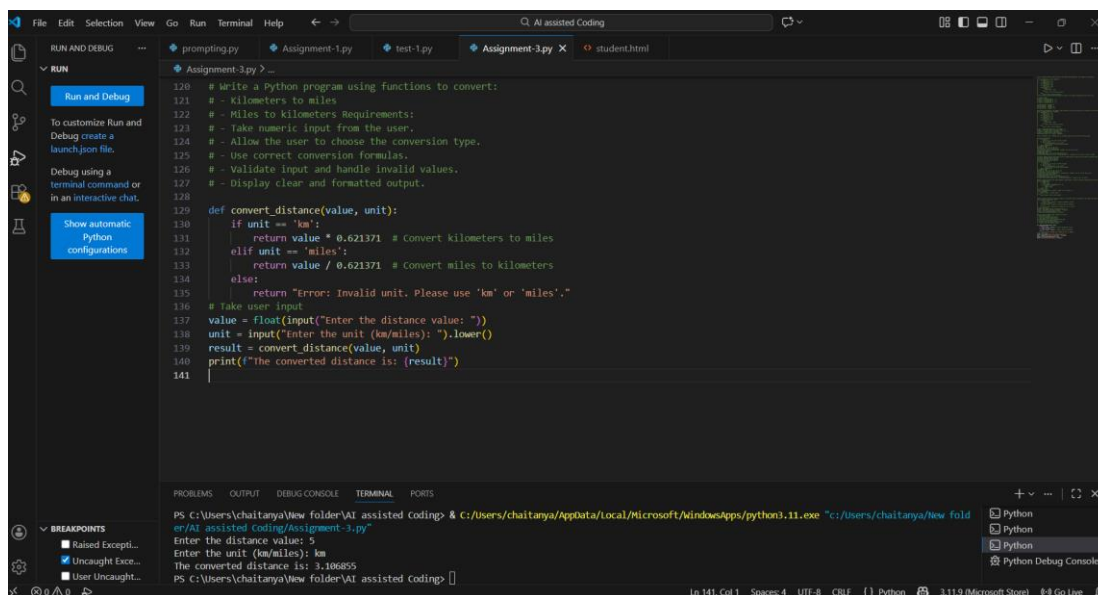
Write a Python program using functions to convert:

- 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
Assignment-3.py x student.html

RUN AND DEBUG
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

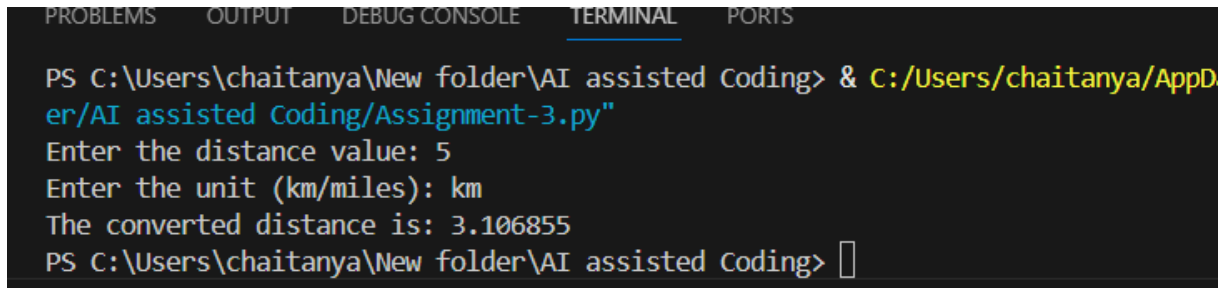
120 # Write a Python program using functions to convert:
121 # - Kilometers to miles
122 # - Miles to kilometers Requirements:
123 # - Take numeric input from the user.
124 # - Allow the user to choose the conversion type.
125 # - Use correct conversion formulas.
126 # - Validate input and handle invalid 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
137 # Take user input
138 value = float(input("Enter the distance value: "))
139 unit = input("Enter the unit (km/miles): ").lower()
140 result = convert_distance(value, unit)
141 print(f"The converted distance is: {result}")
```

```
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> |
```

Ln 141, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9 (Microsoft Store) P4 Go Live

## Output-2:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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

