Assignment-3.2

Name:M.Harshith

Ht.no:2303A51444

Batch.No:21

TaskDescription-1

• ProgressivePromptingforCalculatorDesign:AsktheAItodesignasimple calculator program by initially providing only the function name. Gradually enhance the prompt by adding comments and usage examples.

Promptused-1:

writeapythonfunctionnamedcalculatorthattakesthreeparameters:two numbers and an operator (as a string: '+', '-', '*', '/').

Code-1:

```python
# write a python function named calculator that takes three parameters: two numbers
def calculator(num1, num2, operator):
    try:
        if operator == '+':
            return num1 + num2
        elif operator == '-':
            return num1 - num2
        elif operator == '*':
            return num1 * num2
        elif operator == '/':
            if num2 != 0:
                return num1 / num2
            else:
                return "Error: Division by zero is not allowed."
        else:
            return "Error: Invalid operator. Please use '+', '-', '*', or '/'."
    except TypeError:
        return "Error: Invalid input types. Please enter numbers for calculation."
# Example usage:
user_input1 = float(input("Enter the first number: "))
user_input2 = float(input("Enter the second number: "))
user_operator = input("Enter the operator (+, -, *, /): ")
result = calculator(user_input1, user_input2, user_operator)
print(f"The result is: {result}")
print(result)
```

Output-1:

```
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter the first number: 10
Enter the second number: 5
Enter the operator (+, -, *, /): +
The result is: 15.0
15.0
```

Promptused-2:

writeapythonfunctionnamedcalculatorthattakesthreeparameters:two numbers and an operator (as a string: '+', '-', '*', '/').

requirements:

Take2numbersandoperatorasinput Handle

division by zero

Handleinvalidoperatorinputs

exampleinput:10,5,"+"output:15

Code-2:

```
# Take 2 numbers and operator as input
#Handle division by zero
# Handle invalid operator inputs
# example input: 10, 5, "+"  output:15
def calculator(num1, num2, operator):
    try:
        number1 = float(num1)
        number2 = float(num2)

        if operator == '+':
            return number1 + number2
        elif operator == '-':
            return number1 - number2
        elif operator == '*':
            return number1 * number2
        elif operator == '/':
            if number2 == 0:
                return "Error: Division by zero is not allowed."
            return number1 / number2
        else:
            return "Error: Invalid operator. Please use '+', '-', '*', or '/'."
    except ValueError:
        return "Error: Invalid input. Please enter numeric values for numbers."
user_num1 = input("Enter the first number: ")
user_num2 = input("Enter the second number: ")
user_operator = input("Enter the operator (+, -, *, /): ")
result = calculator(user_num1, user_num2, user_operator)
print("Result:", result)
```

Output-2:

```
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter the first number: 20
Enter the second number: 16
Enter the operator (+, -, *, /): -
Result: 4.0
PS D:\AI assist>
```

Comparisonof2codes:

Aspect	Initial CalculatorVersion	ImprovedCalculatorVersion
Input Handling	Assumesnumericinputsare already valid	Convertsinputstofloatand validates them
Error Handling	Handlesdivisionbyzero and invalid operator	Handlesinvalidnumbers,division by zero, and invalid operators
Code Structure	Basicconditionallogic inside function	Cleanerstructurewithexplicit input validation
Readability	Understandablebutless descriptive	Improved readability with commentsandclearvariable names
Robustness	Mayfailfornon-numeric inputs	Morerobustanduser-friendly
User Feedback	Limitederrormessages	Clearandinformativeerror messages
Suitability	Suitableforsimple calculations	Suitableforreal-worldanduser-facing programs

TaskDescription-2

• RefiningPromptsforSortingLogic:Startwithavaguepromptforsorting studentmarks,thenrefineittoclearlyspecifysortingorderandconstraints.

Code-1:

```python
# Write a Python program to sort student marks.
def sort_student_marks(marks):
    try:
        marks_list = [float(mark) for mark in marks]
        marks_list.sort()
        return marks_list
    except ValueError:
        return "Error: All marks should be numeric values."
user_marks = input("Enter student marks separated by spaces: ").split()
sorted_marks = sort_student_marks(user_marks)
print("Sorted student marks:", sorted_marks)
```

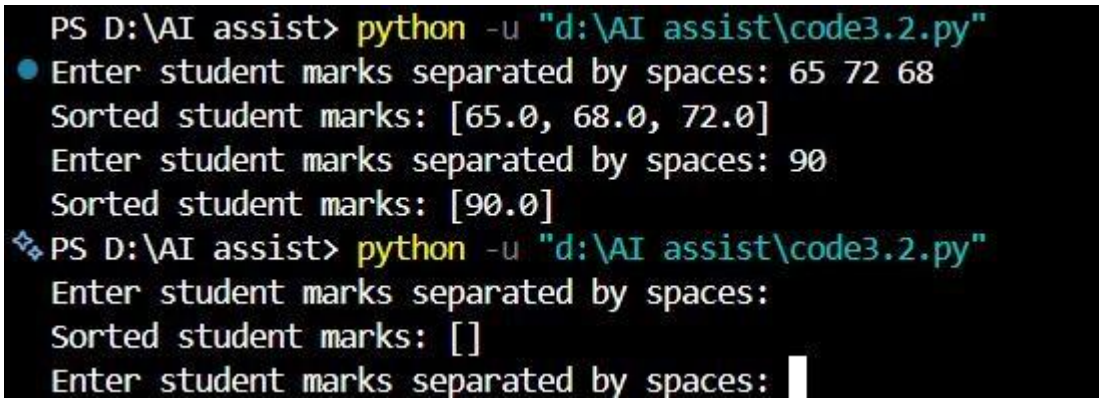
Output-1:

```
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces: 78 85 92 60
Sorted student marks: [60.0, 78.0, 85.0, 92.0]
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces: 90
Sorted student marks: [90.0]
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces:
Sorted student marks: []
PS D:\AI assist>
```

Code-2:

```python
# Write a Python function that takes a list of student marks as input and sorts them in ascending order.
#Constraints:
# Marks are integers between 0 and 100.
# Handle duplicate marks correctly.
# Use an efficient sorting approach.
# Display the sorted list clearly.
# Example input: [88, 92, 75, 88, 95] → Output: [75, 88, 88, 92, 95]
def sort_student_marks(marks):
    try:
        marks_list = [float(mark) for mark in marks]
        marks_list.sort()
        return marks_list
    except ValueError:
        return "Error: All marks should be numeric values."
user_marks = input("Enter student marks separated by spaces: ").split()
sorted_marks = sort_student_marks(user_marks)
print("Sorted student marks:", sorted_marks)
```

Output-2:



Justification:

Refiningthepromptbyspecifyingsortingorderandinputconstraintsimproved the accuracy and structure of the AI-generated sorting logic. TherefinedpromptguidedtheAItohandlenumericconversion,duplicates,and empty inputs more reliably.
ThisshowsthatclearerpromptsleadtomoreefficientandpredictableAI- generated programs.

TaskDescription-3

• Few-Shot Prompting for Prime Number Validation: Provide multiple input-outputexamplesforafunctionthatcheckswhetheranumberisprime.Observe how few-shot prompting improves correctness.

Promptused:

Writeapythonfunctionthatcheckswhetheragivennumberisprimeorprime. example

input: 2 output: prime number

input:7 output: prime number

input:1output:notaprimenumber

Code:

```python
# Write a python function that checks whether a given number is prime or prime.
# example input: 2 output: prime number
# input:7 output: prime number
# input:1 output: not a prime number
def is_prime(number):
    try:
        num = int(number)
        if num <= 1:
            return "Not a prime number"
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                return "Not a prime number"
        return "Prime number"
    except ValueError:
        return "Error: Invalid input. Please enter an integer."
user_number = input("Enter a number: ")
result = is_prime(user_number)
print("Result:", result)
```

Output:

```
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter a number: 2
Result: Prime number
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter a number: 1
Result: Not a prime number
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter a number: 0
Result: Not a prime number
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter a number: 11
Result: Prime number
```

Justification:

Few-shotpromptingimprovestheprime-checkinglogicbyshowingtheAIclear examples of both prime and non-prime numbers. TheseexamplesguidetheAItocorrectlyhandleedgecasessuchas0,1,and negative numbers, which are often missed in zero-shot prompting. Asaresult,thegeneratedfunctionismoreaccurate,efficient,andreliablein validating prime numbers.

TaskDescription-4

• Prompt-GuidedUIDesignforStudentGradingSystem:Createauserinterface for a student grading system that calculates total marks, percentage, and grade based on user input.

PromptUsed:

DesignasimpleanduserfriendlystudentgradingsystemUI Requirements:

1. Createinputfieldstoentermarksformultiplesubjects.
2. Calculatetotalmarksandpercentagebasedontheenteredvalues.
3. Assigngradesbasedonpercentage(e.g.,A,B,C,Fail).
4. Displaytotalmarks,percentage,andgradeclearlyonthescreen.
5. Useacleanandwell-structuredlayout.
6. Validateinputstoensureonlynumericvaluesareaccepted.
7. UseHTMLandCSSfortheuserinterfaceandJavaScriptfor calculations. create all files in one file

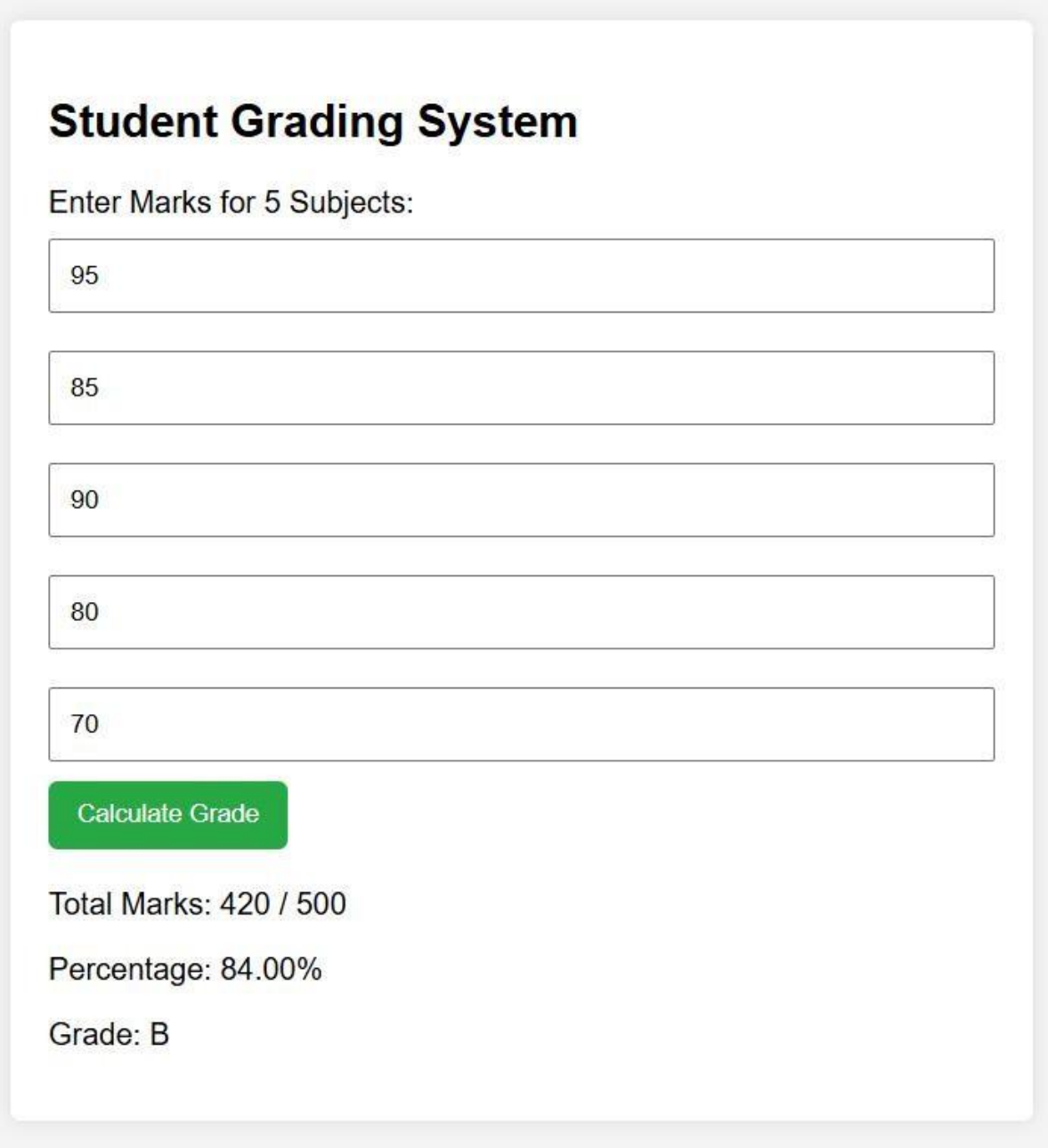
Code:

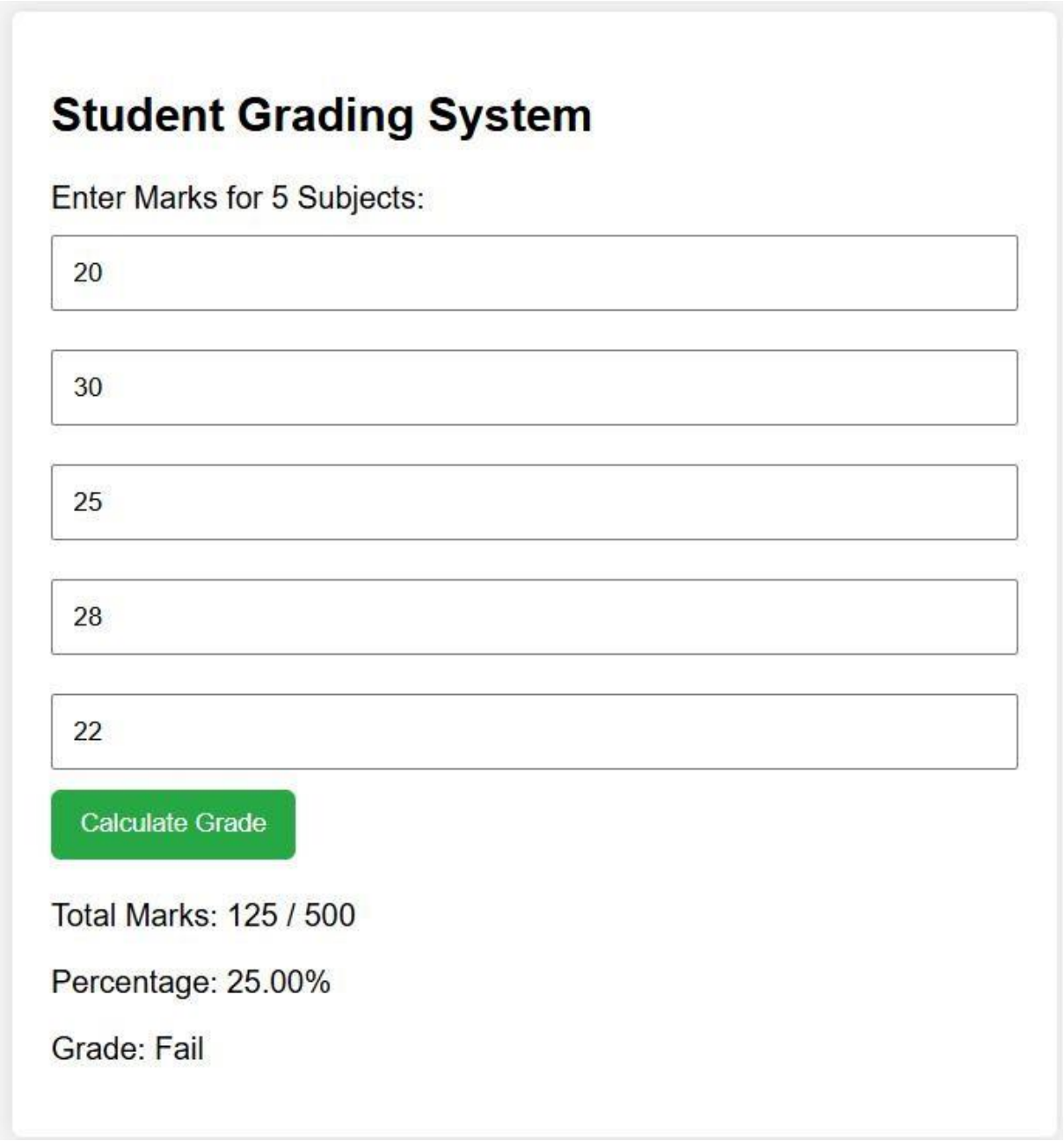
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Grading System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 500px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        input[type="number"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            box-sizing: border-box;
        }
        button {
            padding: 10px 15px;
            background-color: #28a745;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #218838;
        }
        .result {
            margin-top: 20px;
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <h2>Student Grading System</h2>
        <label for="marks">Enter Marks for 5 Subjects:</label>
        <input type="number" id="subject1" placeholder="Subject 1" min="0" max="100">
        <input type="number" id="subject2" placeholder="Subject 2" min="0" max="100">
        <input type="number" id="subject3" placeholder="Subject 3" min="0" max="100">
        <input type="number" id="subject4" placeholder="Subject 4" min="0" max="100">
        <input type="number" id="subject5" placeholder="Subject 5" min="0" max="100">
        <button onclick="calculateGrade()">Calculate Grade</button>
        <div class="result" id="result"></div>
    </div>

    <script>
        function calculateGrade() {
            let marks = [];
            for (let i = 1; i <= 5; i++) {
                let mark = parseFloat(document.getElementById('subject' + i).value);
                if (isNaN(mark) || mark < 0 || mark > 100) {
                    alert('Please enter valid marks between 0 and 100 for all subjects.');
                    return;
                }
                marks.push(mark);
            }

            let totalMarks = marks.reduce((a, b) => a + b, 0);
            let percentage = (totalMarks / 500) * 100;
            let grade;

            if (percentage >= 90) {
                grade = 'A';
            } else if (percentage >= 80) {
                grade = 'B';
            } else if (percentage >= 70) {
                grade = 'C';
            } else if (percentage >= 60) {
                grade = 'D';
            } else {
                grade = 'Fail';
            }

            document.getElementById('result').innerHTML = `
```

```
            document.getElementById('result').innerHTML = `
                <p>Total Marks: ${totalMarks} / 500</p>
                <p>Percentage: ${percentage.toFixed(2)}%</p>
                <p>Grade: ${grade}</p>
            `;
        }
    </script>
</body>
</html>
```

Output:


Student Grading System
Enter Marks for 5 Subjects:
95
85
90
80
70
Calculate Grade
Total Marks: 420 / 500
Percentage: 84.00%
Grade: B



Justification:

Theprompt-guidedUIdesignensuresaccuratecalculationoftotalmarks, percentage, and grade based on user input.
Clearinputvalidationimprovesreliabilitybyhandlinginvalidandmissing values effectively.
AstructuredUIlayoutenhancesreadabilityandprovidesclearoutputfor students and evaluators.

TaskDescription-5

• AnalyzingPromptSpecificityinUnitConversionFunctions:ImprovingaUnit ConversionFunction(KilometerstoMilesandMilestoKilometers)Using Clear Instructions.

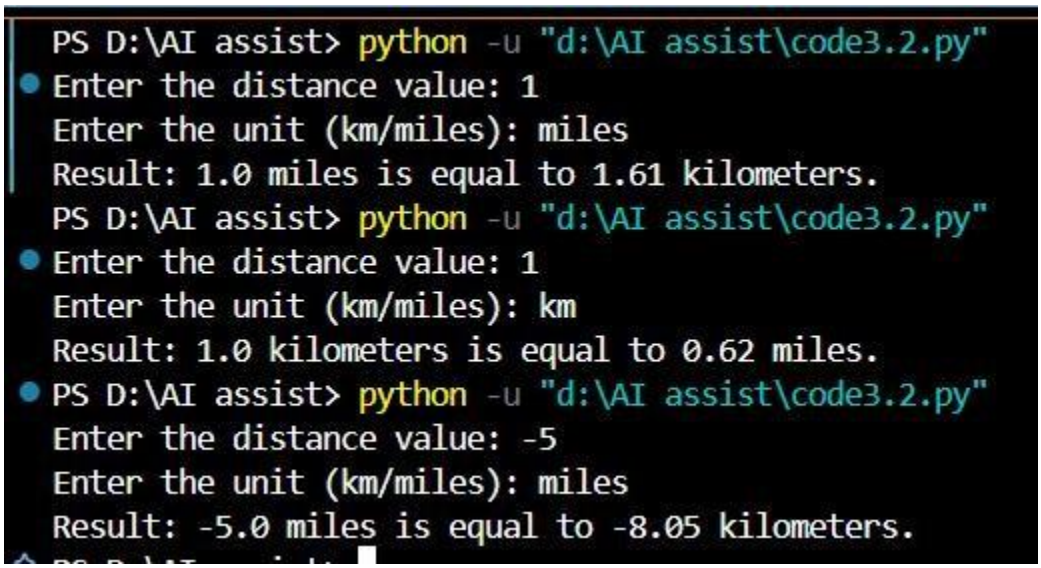
Promptused-1:

WriteaPythonfunctiontoconvertkilometerstomilesandmilestokilometers. Take input from the user and display the result.

Code-1:

```
# Write a Python function to convert kilometers to miles and miles to kilometers.
# Take input from the user and display the result.
def convert_distance(value, unit):
    try:
        distance = float(value)
        if unit.lower() == 'km':
            miles = distance * 0.621371
            return f"{distance} kilometers is equal to {miles:.2f} miles."
        elif unit.lower() == 'miles':
            kilometers = distance / 0.621371
            return f"{distance} miles is equal to {kilometers:.2f} kilometers."
        else:
            return "Error: Invalid unit. Please use 'km' for kilometers or 'miles' for miles."
    except ValueError:
        return "Error: Invalid input. Please enter a numeric value for distance."
user_value = input("Enter the distance value: ")
user_unit = input("Enter the unit (km/miles): ")
result = convert_distance(user_value, user_unit)
print("Result:", result)
```

Output-1:



Promptused-2:

WriteaPythonprogramusingfunctionstoconvert:

- Kilometerstomiles

- Milestokilometers

Requirements:

- Takenumericinputfromtheuser.

- Allowtheusertochoosetheconversiontype.

- Usecorrectconversionformulas.

- Validateinputandhandleinvalidvalues.

- Displayclearandformattedoutput.

Code-2:

```python
# 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.
def convert_distance(value, unit):
    try:
        distance = float(value)
        if unit.lower() == 'km':
            miles = distance * 0.621371
            return f"{distance} kilometers is equal to {miles:.2f} miles."
        elif unit.lower() == 'miles':
            kilometers = distance / 0.621371
            return f"{distance} miles is equal to {kilometers:.2f} kilometers."
        else:
            return "Error: Invalid unit. Please use 'km' for kilometers or 'miles' for miles."
    except ValueError:
        return "Error: Invalid input. Please enter a numeric value for distance."
user_value = input("Enter the distance value: ")
user_unit = input("Enter the unit (km/miles): ")
result = convert_distance(user_value, user_unit)
print("Result:", result)
```

Output-2:

```
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter the distance value: 1
Enter the unit (km/miles): miles
Result: 1.0 miles is equal to 1.61 kilometers.
Enter the distance value: 1
Enter the unit (km/miles): km
Result: 1.0 kilometers is equal to 0.62 miles.
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter the distance value: 0
Enter the unit (km/miles): miles
Result: 0.0 miles is equal to 0.00 kilometers.
Enter the distance value: python -u "d:\AI assist\code3.2.py"
Enter the unit (km/miles): -5
Result: Error: Invalid input. Please enter a numeric value for distance.
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter the distance value: abc
Enter the unit (km/miles): km
Result: Error: Invalid input. Please enter a numeric value for distance.
Enter the distance value:
```

Justification:

Inthefirstprompt,theunitconversionlogicproducescorrectresultsbutlacks strict validation, allowing negative distance values.
Witha moredetailedprompt,theprogramimprovesbyaddingcleareroutput formatting and better handling of invalid inputs.
Overall,increasedpromptspecificityleadstomoreaccurate,robust,anduser- friendly unit conversion code.