Assignment - 3.2

Name :P.Samanvith

Ht no: 2303A52090

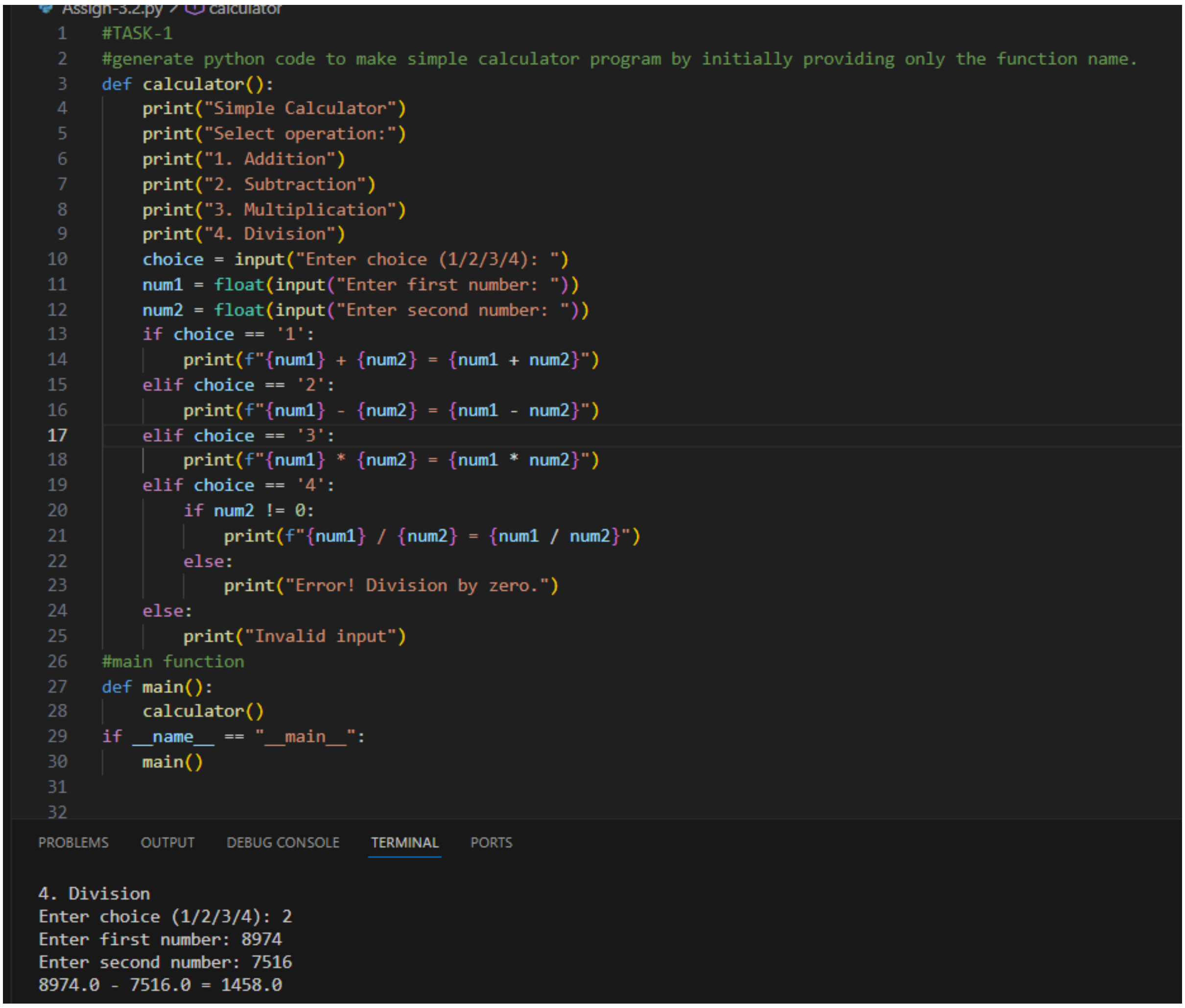
Batch: 40

Prompt Engineering – Improving Prompts and Context Management

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.

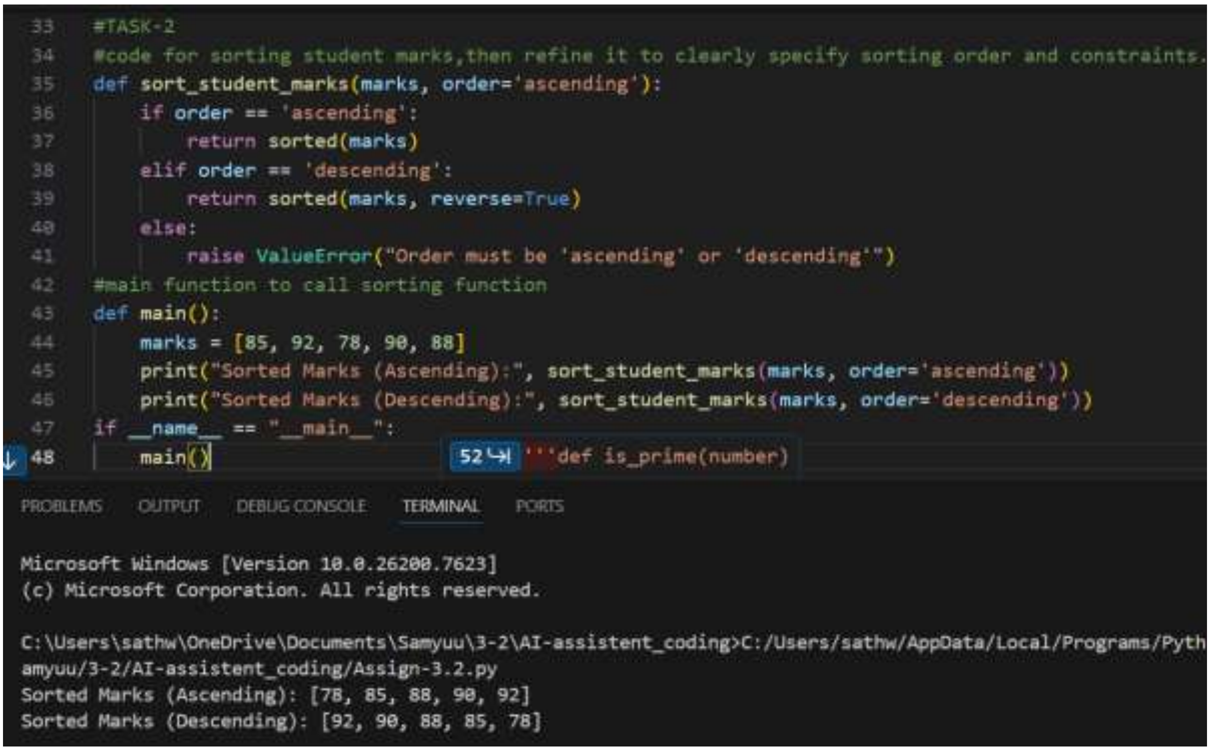


```
#TASK-1
#generate python code to make simple calculator program by initially providing only the function name.
def calculator():
    print("Simple Calculator")
    print("Select operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    choice = input("Enter choice (1/2/3/4): ")
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    if choice == '1':
        print(f"{num1} + {num2} = {num1 + num2}")
    elif choice == '2':
        print(f"{num1} - {num2} = {num1 - num2}")
    elif choice == '3':
        print(f"{num1} * {num2} = {num1 * num2}")
    elif choice == '4':
        if num2 != 0:
            print(f"{num1} / {num2} = {num1 / num2}")
        else:
            print("Error! Division by zero.")
    else:
        print("Invalid input")
#main function
def main():
    calculator()
if __name__ == "__main__":
    main()
```

```
4. Division
Enter choice (1/2/3/4): 2
Enter first number: 8974
Enter second number: 7516
8974.0 - 7516.0 = 1458.0
```

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.



```python
#TASK-2
#code for sorting student marks,then refine it to clearly specify sorting order and constraints.
def sort_student_marks(marks, order='ascending'):
    if order == 'ascending':
        return sorted(marks)
    elif order == 'descending':
        return sorted(marks, reverse=True)
    else:
        raise ValueError("Order must be 'ascending' or 'descending'")
#main function to call sorting function
def main():
    marks = [85, 92, 78, 90, 88]
    print("Sorted Marks (Ascending):", sort_student_marks(marks, order='ascending'))
    print("Sorted Marks (Descending):", sort_student_marks(marks, order='descending'))
if __name__ == "__main__":
    main()
```

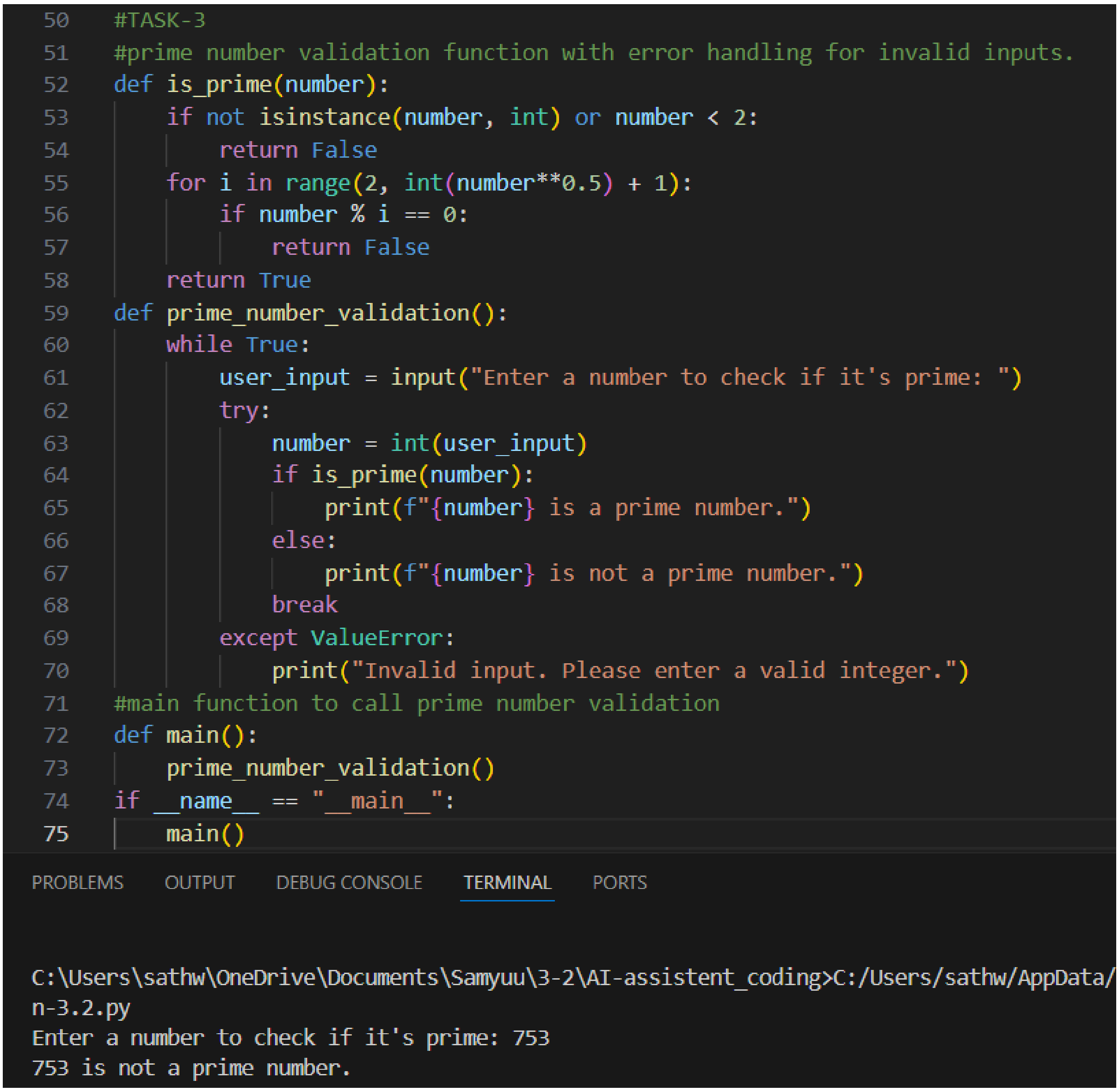
```
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sathw\OneDrive\Documents\Samyuu\3-2\AI-assistent_coding>C:/Users/sathw/AppData/Local/Programs/Pyth
amyuu/3-2/AI-assistent_coding/Assign-3.2.py
Sorted Marks (Ascending): [78, 85, 88, 90, 92]
Sorted Marks (Descending): [92, 90, 88, 85, 78]
```

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.



```python
#TASK-3
#prime number validation function with error handling for invalid inputs.
def is_prime(number):
    if not isinstance(number, int) or number < 2:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True
def prime_number_validation():
    while True:
        user_input = input("Enter a number to check if it's prime: ")
        try:
            number = int(user_input)
            if is_prime(number):
                print(f"{number} is a prime number.")
            else:
                print(f"{number} is not a prime number.")
            break
        except ValueError:
            print("Invalid input. Please enter a valid integer.")
#main function to call prime number validation
def main():
    prime_number_validation()
if __name__ == "__main__":
    main()
```

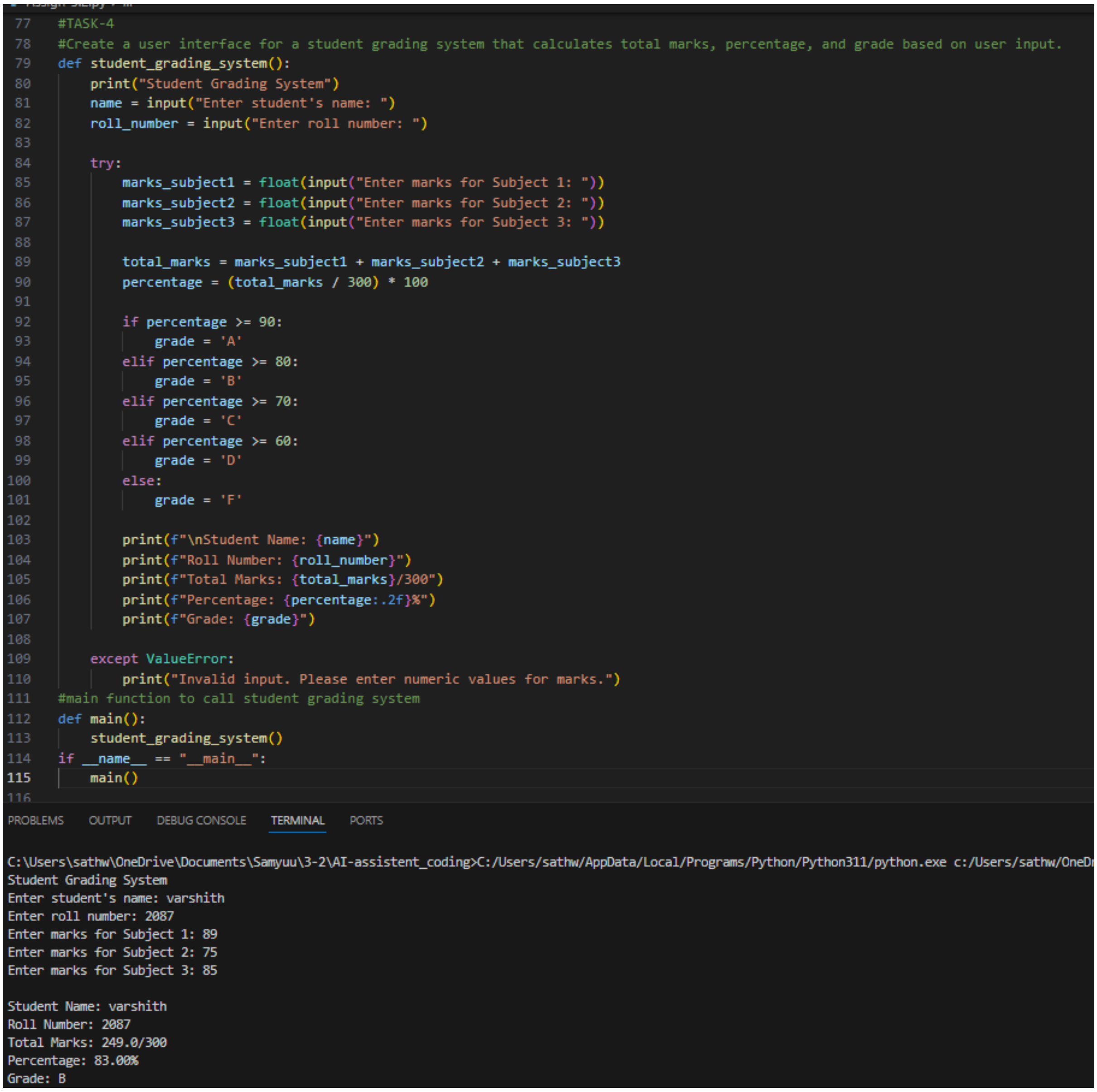
```
C:\Users\sathw\OneDrive\Documents\Samyuu\3-2\AI-assistent_coding>C:/Users/sathw/AppData/
n-3.2.py
Enter a number to check if it's prime: 753
753 is not a prime number.
```

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

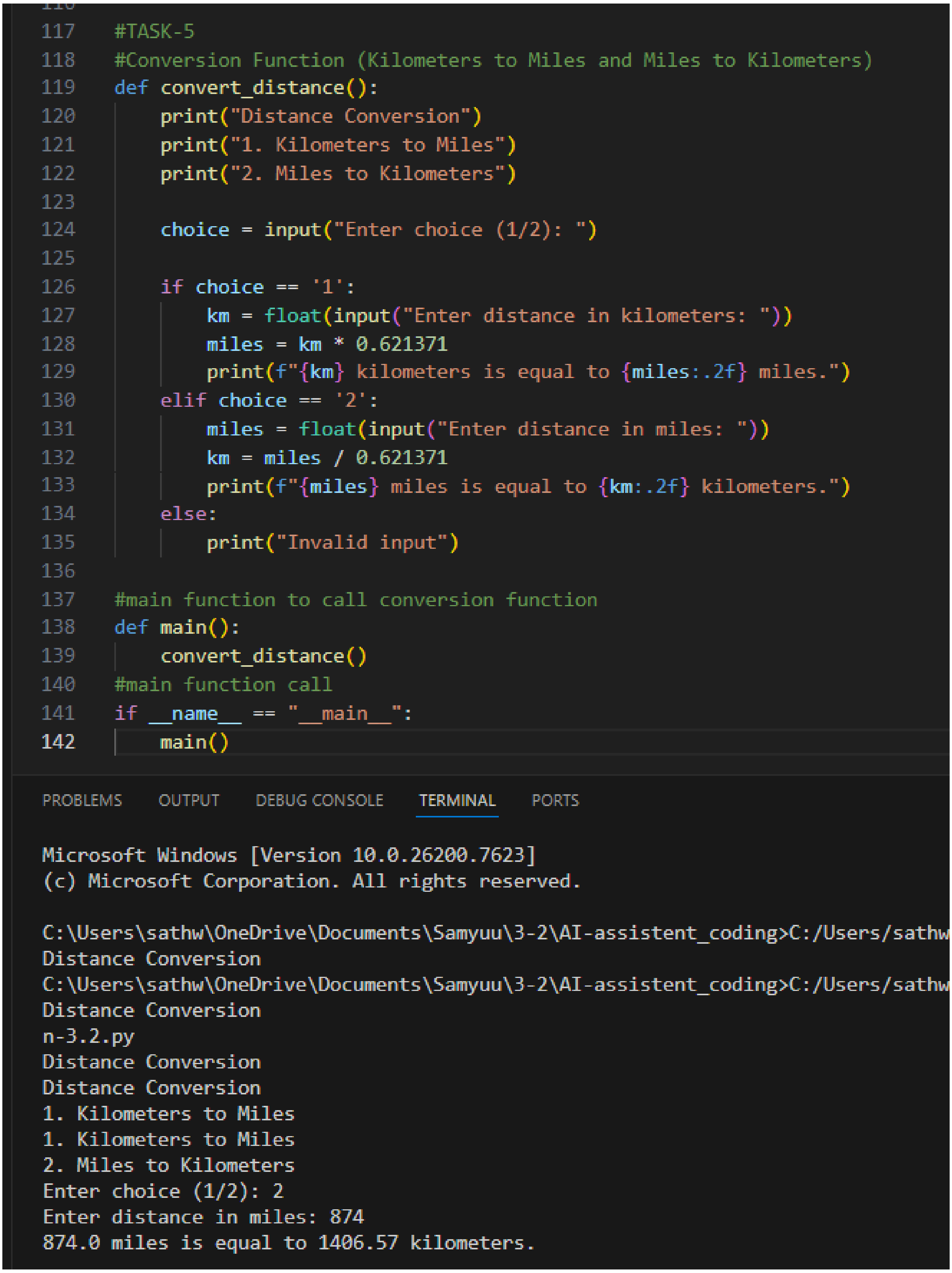


```python
#TASK-4
#Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.
def student_grading_system():
    print("Student Grading System")
    name = input("Enter student's name: ")
    roll_number = input("Enter roll number: ")

    try:
        marks_subject1 = float(input("Enter marks for Subject 1: "))
        marks_subject2 = float(input("Enter marks for Subject 2: "))
        marks_subject3 = float(input("Enter marks for Subject 3: "))

        total_marks = marks_subject1 + marks_subject2 + marks_subject3
        percentage = (total_marks / 300) * 100

        if percentage >= 90:
            grade = 'A'
        elif percentage >= 80:
            grade = 'B'
        elif percentage >= 70:
            grade = 'C'
        elif percentage >= 60:
            grade = 'D'
        else:
            grade = 'F'

        print(f"\nStudent Name: {name}")
        print(f"Roll Number: {roll_number}")
        print(f"Total Marks: {total_marks}/300")
        print(f"Percentage: {percentage:.2f}%")
        print(f"Grade: {grade}")

    except ValueError:
        print("Invalid input. Please enter numeric values for marks.")
#main function to call student grading system
def main():
    student_grading_system()
if __name__ == "__main__":
    main()
```

```
C:\Users\sathw\OneDrive\Documents\Samyuu\3-2\AI-assistent_coding>C:/Users/sathw/AppData/Local/Programs/Python/Python311/python.exe c:/Users/sathw/OneDr
Student Grading System
Enter student's name: varshith
Enter roll number: 2087
Enter marks for Subject 1: 89
Enter marks for Subject 2: 75
Enter marks for Subject 3: 85

Student Name: varshith
Roll Number: 2087
Total Marks: 249.0/300
Percentage: 83.00%
Grade: B
```

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.



```python
#TASK-5
#Conversion Function (Kilometers to Miles and Miles to Kilometers)
def convert_distance():
    print("Distance Conversion")
    print("1. Kilometers to Miles")
    print("2. Miles to Kilometers")

    choice = input("Enter choice (1/2): ")

    if choice == '1':
        km = float(input("Enter distance in kilometers: "))
        miles = km * 0.621371
        print(f"{km} kilometers is equal to {miles:.2f} miles.")
    elif choice == '2':
        miles = float(input("Enter distance in miles: "))
        km = miles / 0.621371
        print(f"{miles} miles is equal to {km:.2f} kilometers.")
    else:
        print("Invalid input")

#main function to call conversion function
def main():
    convert_distance()
#main function call
if __name__ == "__main__":
    main()
```

```
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sathw\OneDrive\Documents\Samyuu\3-2\AI-assistent_coding>C:/Users/sathw
Distance Conversion
C:\Users\sathw\OneDrive\Documents\Samyuu\3-2\AI-assistent_coding>C:/Users/sathw
Distance Conversion
n-3.2.py
Distance Conversion
Distance Conversion
1. Kilometers to Miles
1. Kilometers to Miles
2. Miles to Kilometers
Enter choice (1/2): 2
Enter distance in miles: 874
874.0 miles is equal to 1406.57 kilometers.
```