Assignment-3.2

A.Siddartha

2303A52340

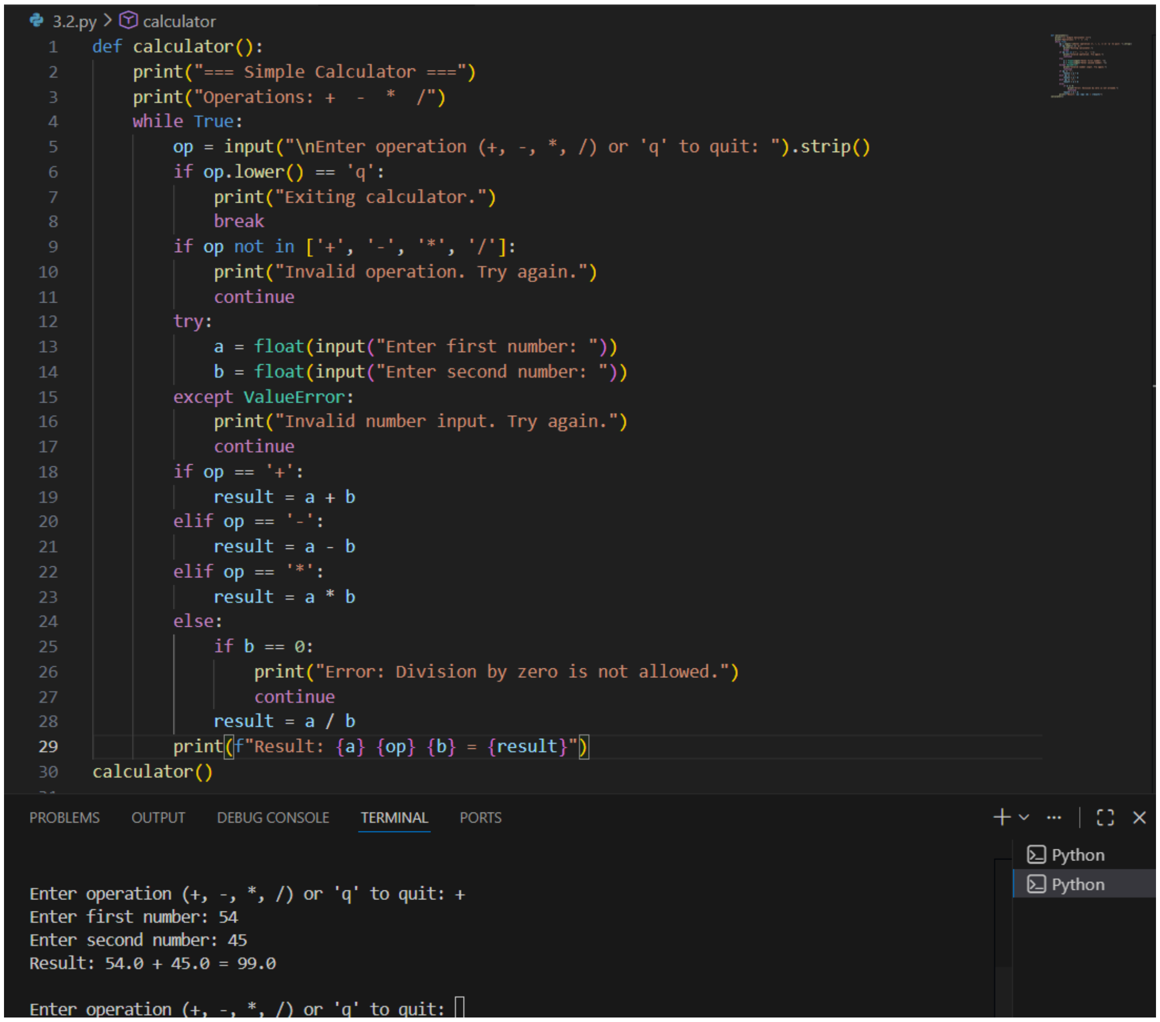
Task Description-1

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.

Code:



```python
def calculator():
    print("=== Simple Calculator ===")
    print("Operations: +  -  *  /")
    while True:
        op = input("\nEnter operation (+, -, *, /) or 'q' to quit: ").strip()
        if op.lower() == 'q':
            print("Exiting calculator.")
            break
        if op not in ['+', '-', '*', '/']:
            print("Invalid operation. Try again.")
            continue
        try:
            a = float(input("Enter first number: "))
            b = float(input("Enter second number: "))
        except ValueError:
            print("Invalid number input. Try again.")
            continue
        if op == '+':
            result = a + b
        elif op == '-':
            result = a - b
        elif op == '*':
            result = a * b
        else:
            if b == 0:
                print("Error: Division by zero is not allowed.")
                continue
            result = a / b
        print(f"Result: {a} {op} {b} = {result}")
calculator()
```

```
Enter operation (+, -, *, /) or 'q' to quit: +
Enter first number: 54
Enter second number: 45
Result: 54.0 + 45.0 = 99.0

Enter operation (+, -, *, /) or 'q' to quit: 
```

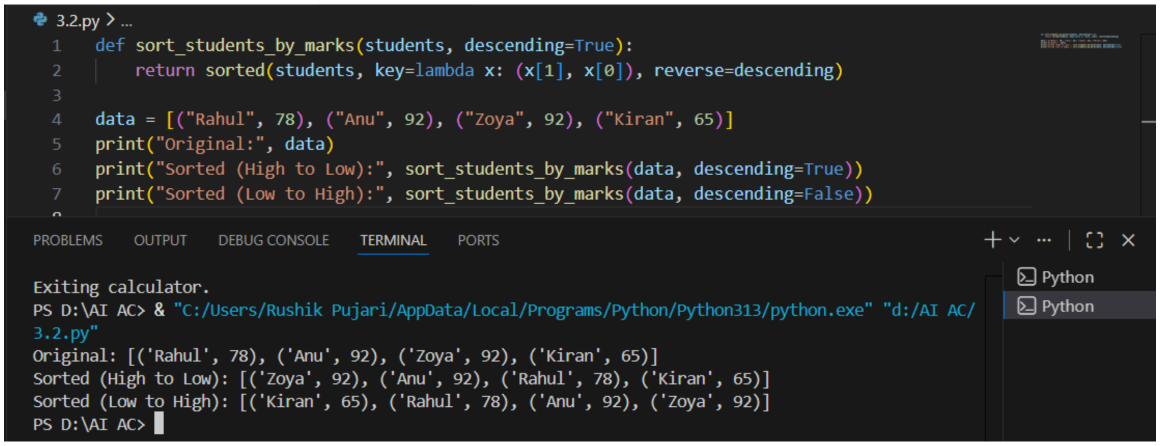
Task Description-2

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.

Code:



```python
def sort_students_by_marks(students, descending=True):
    return sorted(students, key=lambda x: (x[1], x[0]), reverse=descending)

data = [("Rahul", 78), ("Anu", 92), ("Zoya", 92), ("Kiran", 65)]
print("Original:", data)
print("Sorted (High to Low):", sort_students_by_marks(data, descending=True))
print("Sorted (Low to High):", sort_students_by_marks(data, descending=False))
```

```
Exiting calculator.
PS D:\AI AC> & "C:/Users/Rushik Pujari/AppData/Local/Programs/Python/Python313/python.exe" "d:/AI AC/
3.2.py"
Original: [('Rahul', 78), ('Anu', 92), ('Zoya', 92), ('Kiran', 65)]
Sorted (High to Low): [('Zoya', 92), ('Anu', 92), ('Rahul', 78), ('Kiran', 65)]
Sorted (Low to High): [('Kiran', 65), ('Rahul', 78), ('Anu', 92), ('Zoya', 92)]
PS D:\AI AC>
```

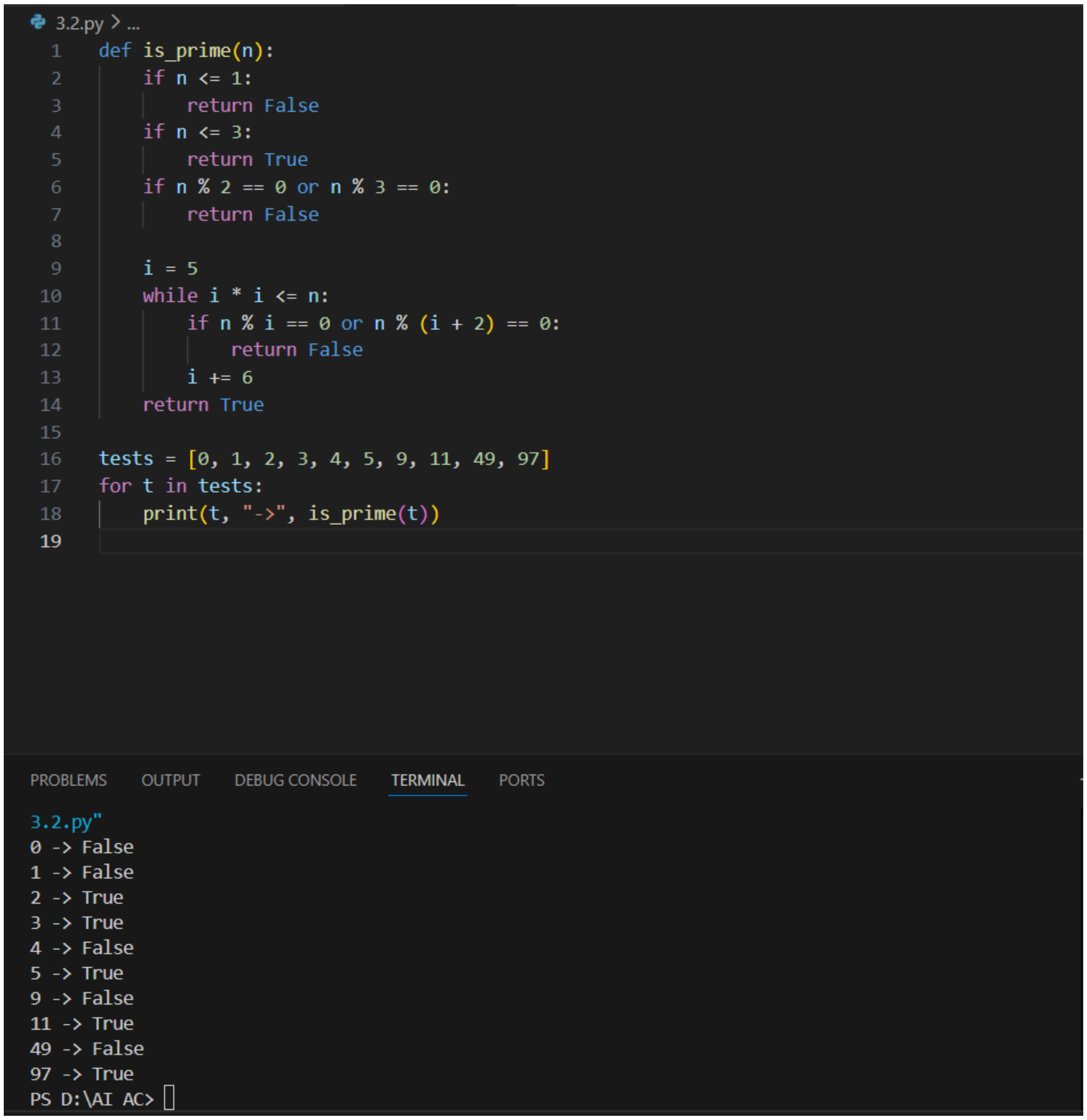
Task Description-3

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.

Code:



3.2.py > ...

```python
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False

    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

tests = [0, 1, 2, 3, 4, 5, 9, 11, 49, 97]
for t in tests:
    print(t, "->", is_prime(t))
```

```
3.2.py"
0 -> False
1 -> False
2 -> True
3 -> True
4 -> False
5 -> True
9 -> False
11 -> True
49 -> False
97 -> True
PS D:\AI AC>
```

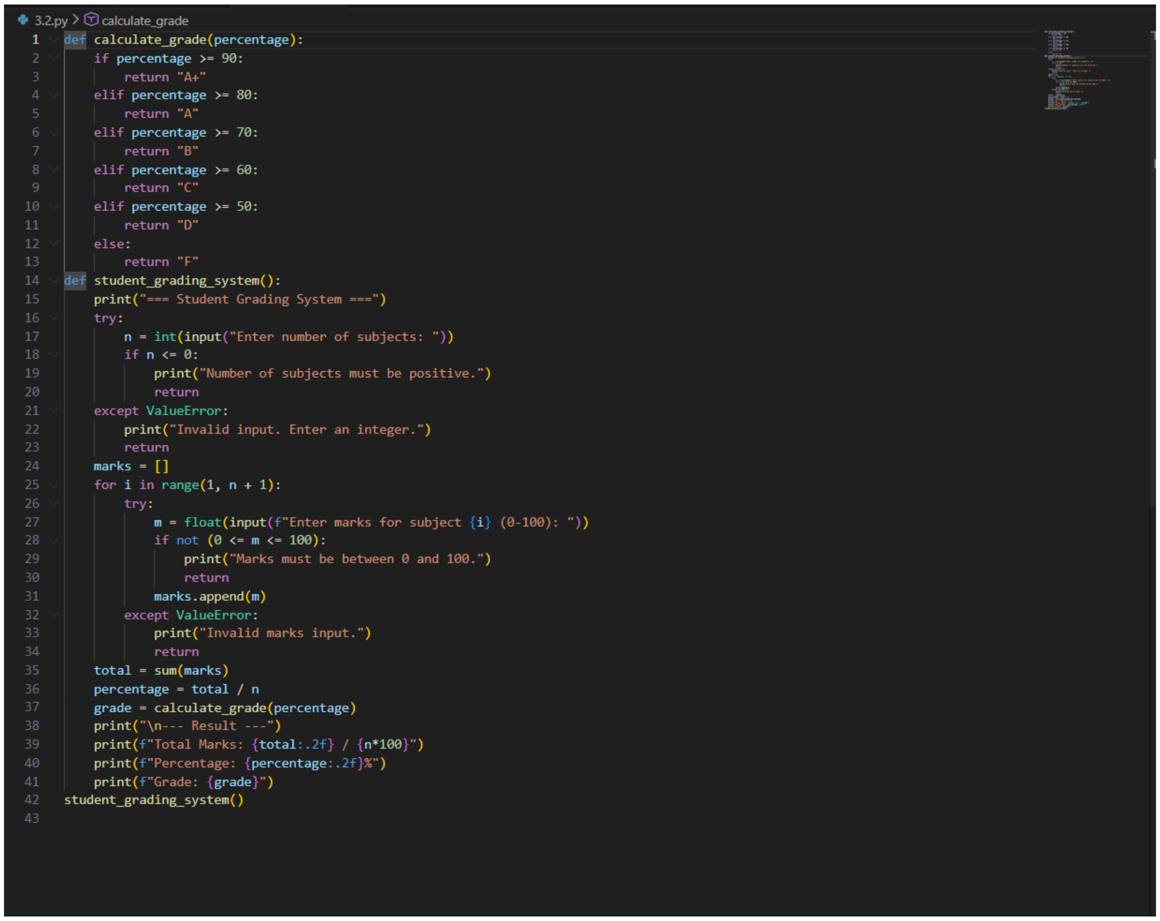
Task Description-4

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

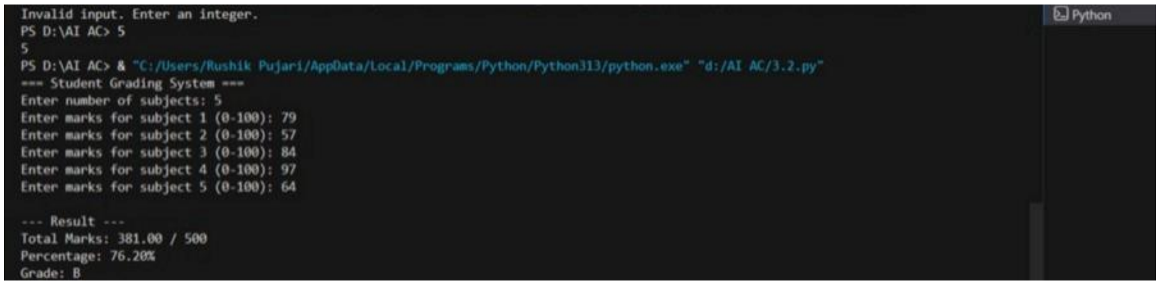
Well-structured UI code with accurate calculations and clear output display.

Code:



3.2.py > calculate_grade

```python
def calculate_grade(percentage):
    if percentage >= 90:
        return "A+"
    elif percentage >= 80:
        return "A"
    elif percentage >= 70:
        return "B"
    elif percentage >= 60:
        return "C"
    elif percentage >= 50:
        return "D"
    else:
        return "F"
def student_grading_system():
    print("=== Student Grading System ===")
    try:
        n = int(input("Enter number of subjects: "))
        if n <= 0:
            print("Number of subjects must be positive.")
            return
    except ValueError:
        print("Invalid input. Enter an integer.")
        return
    marks = []
    for i in range(1, n + 1):
        try:
            m = float(input(f"Enter marks for subject {i} (0-100): "))
            if not (0 <= m <= 100):
                print("Marks must be between 0 and 100.")
                return
            marks.append(m)
        except ValueError:
            print("Invalid marks input.")
            return
    total = sum(marks)
    percentage = total / n
    grade = calculate_grade(percentage)
    print("\n--- Result ---")
    print(f"Total Marks: {total:.2f} / {n*100}")
    print(f"Percentage: {percentage:.2f}%")
    print(f"Grade: {grade}")
student_grading_system()
```



```
Invalid input. Enter an integer.
PS D:\AI AC> 5
5
PS D:\AI AC> & "C:/Users/Rushik Pujari/AppData/Local/Programs/Python/Python313/python.exe" "d:/AI AC/3.2.py"
=== Student Grading System ===
Enter number of subjects: 5
Enter marks for subject 1 (0-100): 79
Enter marks for subject 2 (0-100): 57
Enter marks for subject 3 (0-100): 84
Enter marks for subject 4 (0-100): 97
Enter marks for subject 5 (0-100): 64

--- Result ---
Total Marks: 381.00 / 500
Percentage: 76.20%
Grade: B
```

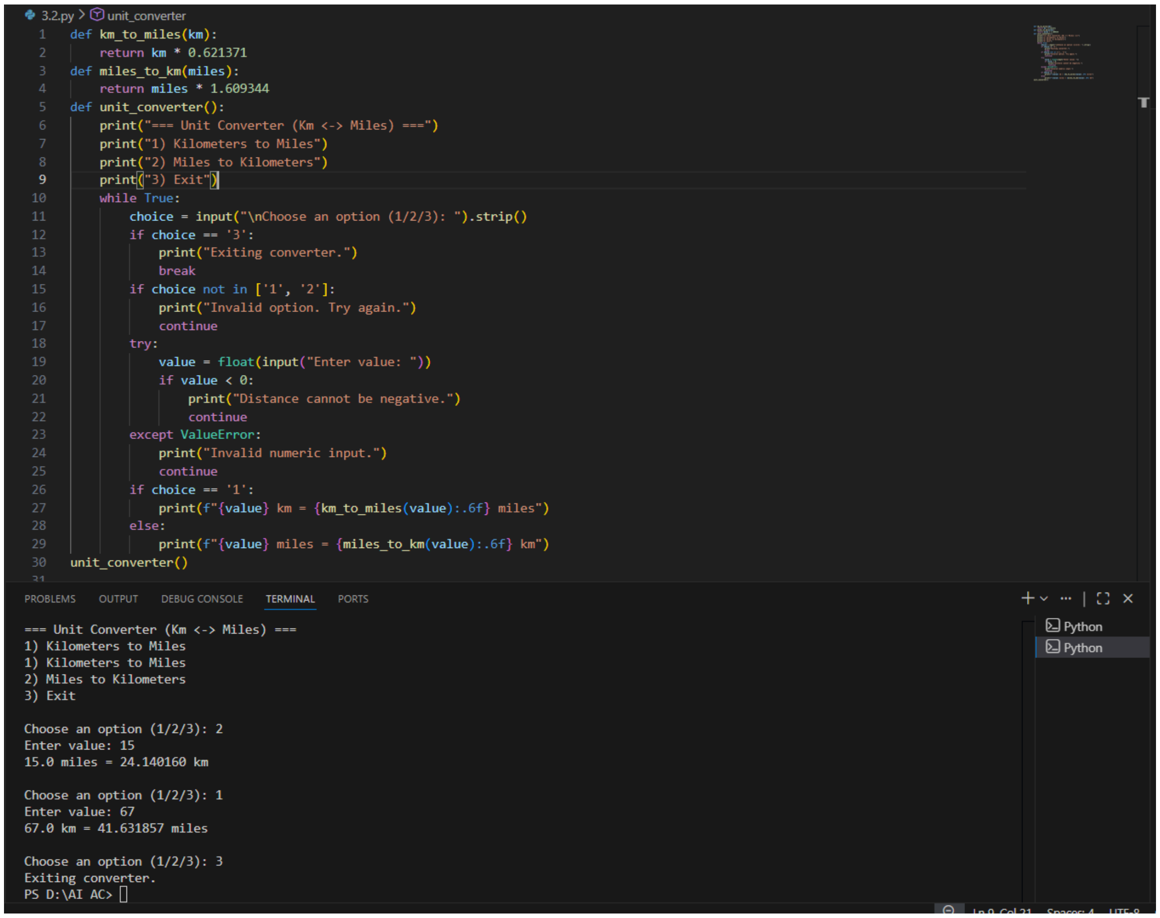
Task Description-5

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

Expected Output-5

Analysis of code quality and accuracy differences across multiple prompt variations.

Code:



3.2.py > unit_converter

```python
def km_to_miles(km):
    return km * 0.621371
def miles_to_km(miles):
    return miles * 1.609344
def unit_converter():
    print("=== Unit Converter (Km <-> Miles) ===")
    print("1) Kilometers to Miles")
    print("2) Miles to Kilometers")
    print("3) Exit")
    while True:
        choice = input("\nChoose an option (1/2/3): ").strip()
        if choice == '3':
            print("Exiting converter.")
            break
        if choice not in ['1', '2']:
            print("Invalid option. Try again.")
            continue
        try:
            value = float(input("Enter value: "))
            if value < 0:
                print("Distance cannot be negative.")
                continue
        except ValueError:
            print("Invalid numeric input.")
            continue
        if choice == '1':
            print(f"{value} km = {km_to_miles(value):.6f} miles")
        else:
            print(f"{value} miles = {miles_to_km(value):.6f} km")
unit_converter()
```

```
=== Unit Converter (Km <-> Miles) ===
1) Kilometers to Miles
1) Kilometers to Miles
2) Miles to Kilometers
3) Exit

Choose an option (1/2/3): 2
Enter value: 15
15.0 miles = 24.140160 km

Choose an option (1/2/3): 1
Enter value: 67
67.0 km = 41.631857 miles

Choose an option (1/2/3): 3
Exiting converter.
PS D:\AI AC>
```