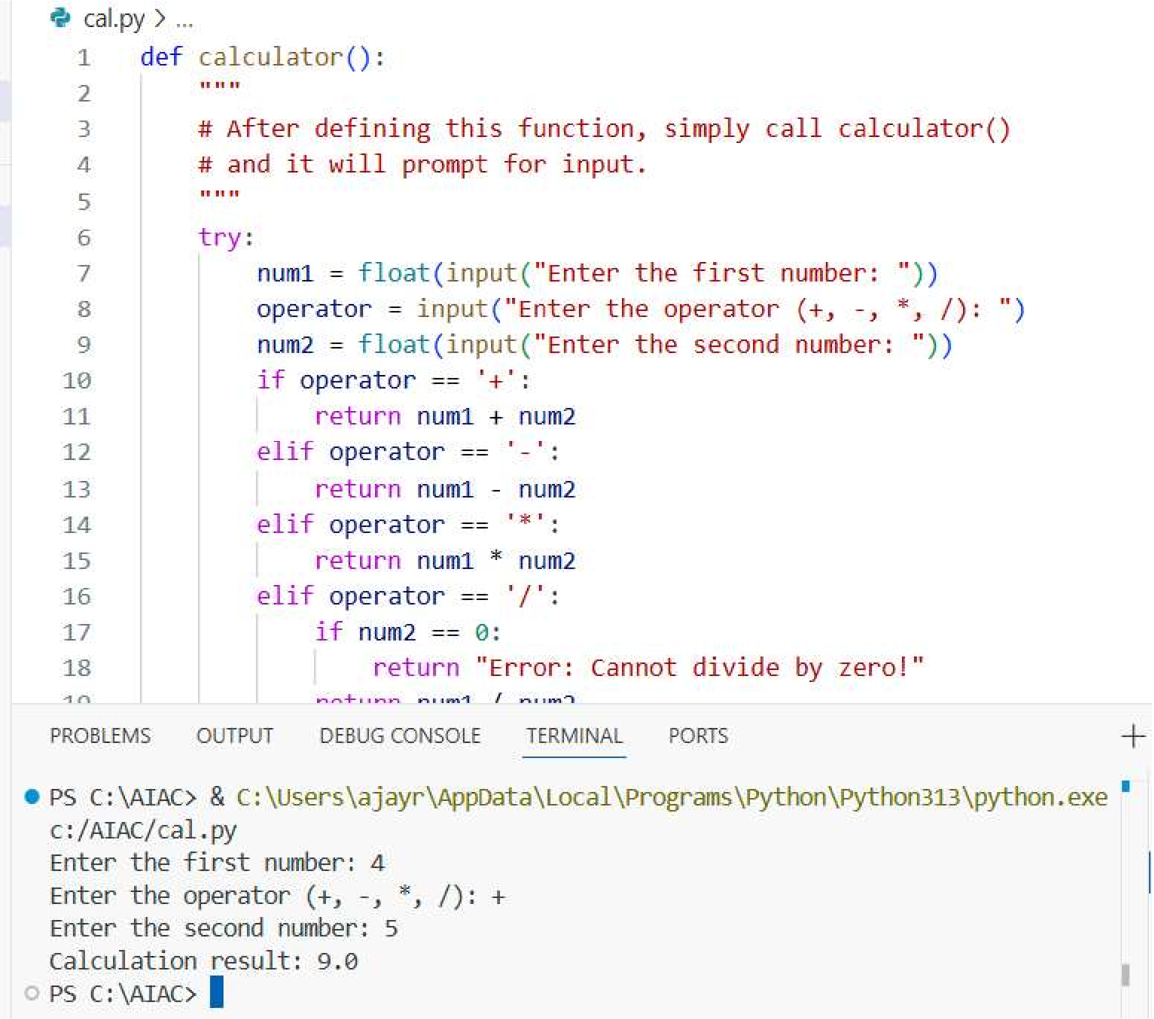
# AI - Assisted Coding-Assignment -3

**HTNO:2303A52228**
**Batch:37**

**Lab 3: 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.
**Expected Output:**
Comparison showing improvement in AI-generated calculator logic and structure.
**Prompt:**
Design a simple calculator function.It should take two numbers and an operator Support +, -, *, / ,Handle division by zero.
**CODE AND OUTPUT:**

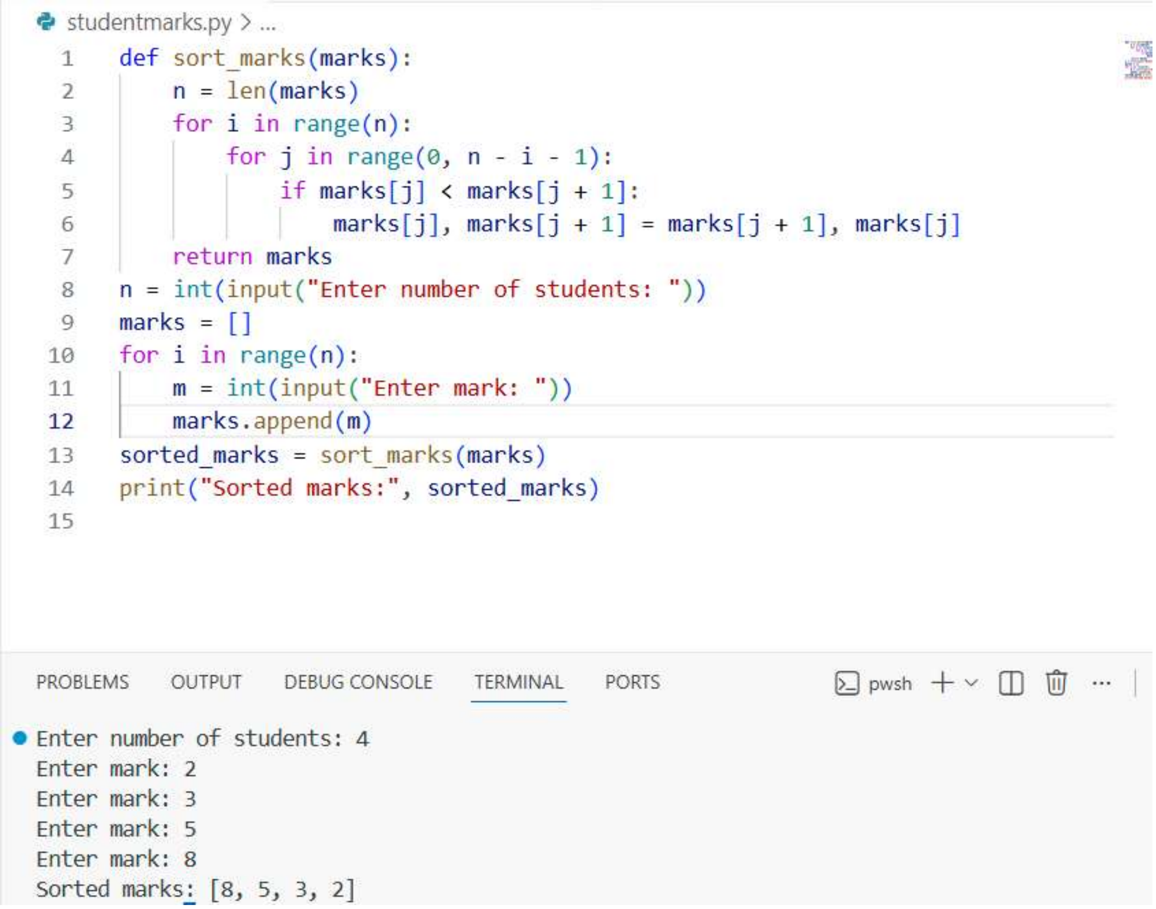


```python
def calculator():
    """
    # After defining this function, simply call calculator()
    # and it will prompt for input.
    """
    try:
        num1 = float(input("Enter the first number: "))
        operator = input("Enter the operator (+, -, *, /): ")
        num2 = float(input("Enter the second number: "))
        if operator == '+':
            return num1 + num2
        elif operator == '-':
            return num1 - num2
        elif operator == '*':
            return num1 * num2
        elif operator == '/':
            if num2 == 0:
                return "Error: Cannot divide by zero!"
            return num1 / num2
```

```
PS C:\AIAC> & C:\Users\ajayr\AppData\Local\Programs\Python\Python313\python.exe c:/AIAC/cal.py
Enter the first number: 4
Enter the operator (+, -, *, /): +
Enter the second number: 5
Calculation result: 9.0
PS C:\AIAC>
```

**Explanation:**
The program takes two numbers and an operator as input from the user at runtime.
The calculator function performs the selected arithmetic operation.
It checks for division by zero and invalid operators to avoid errors.
The final result is displayed to the user.

**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:**
AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.
**Prompt:**
Design a function to sort student marks in descending order using Bubble Sort.
**CODE AND OUTPUT:**



```python
def sort_marks(marks):
    n = len(marks)
    for i in range(n):
        for j in range(0, n - i - 1):
            if marks[j] < marks[j + 1]:
                marks[j], marks[j + 1] = marks[j + 1], marks[j]
    return marks
n = int(input("Enter number of students: "))
marks = []
for i in range(n):
    m = int(input("Enter mark: "))
    marks.append(m)
sorted_marks = sort_marks(marks)
print("Sorted marks:", sorted_marks)
```

```
Enter number of students: 4
Enter mark: 2
Enter mark: 3
Enter mark: 5
Enter mark: 8
Sorted marks: [8, 5, 3, 2]
```

**Explanation:**
The program first asks the user for the number of students.
It then takes each student's mark as input and stores them in a list.
A bubble sort algorithm is used to sort the marks in descending order without built-in functions.Finally, the sorted marks list is displayed to the user.

**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.

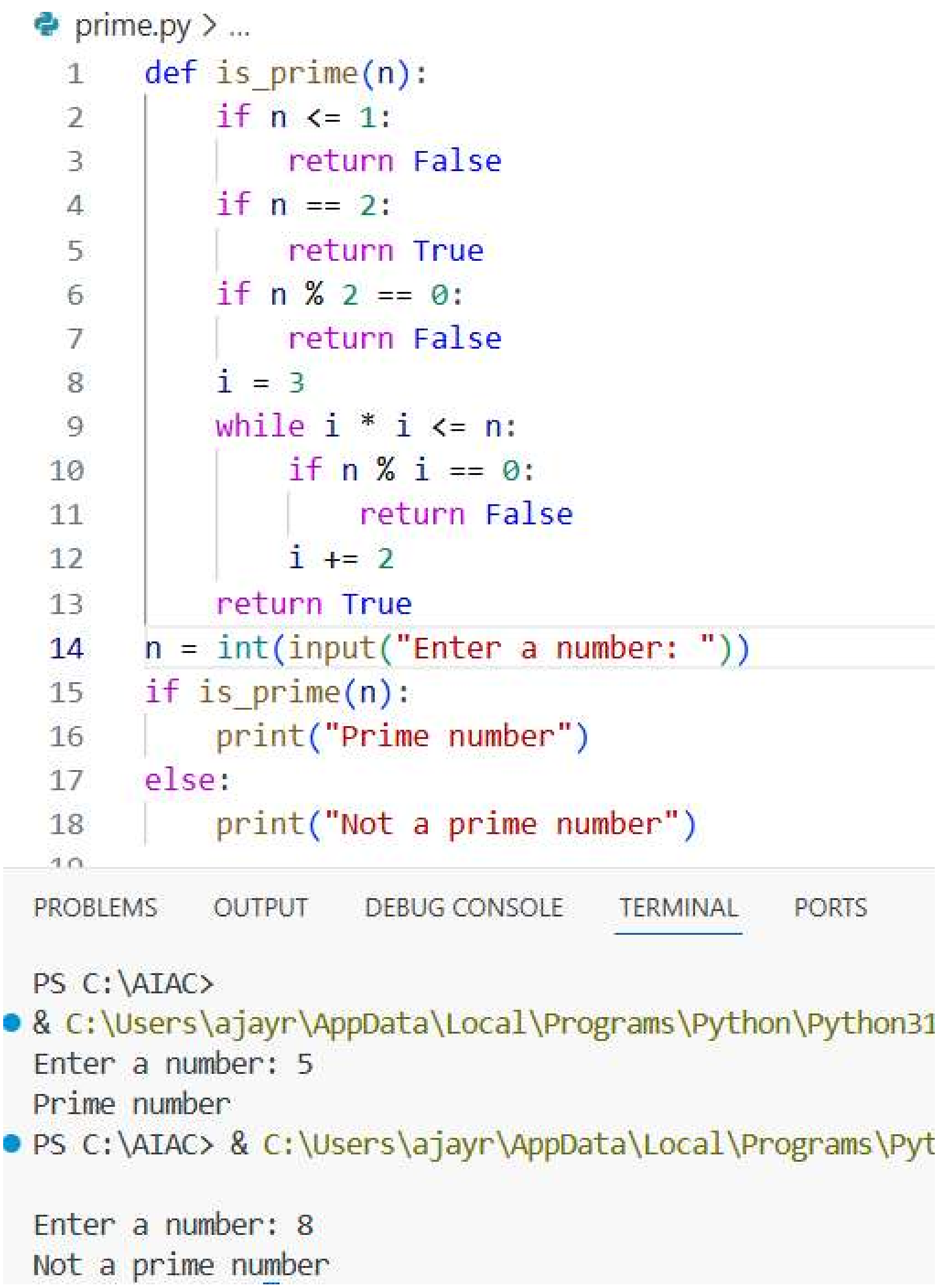
**Expected Output:**

Improved prime-checking function with better edge-case handling.

**Prompt:**

Design a Python program to check whether a number is prime by taking the input dynamically from the user. The program should correctly handle edge cases such as 0 and 1 (not prime), identify prime numbers like 2 and 17, and non-prime numbers like 4 and 18. It should use efficient approach by checking divisibility only up to the square root of the number and then display whether the entered number is prime or not.

**CODE AND OUTPUT:**



```python
def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2
    return True
n = int(input("Enter a number: "))
if is_prime(n):
    print("Prime number")
else:
    print("Not a prime number")
```

```
PS C:\AIAC>
& C:\Users\ajayr\AppData\Local\Programs\Python\Python31
Enter a number: 5
Prime number
PS C:\AIAC> & C:\Users\ajayr\AppData\Local\Programs\Pyt

Enter a number: 8
Not a prime number
```

**Explanation:**
The code first takes a number as input from the user. It then checks special cases such as numbers less than or equal to 1, which are not prime. For efficiency, it eliminates even numbers greater than 2 and checks divisibility only up to the square root of the number. Finally, it prints whether the given number is a prime number or not.

**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:**

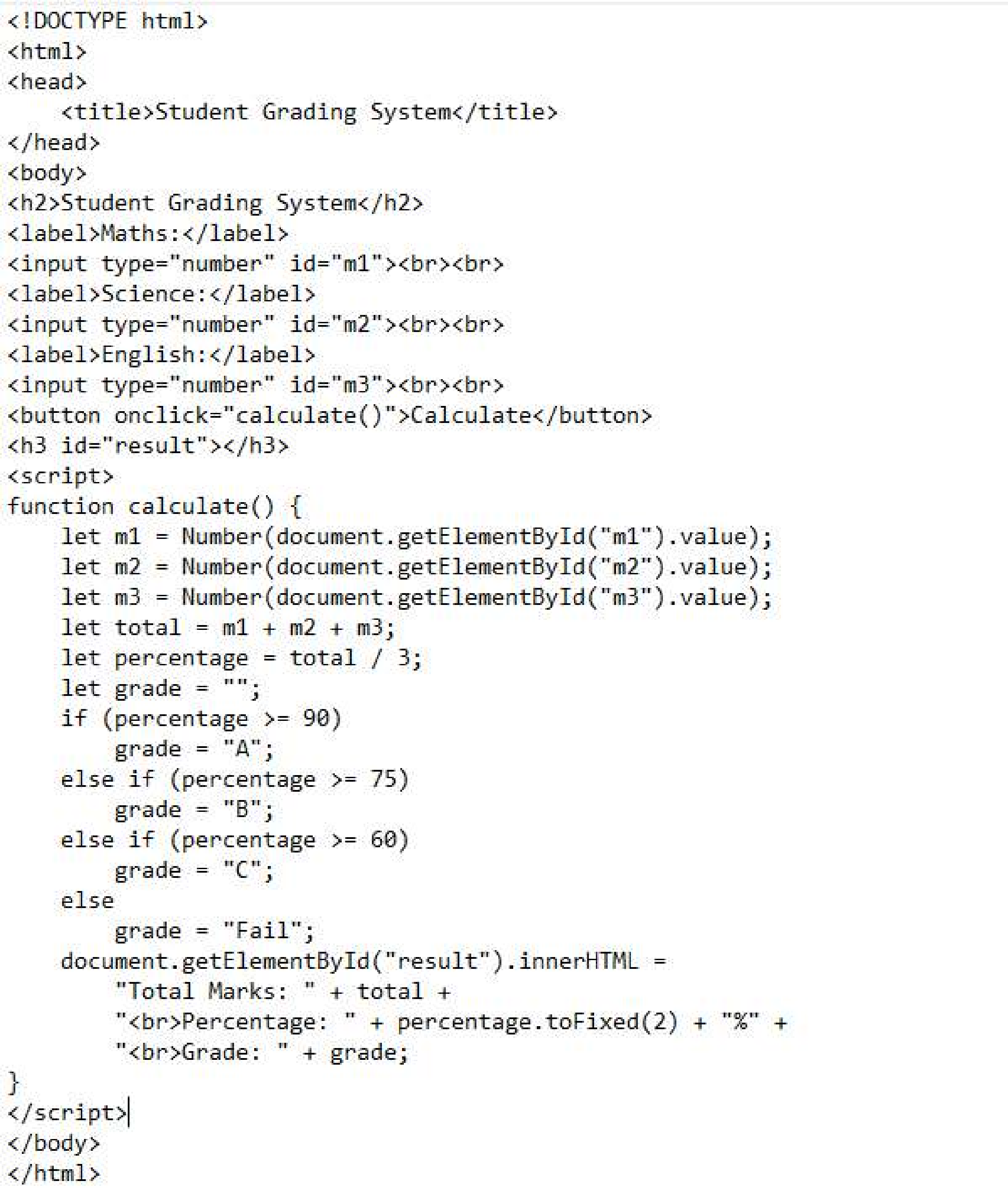
Well-structured UI code with accurate calculations and clear output display.

**Prompt:**

Create a simple user interface for a student grading system.

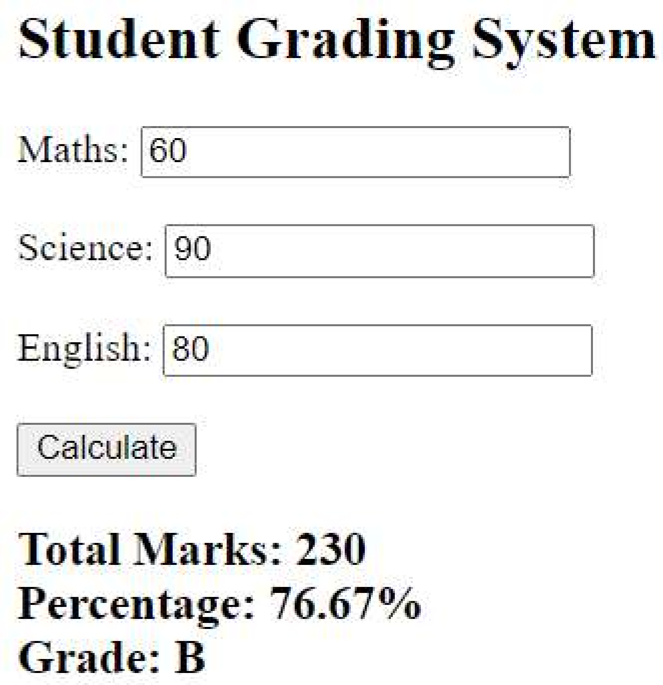
The UI should take student marks as input, calculate total marks, percentage, and grade,and clearly display the results. Use accurate calculations and a clean layout.

**CODE :**



```
<!DOCTYPE html>
<html>
<head>
    <title>Student Grading System</title>
</head>
<body>
<h2>Student Grading System</h2>
<label>Maths:</label>
<input type="number" id="m1"><br><br>
<label>Science:</label>
<input type="number" id="m2"><br><br>
<label>English:</label>
<input type="number" id="m3"><br><br>
<button onclick="calculate()">Calculate</button>
<h3 id="result"></h3>
<script>
function calculate() {
    let m1 = Number(document.getElementById("m1").value);
    let m2 = Number(document.getElementById("m2").value);
    let m3 = Number(document.getElementById("m3").value);
    let total = m1 + m2 + m3;
    let percentage = total / 3;
    let grade = "";
    if (percentage >= 90)
        grade = "A";
    else if (percentage >= 75)
        grade = "B";
    else if (percentage >= 60)
        grade = "C";
    else
        grade = "Fail";
    document.getElementById("result").innerHTML =
        "Total Marks: " + total +
        "<br>Percentage: " + percentage.toFixed(2) + "%" +
        "<br>Grade: " + grade;
}
</script>
</body>
</html>
```

**OUTPUT:**



Student Grading System

Maths: 60

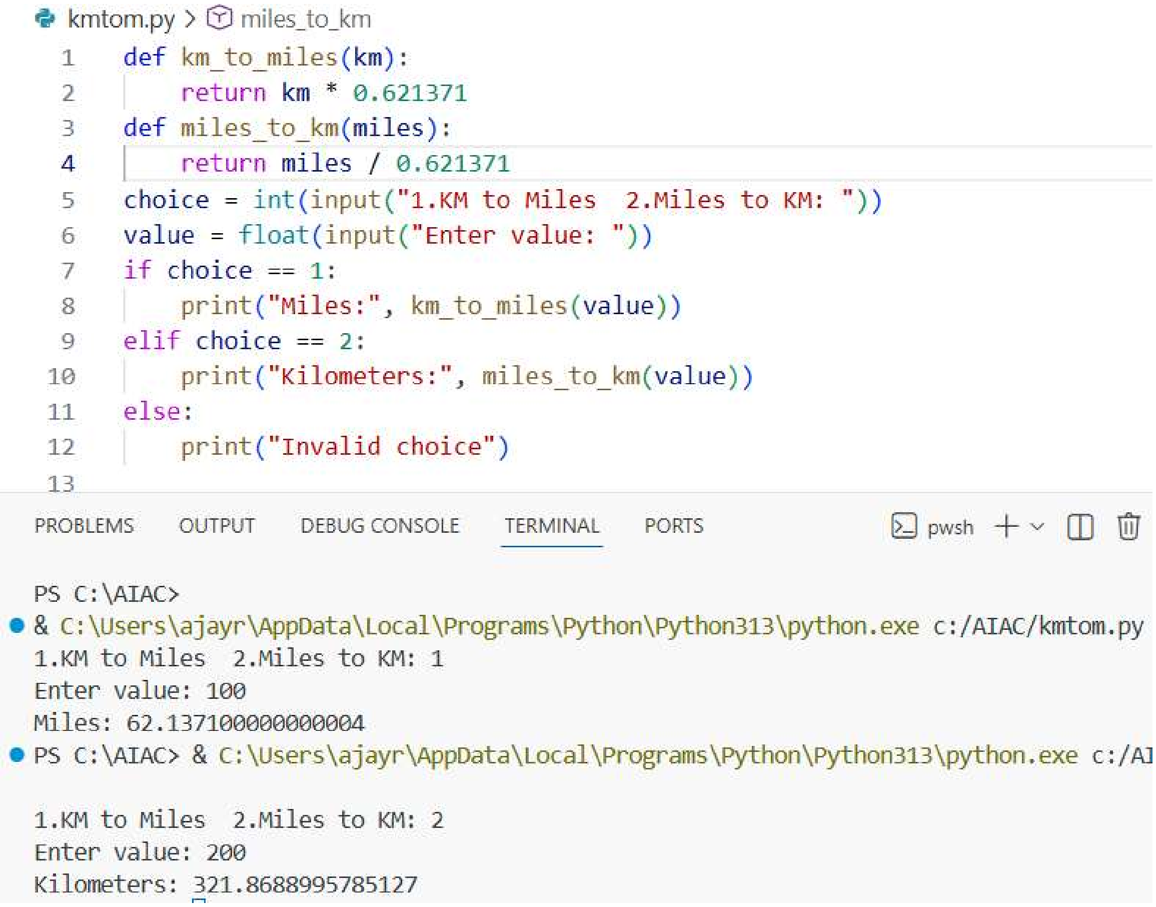
Science: 90

English: 80

Total Marks: 230
Percentage: 76.67%
Grade: B

**Explanation:**
The user interface takes marks for different subjects through input fields. When the Calculate button is clicked, JavaScript reads the entered values and computes the total marks and percentage. Based on the percentage, the program assigns a grade using conditional statements. Finally, the calculated total, percentage, and grade are displayed clearly on the screen.

**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:**
Analysis of code quality and accuracy differences across multiple prompt variations.
**Prompt:**
Design a Python program that takes user input and converts:
kilometers to miles,miles to kilometers. Display accurate results with clear output
**CODE AND OUTPUT:**



```python
def km_to_miles(km):
    return km * 0.621371
def miles_to_km(miles):
    return miles / 0.621371
choice = int(input("1.KM to Miles  2.Miles to KM: "))
value = float(input("Enter value: "))
if choice == 1:
    print("Miles:", km_to_miles(value))
elif choice == 2:
    print("Kilometers:", miles_to_km(value))
else:
    print("Invalid choice")
```

```
PS C:\AIAC>
& C:\Users\ajayr\AppData\Local\Programs\Python\Python313\python.exe c:/AIAC/kmtom.py
1.KM to Miles  2.Miles to KM: 1
Enter value: 100
Miles: 62.137100000000004
PS C:\AIAC> & C:\Users\ajayr\AppData\Local\Programs\Python\Python313\python.exe c:/AI

1.KM to Miles  2.Miles to KM: 2
Enter value: 200
Kilometers: 321.8688995785127
```

**Explanation**:
The clearer the prompt, the better the code. A vague prompt can give incomplete or wrong functions, but a well-explained prompt helps create accurate, easy-to-use conversion functions that handle user input and show results clearly.