2303A51203
Batch : 04

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

Prompt :

Give me a simple calculator program keep the function name as calculator_program

```
def calculator_program():
    """
    A simple calculator program that performs basic arithmetic operations.
    """
    print("Simple Calculator")
    while True:
        try:
            num1 = float(input("Enter first number: "))
            operator = input("Enter operator (+, -, *, /) or 'q' to quit: ")
            if operator.lower() == 'q':
                print("Exiting calculator. Goodbye!")
                break
            num2 = float(input("Enter second number: "))


            if operator == '+':
                result = num1 + num2
            elif operator == '-':
                result = num1 - num2
            elif operator == '*':
                result = num1 * num2
            elif operator == '/':
                if num2 == 0:
                    print("Error: Division by zero is not allowed.")
                    continue
                result = num1 / num2
            else:
                print("Error: Invalid operator. Please use +, -, *, or /.")
                continue
            print(f"Result: {num1} {operator} {num2} = {result}")
        except ValueError:
            print("Error: Invalid input. Please enter numbers for calculations.")
        except Exception as e:
```

```
        print(f"An unexpected error occurred: {e}")


# To run the calculator, uncomment the line below:
calculator_program()
```

Output :

```
Simple Calculator
Enter first number: 20
Enter operator (+, -, *, /) or 'q' to quit: +
Enter second number: 14
Result: 20.0 + 14.0 = 34.0
Enter first number: 29
Enter operator (+, -, *, /) or 'q' to quit: +
Enter second number: 14
Result: 29.0 + 14.0 = 43.0
```

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

Prompt:

Give me program for sorting list of student marks

Code :

```
# Example list of student marks
student_marks = [85, 92, 78, 65, 95, 88, 70, 90]

print(f"Original list of marks: {student_marks}")

# Sort the list of marks in ascending order
sorted_marks_ascending = sorted(student_marks)
print(f"Sorted marks (ascending): {sorted_marks_ascending}")

# To sort in descending order, you can use the 'reverse=True' argument
sorted_marks_descending = sorted(student_marks, reverse=True)
print(f"Sorted marks (descending): {sorted_marks_descending}")

# If you want to sort the list in-place, you can use the .sort() method:
# student_marks.sort() # This will modify student_marks directly
```

Output:

```
Original list of marks: [85, 92, 78, 65, 95, 88, 70, 90]
Sorted marks (ascending): [65, 70, 78, 85, 88, 90, 92, 95]
Sorted marks (descending): [95, 92, 90, 88, 85, 78, 70, 65]
```

Prompt : Make it efficient to sort larger inputs and make it more accurate

```
# Example list of student marks
student_marks = [85, 92, 78, 65, 95, 88, 70, 90]

print(f"Original list of marks: {student_marks}")

# Sort the list of marks in ascending order using sorted()
# This creates a *new* sorted list, leaving the original unchanged.
sorted_marks_ascending = sorted(student_marks)
print(f"Sorted marks (ascending, using sorted()): {sorted_marks_ascending}")

# To sort in descending order using sorted(), use the 'reverse=True' argument
sorted_marks_descending = sorted(student_marks, reverse=True)
print(f"Sorted marks (descending, using sorted()): {sorted_marks_descending}")

# To sort the list in-place, use the .sort() method:
# This modifies the original list and returns None.
print(f"\nOriginal list before in-place sort: {student_marks}")
student_marks.sort() # Sorts in ascending order by default
print(f"In-place sorted marks (ascending, using .sort()): {student_marks}")

# You can also sort in-place in descending order:
student_marks.sort(reverse=True)
print(f"In-place sorted marks (descending, using .sort(reverse=True)): {student_marks}")
```

Output :

```
Original list before in-place sort: [85, 92, 78, 65, 95, 88, 70, 90]
In-place sorted marks (ascending, using .sort()): [65, 70, 78, 85, 88, 90, 92, 95]
In-place sorted marks (descending, using .sort(reverse=True)): [95, 92, 90, 88, 85, 78, 70, 65]
```

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:
Give me a function to check given number is prime or not

```
def is_prime(number):
    """
    Checks if a given number is prime.


    A prime number is a natural number greater than 1 that has no positive
    divisors other than 1 and itself.
    """
    if number <= 1:
        return False  # Numbers less than or equal to 1 are not prime
    if number == 2:
        return True   # 2 is the only even prime number
    if number % 2 == 0:
        return False  # Other even numbers are not prime


    # Check for divisibility from 3 up to the square root of the number
    # We only need to check odd divisors
    i = 3
    while i * i <= number:
        if number % i == 0:
            return False
        i += 2
    return True


# Example usage:
print(f"Is 7 prime? {is_prime(7)}")       # Expected: True
print(f"Is 10 prime? {is_prime(10)}")     # Expected: False
print(f"Is 11 prime? {is_prime(11)}")     # Expected: True
print(f"Is 1 prime? {is_prime(1)}")       # Expected: False
print(f"Is 2 prime? {is_prime(2)}")       # Expected: True
print(f"Is 0 prime? {is_prime(0)}")       # Expected: False
print(f"Is 97 prime? {is_prime(97)}")     # Expected: True
```

Output :

```
Is 7 prime? True
Is 10 prime? False
Is 11 prime? True
Is 1 prime? False
Is 2 prime? True
Is 0 prime? False
Is 97 prime? True
```

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.

```
def student_grade_calculator():
    print("\n--- Student Grade Calculator ---")

    # Get student name
    student_name = input("Enter student's name: ")

    # Get marks for 5 subjects
    marks = []
    subject_names = ["Subject 1", "Subject 2", "Subject 3", "Subject 4", "Subject 5"]
    for i in range(5):
        while True:
            try:
                mark = float(input(f"Enter marks for {subject_names[i]} (out of 100): "))
                if 0 <= mark <= 100:
                    marks.append(mark)
                    break
                else:
                    print("Marks must be between 0 and 100.")
            except ValueError:
                print("Invalid input. Please enter a number.")

    # Calculate total marks
    total_marks = sum(marks)

    # Calculate percentage
    # Assuming each subject is out of 100, total possible marks = 5 * 100 = 500
    percentage = (total_marks / 500) * 100

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

    # Display results
    print("\n--- Student Report ---")
    print(f"Student Name: {student_name}")
    print(f"Marks Obtained: {marks}")
    print(f"Total Marks: {total_marks}")
    print(f"Percentage: {percentage:.2f}%")
    print(f"Grade: {grade}")
    print("---------------------")

# Run the student grade calculator
student_grade_calculator()
```

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.

```
def km_to_miles(kilometers):
    """
    Converts kilometers to miles.

    Args:
        kilometers (float or int): The distance in kilometers.

    Returns:
        float: The equivalent distance in miles.
    """
    miles = kilometers * 0.621371
    return miles

def miles_to_km(miles):
    """
    Converts miles to kilometers.

    Args:
        miles (float or int): The distance in miles.
    Returns:
        float: The equivalent distance in kilometers.
    """
    kilometers = miles * 1.60934
    return kilometers

# Test cases
print(f"10 kilometers is equal to {km_to_miles(10):.2f} miles.")
print(f"5 miles is equal to {miles_to_km(5):.2f} kilometers.")
print(f"1 kilometer is equal to {km_to_miles(1):.2f} miles.")
print(f"1 mile is equal to {miles_to_km(1):.2f} kilometers.")
```