

School of Computer Science and Artificial Intelligence

Lab Assignment #3.2

Program : B. Tech (CSE)

Specialization : AIML

Course Title : AI Assisted Coding

Course Code : 23CS002PC304

Semester : VI

Academic Session : 2025-2026

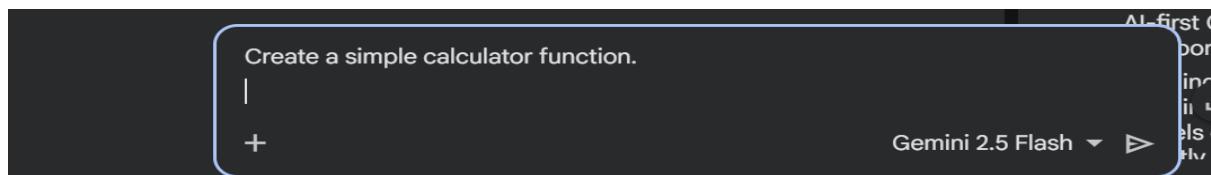
Name of Student : CH.Ramcharan

Enrollment No. : 2303A52095

Batch No. : 33

Task 1 – Progressive Prompting (Calculator)

Prompt1:



```
▶ def calculator(num1, num2, operator):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        num1 (float): The first number.
        num2 (float): The second number.
        operator (str): The operation to perform ('+', '-', '*', '/').

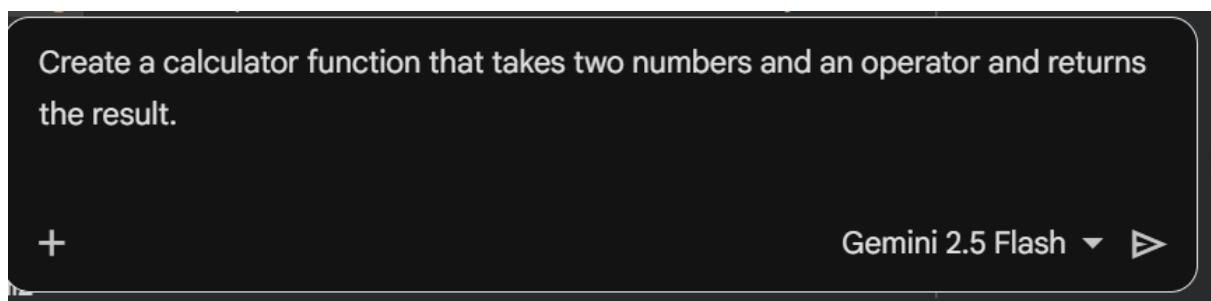
    Returns:
        float: The result of the operation.
        str: An error message if the operation is invalid or division by zero occurs.
    """
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            return "Error: Division by zero is not allowed."
        return num1 / num2
```

```
▶     return num1 / num2
else:
    return "Error: Invalid operator. Please use '+', '-', '*', or '/'.

# Example usage:
print(f"3 + 5 = {calculator(3, 5, '+')}")
print(f"10 - 4 = {calculator(10, 4, '-')}")
print(f"7 * 6 = {calculator(7, 6, '*')}")
print(f"20 / 5 = {calculator(20, 5, '/')}")
print(f"10 / 0 = {calculator(10, 0, '/')}")
print(f"5 ^ 2 = {calculator(5, 2, '^')}")

...
3 + 5 = 8
10 - 4 = 6
7 * 6 = 42
20 / 5 = 4.0
10 / 0 = Error: Division by zero is not allowed.
5 ^ 2 = Error: Invalid operator. Please use '+', '-', '*', or '/'.
```

Prompt2:



```
def calculator(num1, num2, operator):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        num1 (float): The first number.
        num2 (float): The second number.
        operator (str): The operation to perform ('+', '-', '*', '/').

    Returns:
        float: The result of the operation.
    - str: An error message if the operation is invalid or division by zero occurs.

    Raises:
    + ValueError: If an invalid operator is provided.
    + ZeroDivisionError: If division by zero is attempted.
    """
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            return "Error: Division by zero is not allowed."
        else:
            return num1 / num2
    else:
        return "Error: Invalid operator. Please use '+', '-', '*', or '/'"

# Example usage:
# Example usage with improved error handling:
print(f"3 + 5 = {calculator(3, 5, '+')}")
print(f"10 - 4 = {calculator(10, 4, '-')}")
print(f"7 * 6 = {calculator(7, 6, '*')}")
print(f"20 / 5 = {calculator(20, 5, '/')}")
print(f"10 / 0 = {calculator(10, 0, '/')}")
print(f"5 ^ 2 = {calculator(5, 2, '^')}")


try:
    print(f"10 / 0 = {calculator(10, 0, '/')}")
except ZeroDivisionError as e:
    print(f"10 / 0 = Error: {e}")

try:
    print(f"5 ^ 2 = {calculator(5, 2, '^')}")
except ValueError as e:
    print(f"5 ^ 2 = Error: {e}")
```

```
def calculator(num1, num2, operator):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        num1 (float): The first number.
        num2 (float): The second number.
        operator (str): The operation to perform ('+', '-', '*', '/').

    Returns:
        float: The result of the operation.
    - str: An error message if the operation is invalid or division by zero occurs.

    Raises:
    + ValueError: If an invalid operator is provided.
    + ZeroDivisionError: If division by zero is attempted.
    """
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            raise ZeroDivisionError("Division by zero is not allowed.")
        else:
            return num1 / num2
    else:
        raise ValueError("Invalid operator. Please use '+', '-', '*', or '/'")

# Example usage:
# Example usage with improved error handling:
print(f"3 + 5 = {calculator(3, 5, '+')}")
print(f"10 - 4 = {calculator(10, 4, '-')}")
print(f"7 * 6 = {calculator(7, 6, '*')}")
print(f"20 / 5 = {calculator(20, 5, '/')}")
print(f"10 / 0 = {calculator(10, 0, '/')}")
print(f"5 ^ 2 = {calculator(5, 2, '^')}")
```

```
def calculator(num1, num2, operator):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        num1 (float): The first number.
        num2 (float): The second number.
        operator (str): The operation to perform ('+', '-', '*', '/').

    Returns:
        float: The result of the operation.
    - str: An error message if the operation is invalid or division by zero occurs.

    Raises:
    + ValueError: If an invalid operator is provided.
    + ZeroDivisionError: If division by zero is attempted.
    """
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            raise ZeroDivisionError("Division by zero is not allowed.")
        else:
            return num1 / num2
    else:
        raise ValueError("Invalid operator. Please use '+', '-', '*', or '/'")

# Example usage:
# Example usage with improved error handling:
print(f"3 + 5 = {calculator(3, 5, '+')}")
print(f"10 - 4 = {calculator(10, 4, '-')}")
print(f"7 * 6 = {calculator(7, 6, '*')}")
print(f"20 / 5 = {calculator(20, 5, '/')}")
print(f"10 / 0 = {calculator(10, 0, '/')}")
print(f"5 ^ 2 = {calculator(5, 2, '^')}")
```

Prompt 3:

▷ Accept and run ✓ Accept X Cancel

Design a calculator function with proper error handling.

Input: two numbers and an operator

Output: calculated result

Example: calculator(10, 5, '+') → 15

+

Gemini 2.5 Flash ▶

Gemini can make mistakes, so double-check it and use code with caution. [Learn more](#)

```
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all ▾
16 0s
def calculator(num1, num2, operator):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        num1 (float): The first number.
        num2 (float): The second number.
        operator (str): The operation to perform ('+', '-', '*', '/').

    Returns:
        float: The result of the operation.

    Raises:
        ValueError: If an invalid operator is provided.
        ZeroDivisionError: If division by zero is attempted.
    """
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        return num1 / num2
```

```
[6] 0s
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            raise ZeroDivisionError("Division by zero is not allowed.")
        return num1 / num2
    else:
        raise ValueError("Invalid operator. Please use '+', '-', '*', or '/'.")

# Example usage with improved error handling:
print(f"3 + 5 = {calculator(3, 5, '+')}")
print(f"10 - 4 = {calculator(10, 4, '-')}")
print(f"7 * 6 = {calculator(7, 6, '*')}")
print(f"20 / 5 = {calculator(20, 5, '/')}")

try:
    print(f"10 / 0 = {calculator(10, 0, '/')}")
except ZeroDivisionError as e:
    print(f"10 / 0 = Error: {e}")

try:
    print(f"5 ^ 2 = {calculator(5, 2, '^')}")
except ValueError as e:
    print(f"5 ^ 2 = Error: {e}")
```

```
[7] ... 3 + 5 = 8
      10 - 4 = 6
      7 * 6 = 42
      20 / 5 = 4.0
      10 / 0 = Error: Division by zero is not allowed.
      5 ^ 2 = Error: Invalid operator. Please use '+', '-', '*', or '/'
```

Task 2 – Sorting Student Marks

Initial Prompt:

Write a program to sort student marks.

```
[7] ✓ 0s C # Define a list of student marks (unsorted)
student_marks = [85, 92, 78, 65, 95, 88, 70, 81]

print("Original marks:", student_marks)

# Sort the marks in ascending order using sorted()
sorted_marksAscending = sorted(student_marks)
print("Marks sorted in ascending order:", sorted_marksAscending)

# Sort the marks in descending order using sorted() with reverse=True
sorted_marksDescending = sorted(student_marks, reverse=True)
print("Marks sorted in descending order:", sorted_marksDescending)

# Demonstrate in-place sorting using .sort()
# Note: .sort() modifies the list directly and returns None
marksForInplaceSort = [85, 92, 78, 65, 95, 88, 70, 81]
print("\nOriginal marks for in-place sort:", marksForInplaceSort)
marksForInplaceSort.sort()
print("Marks sorted in-place (ascending):", marksForInplaceSort)
```

```
[7] ... Original marks: [85, 92, 78, 65, 95, 88, 70, 81]
      Marks sorted in ascending order: [65, 70, 78, 81, 85, 88, 92, 95]
      Marks sorted in descending order: [95, 92, 88, 85, 81, 78, 70, 65]

      Original marks for in-place sort: [85, 92, 78, 65, 95, 88, 70, 81]
      Marks sorted in-place (ascending): [65, 70, 78, 81, 85, 88, 92, 95]
```

Refined Prompt:

Write a Python function to sort student marks in descending order without modifying the original list.

```

  def sort_marks_descending(marks_list):
    """
    Sorts a list of student marks in descending order without modifying the original list.

    Args:
        marks_list (list): A list of student marks (numbers).

    Returns:
        list: A new list with marks sorted in descending order.
    """
    return sorted(marks_list, reverse=True)

# Example usage:
original_marks = [85, 92, 78, 65, 95, 88, 70, 81]
print("Original marks before sorting:", original_marks)

descending_marks = sort_marks_descending(original_marks)
print("Marks sorted in descending order (new list):", descending_marks)
print("Original marks after sorting (unchanged):", original_marks)

```

```
print("Original marks after sorting (unchanged):", original_marks)
```

```
Original marks before sorting: [85, 92, 78, 65, 95, 88, 70, 81]
Marks sorted in descending order (new list): [95, 92, 88, 85, 81, 78, 70, 65]
Original marks after sorting (unchanged): [85, 92, 78, 65, 95, 88, 70, 81]
```

Task 3 – Few-Shot Prompting (Prime Number)

Create a function `is_prime(n)` that checks whether a number is prime.
 Examples: `is_prime(2) → True` `is_prime(4) → False` `is_prime(17) → True`
`is_prime(1) → False`

```

import math

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

    Args:
        n (int): The number to check.

    Returns:
        bool: True if the number is prime, False otherwise.
    """
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(math.sqrt(n)) + 1, 2):
        if n % i == 0:
            return False
    return True

```



```
# Examples:  
print(f"is_prime(2) -> {is_prime(2)}")  
print(f"is_prime(4) -> {is_prime(4)}")  
print(f"is_prime(17) -> {is_prime(17)}")  
print(f"is_prime(1) -> {is_prime(1)}")  
print(f"is_prime(23) -> {is_prime(23)}")  
print(f"is_prime(25) -> {is_prime(25)}")  
  
... is_prime(2) -> True  
is_prime(4) -> False  
is_prime(17) -> True  
is_prime(1) -> False  
is_prime(23) -> True  
is_prime(25) -> False
```

Task 4 – Prompt-Guided UI Design

Create a user interface program in Python that accepts student name and marks of 5 subjects, calculates total marks, percentage and displays grade.

```
10] 23s ⏪|student_name = input("Enter student's name: ")  
subject_marks = []  
  
for i in range(1, 6):  
    while True:  
        try:  
            mark_str = input(f"Enter mark for Subject {i} (0-100): ")  
            mark = float(mark_str)  
  
            if 0 <= mark <= 100:  
                subject_marks.append(mark)  
                break # Exit inner while loop if input is valid  
            else:  
                print("Invalid mark. Please enter a value between 0 and 100.")  
        except ValueError:  
            print("Invalid input. Please enter a numerical value for the mark.")  
  
print(f"\nStudent Name: {student_name}")  
print(f"Subject Marks: {subject_marks}")
```

```
... Enter student's name: sowmya
Enter mark for Subject 1 (0-100): 90
Enter mark for Subject 2 (0-100): 99
Enter mark for Subject 3 (0-100): 89
Enter mark for Subject 4 (0-100): 79
Enter mark for Subject 5 (0-100): 99

Student Name: sowmya
Subject Marks: [90.0, 99.0, 89.0, 79.0, 99.0]
```

```
total_marks = sum(subject_marks)
max_possible_marks = 5 * 100 # Assuming each subject is out of 100
percentage = (total_marks / max_possible_marks) * 100

print(f"Total Marks: {total_marks:.2f}")
print(f"Percentage: {percentage:.2f}%")
```

```
... Total Marks: 456.00
Percentage: 91.20%
```

```
▶ 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"Grade: {grade}")
```

```
... Grade: A
```

```
▶ print("\n--- Student Report ---")
print(f"Student Name: {student_name}")
print("Subject Marks:")
for i, mark in enumerate(subject_marks):
    print(f"  Subject {i+1}: {mark:.2f}")
print(f"Total Marks: {total_marks:.2f}")
print(f"Percentage: {percentage:.2f}%")
print(f"Grade: {grade}")
print("-----")  

...
--- Student Report ---
Student Name: sowmya
Subject Marks:
  Subject 1: 90.00
  Subject 2: 99.00
  Subject 3: 89.00
  Subject 4: 79.00
  Subject 5: 99.00
Total Marks: 456.00
Percentage: 91.20%
Grade: A
-----
```

Task 5 – Unit Conversion Analysis

Basic Prompt:

Empty cell X

Create a function to convert kilometers to miles.

|

+

Gemini 2.5 Flash ▾ ➤

```
▶ def km_to_miles(kilometers):
    """
    Converts a distance from kilometers to miles.

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

    Returns:
        float: The equivalent distance in miles.
    """
    # 1 kilometer is approximately 0.621371 miles
    miles = kilometers * 0.621371
    return miles

    # Example usage:
print(f"10 kilometers is {km_to_miles(10):.2f} miles.")
print(f"50 kilometers is {km_to_miles(50):.2f} miles.")
print(f"100 kilometers is {km_to_miles(100):.2f} miles.")

... 10 kilometers is 6.21 miles.
50 kilometers is 31.07 miles.
100 kilometers is 62.14 miles.
```

Improved Prompt:

1. km_to_miles(km)
 2. miles_to_km(miles)
- Use conversion rate: 1 km = 0.621371 miles
- Include sample outputs.

Gemini 2.5 Flash ▾ ➤

```
Returns:
    float: The equivalent distance in kilometers.
"""

# 1 mile is approximately 1 / 0.621371 kilometers
kilometers = miles / 0.621371
return kilometers

# Example usage:
print(f"10 kilometers is {km_to_miles(10):.2f} miles.")
print(f"50 kilometers is {km_to_miles(50):.2f} miles.")
print(f"100 kilometers is {km_to_miles(100):.2f} miles.")

print(f"\n10 miles is {miles_to_km(10):.2f} kilometers.")
print(f"31.07 miles is {miles_to_km(31.07):.2f} kilometers.")
print(f"62.14 miles is {miles_to_km(62.14):.2f} kilometers.")
```

```
Returns:  
    float: The equivalent distance in kilometers.  
"""  
# 1 mile is approximately 1 / 0.621371 kilometers  
kilometers = miles / 0.621371  
return kilometers  
  
# Example usage:  
print(f"10 kilometers is {km_to_miles(10):.2f} miles.")  
print(f"50 kilometers is {km_to_miles(50):.2f} miles.")  
print(f"100 kilometers is {km_to_miles(100):.2f} miles.")  
  
print(f"\n10 miles is {miles_to_km(10):.2f} kilometers.")  
print(f"31.07 miles is {miles_to_km(31.07):.2f} kilometers.")  
print(f"62.14 miles is {miles_to_km(62.14):.2f} kilometers.")
```



```
... 10 kilometers is 6.21 miles.  
50 kilometers is 31.07 miles.  
100 kilometers is 62.14 miles.  
  
10 miles is 16.09 kilometers.  
31.07 miles is 50.00 kilometers.  
62.14 miles is 100.00 kilometers.
```