

# **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 : k.sathwika**

**Enrollment No. : 2303A52096**

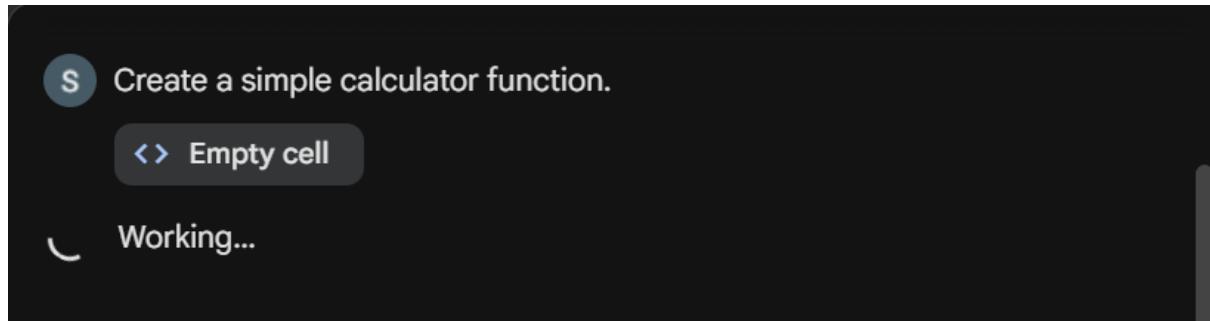
**Batch No. : 33**

**Date :13/01/26**

# TASK\_01: Progressive Prompting (Calculator)

STAGE 1

Prompt:



Code:

```
[10] 2096_Assignment 3.1.ipynb  ⭐ ⓘ
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all ▾

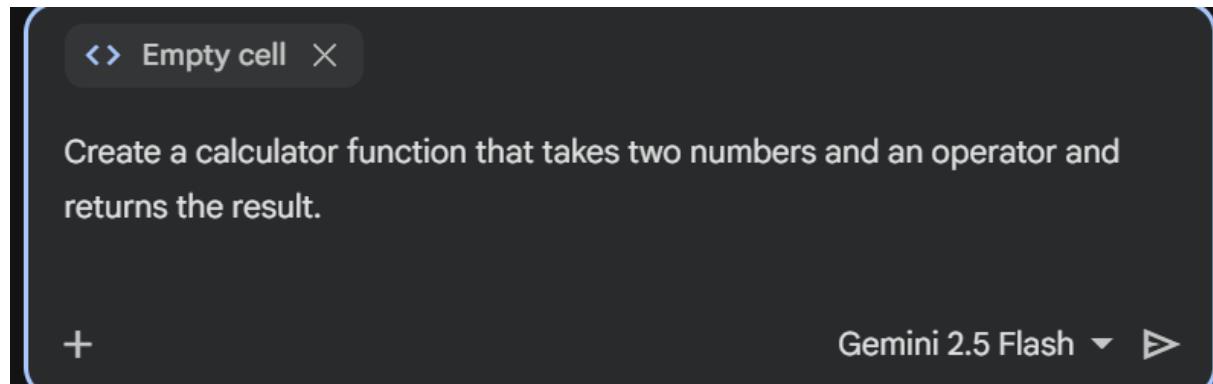
def calculator(num1, num2, operator):
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            return "Error: Cannot divide by zero!"
        return num1 / num2
    else:
        return "Error: Invalid operator. Please use '+', '-', '*', or '/'."
print(calculator(10, 5, '+'))
print(calculator(10, 3, '-'))
print(calculator(4, 6, '*'))
print(calculator(20, 4, '/'))
print(calculator(10, 0, '/'))
print(calculator(7, 2, '%'))
```

Output:

```
print(calculator(7, 2, '%'))
...
15
7
24
5.0
Error: Cannot divide by zero!
Error: Invalid operator. Please use '+', '-', '*', or '/'.
```

STAGE2:

Prompt:



Code:

```
▶ def calculator(num1, num2, operator):
    operations = {
        '+': lambda a, b: a + b,
        '-': lambda a, b: a - b,
        '*': lambda a, b: a * b,
        '/': lambda a, b: a / b,
        '%': lambda a, b: a % b
    }

    if operator == '/' and num2 == 0:
        return "Error: Cannot divide by zero!"

    if operator not in operations:
        return "Error: Invalid operator. Please use '+', '-', '*', '/', or '%'."

    return operations[operator](num1, num2)

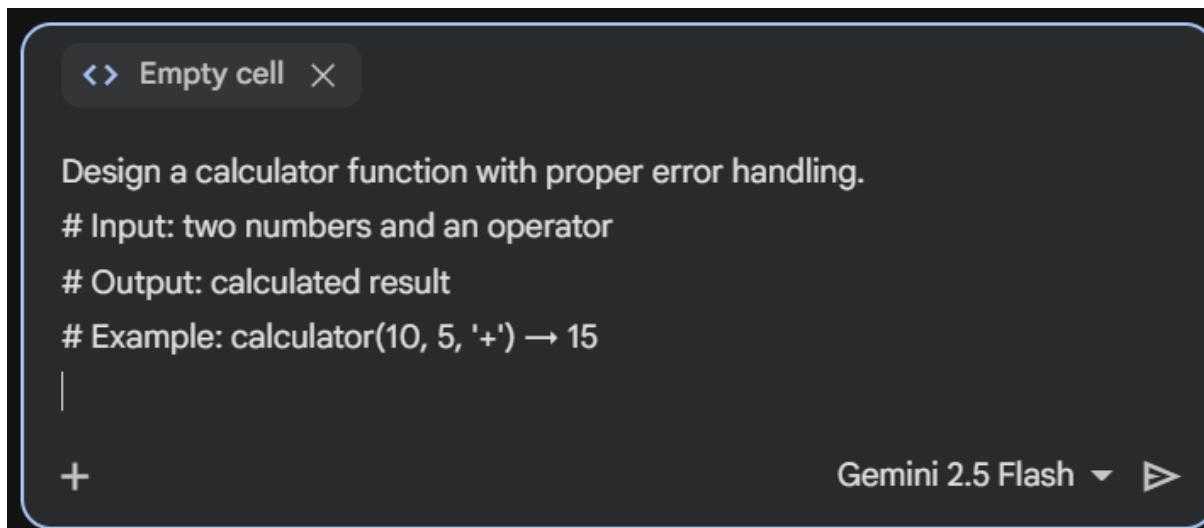
# Example usage with improved function:
print(f"10 + 5 = {calculator(10, 5, '+')}")
print(f"10 - 3 = {calculator(10, 3, '-')}")
print(f"4 * 6 = {calculator(4, 6, '*')}")
print(f"20 / 4 = {calculator(20, 4, '/')}")
print(f"10 % 3 = {calculator(10, 3, '%')}")
print(f"10 / 0 = {calculator(10, 0, '/')}")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

Output:

```
... 10 + 5 = 15
10 - 3 = 7
4 * 6 = 24
20 / 4 = 5.0
10 % 3 = 1
10 / 0 = Error: Cannot divide by zero!
7 ^ 2 = Error: Invalid operator. Please use '+', '-', '*', '/', or '%'.
```

STAGE 3:

Prompt:



Code:

```
[14]  def calculator(num1, num2, operator):
    operations = {
        '+': lambda a, b: a + b,
        '-': lambda a, b: a - b,
        '*': lambda a, b: a * b,
        '/': lambda a, b: a / b,
        '%': lambda a, b: a % b
    }

    if operator == '/' and num2 == 0:
        return "Error: Cannot divide by zero!"

    if operator not in operations:
        return "Error: Invalid operator. Please use '+', '-', '*', '/', or '%'."

    return operations[operator](num1, num2)

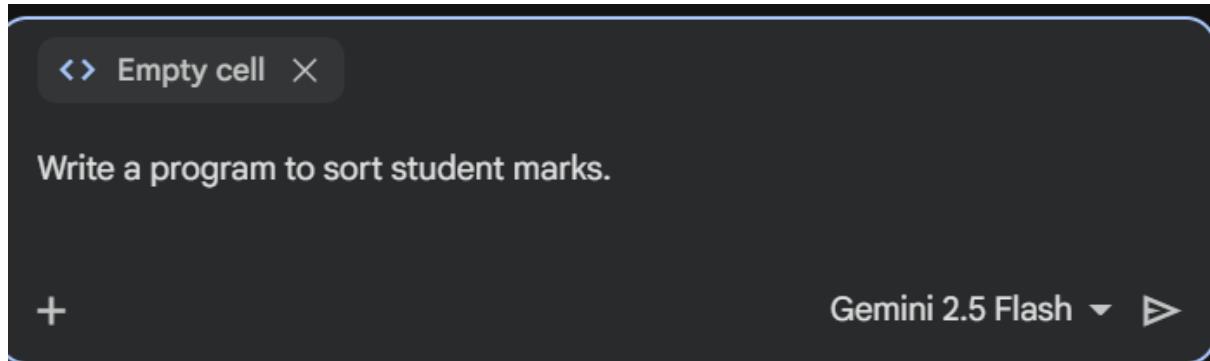
# Example usage:
print(f"calculator(10, 5, '+') -> {calculator(10, 5, '+')}") # 15
print(f"calculator(10, 3, '-') -> {calculator(10, 3, '-')}") # 7
print(f"calculator(4, 6, '*') -> {calculator(4, 6, '*')}") # 24
print(f"calculator(20, 4, '/') -> {calculator(20, 4, '/')}") # 5.0
print(f"calculator(10, 3, '%') -> {calculator(10, 3, '%')}") # 1
print(f"calculator(10, 0, '/') -> {calculator(10, 0, '/')}") # Error: Cannot divide by zero!
print(f"calculator(7, 2, '^') -> {calculator(7, 2, '^')}") # Error: Invalid operator. Please use '+', '-', '*', '/', or '%'.
```

Output:

```
... calculator(10, 5, '+') -> 15
calculator(10, 3, '-') -> 7
calculator(4, 6, '*') -> 24
calculator(20, 4, '/') -> 5.0
calculator(10, 3, '%') -> 1
calculator(10, 0, '/') -> Error: Cannot divide by zero!
calculator(7, 2, '^') -> Error: Invalid operator. Please use '+', '-', '*', '/', or '%'.
```

## TASK\_02: Sorting Student Marks

Initial prompt:



Code:

```
# Define a list of student marks
student_marks = [85, 92, 78, 65, 95, 88, 70, 81]

print("Original student marks:", student_marks)

# Sort the student marks in ascending order
student_marks.sort()

print("Sorted student marks (ascending):", student_marks)

# To sort in descending order, you can use:
# student_marks.sort(reverse=True)
# print("Sorted student marks (descending):", student_marks)
```

Output:

```
... Original student marks: [85, 92, 78, 65, 95, 88, 70, 81]
Sorted student marks (ascending): [65, 70, 78, 81, 85, 88, 92, 95]
```

Refined prompt:

Empty cell X

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

+

Gemini 2.5 Flash ▾ ➔

Code:

```
def sort_student_marks_descending(marks_list):
    # Create a copy of the original list to avoid modifying it
    sorted_marks = sorted(marks_list, reverse=True)
    return sorted_marks

# Define a list of student marks (using the existing student_marks for demonstration)
# student_marks = [85, 92, 78, 65, 95, 88, 70, 81] # If starting fresh

print("Original student marks:", student_marks)

# Sort the student marks in descending order using the new function
descending_marks = sort_student_marks_descending(student_marks)

print("Sorted student marks (descending, new list):", descending_marks)
print("Original list after sorting function call:", student_marks)
```

Output:

```
... Original student marks: [65, 70, 78, 81, 85, 88, 92, 95]
      Sorted student marks (descending, new list): [95, 92, 88, 85, 81, 78, 70, 65]
      Original list after sorting function call: [65, 70, 78, 81, 85, 88, 92, 95]
```

## TASK\_03: Few-Shot Prompting (Prime Number)

Prompt:

Explain the 'calculator' function in more detail

Show how to find the average of the sorted s

Empty cell X

Create a function `is_prime(n)` that checks whether a number is prime.

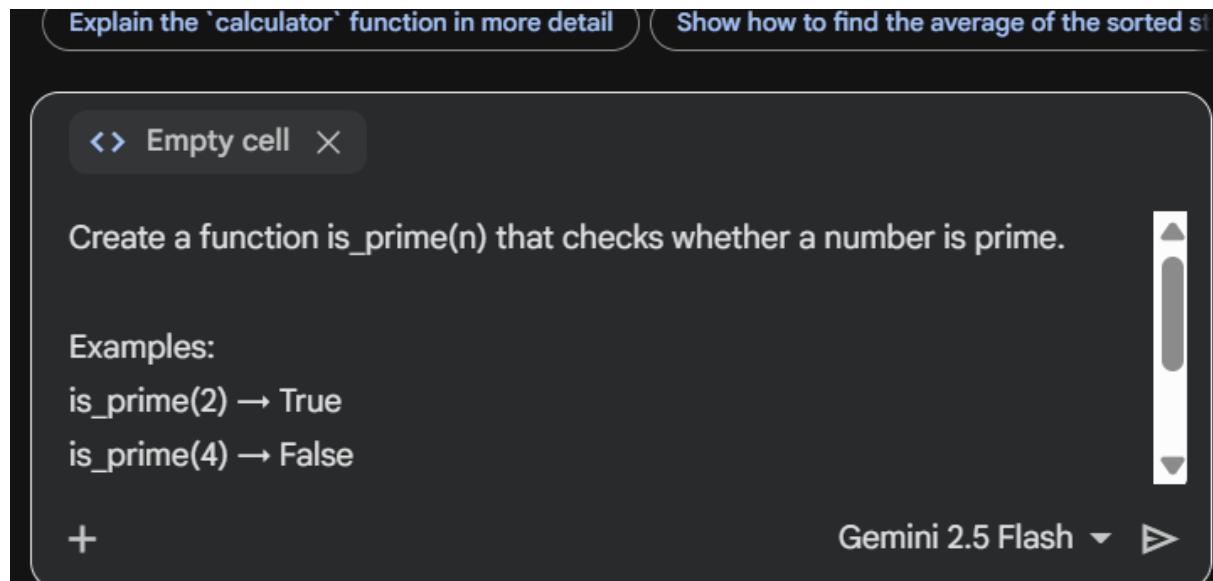
Examples:

`is_prime(2) → True`

`is_prime(4) → False`

+

Gemini 2.5 Flash ▾ ➔



Code:

The screenshot shows a Jupyter Notebook interface with a dark theme. The file name is '2096\_Assignment 3.1.ipynb'. The code cell contains the following Python script:

```
[19] ✓ 0s
import math

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

    # Check for divisors from 3 up to sqrt(n), skipping even numbers
    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2
    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)}")
```

Output:

```
*** is_prime(2) -> True
is_prime(4) -> False
is_prime(17) -> True
is_prime(1) -> False
```

## TASK\_04: Prompt-Guided UI Design

Prompt:

A task prompt with a blue circular icon containing a white letter 'S'. The text reads: 'Create a user interface program in Python that accepts student name and marks of 5 subjects, calculates total marks, percentage and displays grade.' Below this is a button labeled 'Empty cell'.

Code:

```
❶ def student_grade_calculator_ui():
    print("--- Student Marksheets and Grade Calculator ---")

    # 1. Accept student name
    student_name = input("Enter student name: ")

    marks = []
    num_subjects = 5
    for i in range(num_subjects):
        while True:
            try:
                mark = float(input(f"Enter marks for subject {i + 1} (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 for marks.")

    # 2. Calculate total marks
    total_marks = sum(marks)

    # 3. Calculate percentage
    max_total_marks = num_subjects * 100 # Assuming each subject is out of 100
    percentage = (total_marks / max_total_marks) * 100
```

```
❷ # 4. Determine grade
grade = ''
if percentage >= 90:
    grade = 'A+'
elif percentage >= 80:
    grade = 'A'
elif percentage >= 70:
    grade = 'B'
elif percentage >= 60:
    grade = 'C'
elif percentage >= 50:
    grade = 'D'
else:
    grade = 'F'

❸ # 5. Display results
print("\n--- Results ---")
print(f"Student Name: {student_name}")
print(f"Marks Obtained: {', '.join(map(str, marks))}")
print(f"Total Marks: {total_marks}/{max_total_marks}")
print(f"Percentage: {percentage:.2f}%")
print(f"Grade: {grade}")
print("-----")

# Call the function to run the UI program
student_grade_calculator_ui()
```

Output:

```
... --- Student Marksheets and Grade Calculator ---  
Enter student name: sathwika  
Enter marks for subject 1 (out of 100): 96  
Enter marks for subject 2 (out of 100): 83  
Enter marks for subject 3 (out of 100): 99  
Enter marks for subject 4 (out of 100): 78  
Enter marks for subject 5 (out of 100): 65
```

```
--- Results ---  
Student Name: sathwika  
Marks Obtained: 96.0, 83.0, 99.0, 78.0, 65.0  
Total Marks: 421.0/500  
Percentage: 84.20%  
Grade: A  
-----
```

## TASK\_05: Unit Conversion Analysis

**Basic Prompt:**

The screenshot shows the Gemini 2.5 Flash interface. At the top, there are two circular buttons: "Add functionality to calculate average marks for the student" on the left and "Refactor the student grade calc" on the right. Below these is a large input field containing the text: "Create a function to convert kilometers to miles." To the left of the input field is a plus sign (+). To the right is a button labeled "Gemini 2.5 Flash" with a dropdown arrow and a send icon. At the bottom of the input field, a note says: "Gemini can make mistakes, so double-check it and use code with caution. [Learn more](#)". To the right of this note is a "Send" button.

Code:

```
[21] 0s ➔ def kilometers_to_miles(km):
    conversion_factor = 0.621371 # 1 kilometer = 0.621371 miles
    miles = km * conversion_factor
    return miles

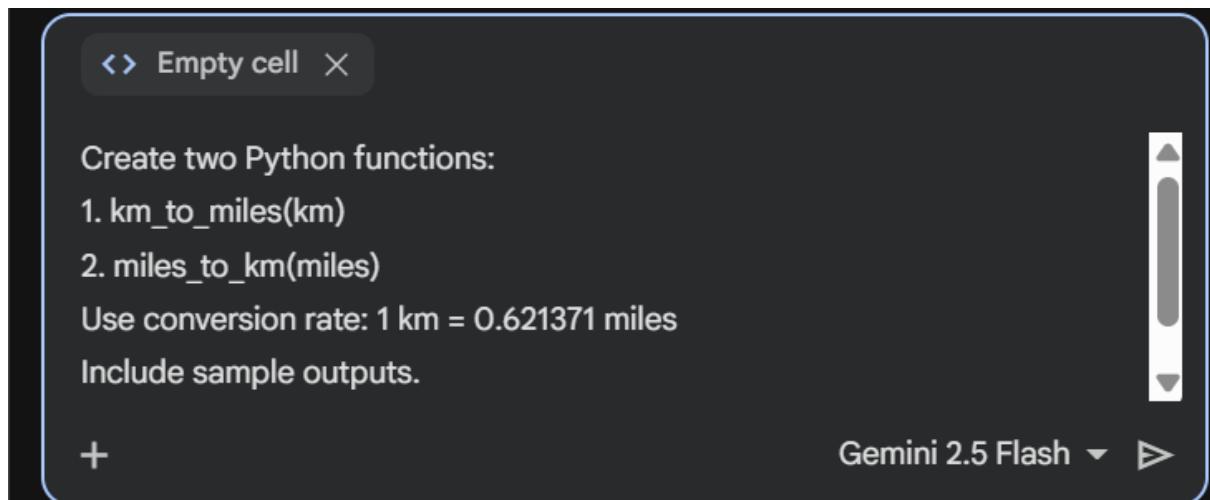
# Example usage:
kilometers = 10
milesConverted = kilometers_to_miles(kilometers)
print(f"{kilometers} kilometers is equal to {milesConverted:.2f} miles")

kilometers = 50
milesConverted = kilometers_to_miles(kilometers)
print(f"{kilometers} kilometers is equal to {milesConverted:.2f} miles")
```

Output:

```
... 10 kilometers is equal to 6.21 miles
50 kilometers is equal to 31.07 miles
```

Improved Prompt:



Code:

```
▶ def km_to_miles(km):
    conversion_factor = 0.621371 # 1 kilometer = 0.621371 miles
    miles = km * conversion_factor
    return miles

def miles_to_km(miles):
    conversion_factor = 0.621371 # 1 kilometer = 0.621371 miles
    km = miles / conversion_factor
    return km

# Sample outputs for km_to_miles:
print(f"10 kilometers is equal to {km_to_miles(10):.2f} miles")
print(f"50 kilometers is equal to {km_to_miles(50):.2f} miles")

# Sample outputs for miles_to_km:
print(f"10 miles is equal to {miles_to_km(10):.2f} kilometers")
print(f"31.07 miles is equal to {miles_to_km(31.07):.2f} kilometers")
```

Output:

```
... 10 kilometers is equal to 6.21 miles
50 kilometers is equal to 31.07 miles
10 miles is equal to 16.09 kilometers
31.07 miles is equal to 50.00 kilometers
```