

## AI ASSISTED CODING

**Name:** P.Swaran Raj

**Batch:** 37

**Roll.No:** 2303A52247

**Assignment-3**

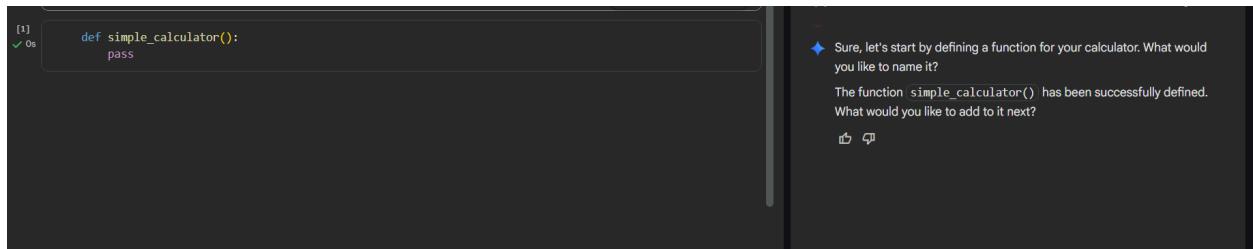
### **Task Description 1:**

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 used:**

**“#design a simple calculator by initially providing only the function name”**



The screenshot shows a dark-themed code editor interface. On the left, a code completion window is open, displaying the following Python code:

```
[1] 0s def simple_calculator():
    pass
```

To the right of the code, the AI's response is shown in a message box:

◆ Sure, let's start by defining a function for your calculator. What would you like to name it?  
The function simple\_calculator() has been successfully defined.  
What would you like to add to it next?

#### **Code Explanation:**

Here, only the function name is mentioned without any logic or instructions. Due to the minimal prompt, the AI generated an empty or incomplete function, showing that vague prompts lead to limited output.

#### **Prompt used:**

**“#Design a calculator function that performs addition, subtraction, multiplication, and division.”**

```

[2] 0s
def simple_calculator():
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        operation = input("Choose an operation (+, -, *, /): ")

        if operation == '+':
            result = num1 + num2
            print(f"Result: {num1} + {num2} = {result}")
        elif operation == '-':
            result = num1 - num2
            print(f"Result: {num1} - {num2} = {result}")
        elif operation == '*':
            result = num1 * num2
            print(f"Result: {num1} * {num2} = {result}")
        elif operation == '/':
            if num2 != 0:
                result = num1 / num2
                print(f"Result: {num1} / {num2} = {result}")
            else:
                print("Error: Division by zero is not allowed.")
        else:
            print("Error: Invalid operation. Please choose from +, -, *, /.")

    except ValueError:
        print("Error: Invalid input. Please enter valid numbers.")

# You can call the function to test it:
# simple_calculator()

```

**Gemini**

P Design a calculator function that performs addition, subtraction, multiplication, and division

Okay, I will modify the `simple_calculator` function to include addition, subtraction, multiplication, and division. It will prompt the user to input two numbers and choose an operation.

The `simple_calculator()` function has been updated and the cell executed. To test it, you can call the function, for example, by uncommenting `simple_calculator()` at the end of the cell or by adding `simple_calculator()` in a new code cell.

What can I help you build?

+ Gemini 2.5 Flash ▶

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

## Code Explanation:

With clearer instructions, the AI generated a calculator function using conditional statements. The logic handled basic arithmetic operations and included error handling for invalid inputs.

## Prompt used:

**“#Create a calculator function that performs addition, subtraction, multiplication, and division with function names and usage examples”**

```

Usage Examples:

# Addition
print("5 + 3 = " + str(add(5, 3)))
print("-10 + 7 = " + str(add(-10, 7)))

# Subtraction
print("10 - 4 = " + str(subtract(10, 4)))
print("5 - 8 = " + str(subtract(5, 8)))

# Multiplication
print("6 * 7 = " + str(multiply(6, 7)))
print("-3 * 4 = " + str(multiply(-3, 4)))

# Division
print("15 / 3 = " + str(divide(15, 3)))
print("10 / 0 = " + str(divide(10, 0))) # Example of division by zero

...
5 + 3 = 8
-10 + 7 = -3
10 - 4 = 6
5 - 8 = -3
6 * 7 = 42
-3 * 4 = -12
15 / 3 = 5.0
10 / 0 = Error: Cannot divide by zero.

```

**Gemini**

P Create a calculator function that performs addition, subtraction, multiplication, and division with function names and usage examples

Certainly! Here are separate functions for addition, subtraction, multiplication, and division, each with a docstring and usage examples.

The calculator functions have been successfully defined and executed with the provided examples. You can see the results for addition, subtraction, multiplication, and division, including the error message for division by zero.

We have successfully designed a simple calculator that performs addition, subtraction, multiplication, and division. We first created a

Test the 'add' function with example inputs

Explain the 'divide' function's error handling

Show how to integrate these functions into a main calculator script

What can I help you build?

+ Gemini 2.5 Flash ▶

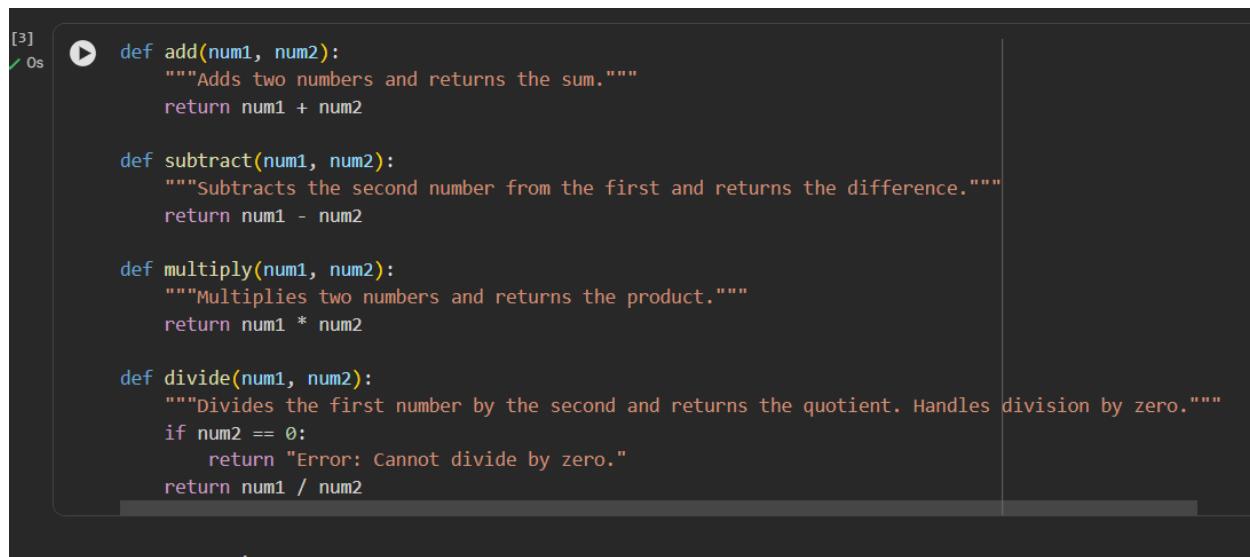
Gemini can make mistakes, so double-check it and use code with caution. [Learn more](#)

## Output:

$10 + 5 = 15$   
 $10 - 5 = 5$   
 $10 \times 5 = 50$   
 $10 \div 5 = 2$

## Code Explanation:

Providing examples helped the AI generate complete and testable code. The calculator logic became more structured and accurate due to refined prompting.



```
[3] 0s  def add(num1, num2):
    """Adds two numbers and returns the sum."""
    return num1 + num2

def subtract(num1, num2):
    """Subtracts the second number from the first and returns the difference."""
    return num1 - num2

def multiply(num1, num2):
    """Multiplies two numbers and returns the product."""
    return num1 * num2

def divide(num1, num2):
    """Divides the first number by the second and returns the quotient. Handles division by zero."""
    if num2 == 0:
        return "Error: Cannot divide by zero."
    return num1 / num2
```

## 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 used:

**“#Sort student marks.”**

The screenshot shows a code editor on the left and an AI interface on the right. The code editor contains Python code for sorting student marks. The AI interface has a 'Gemini' header and a 'Sort student marks' section. It includes a message from Gemini, a 'Show me the sorted student marks' button, and a 'Test the simple\_calculator function' button.

```
[6] 0s # Sort by marks in ascending order
sorted_students_asc = sorted(students, key=lambda x: x['mark'])

print("\nStudent marks sorted in ascending order by mark:")
for student in sorted_students_asc:
    print(f"Name: {student['name']}, Mark: {student['mark']}")

# Sort by marks in descending order
sorted_students_desc = sorted(students, key=lambda x: x['mark'], reverse=True)

print("\nStudent marks sorted in descending order by mark:")
for student in sorted_students_desc:
    print(f"Name: {student['name']}, Mark: {student['mark']}")

...
Student marks sorted in ascending order by mark:
Name: Charlie, Mark: 78
Name: Alice, Mark: 85
Name: Eve, Mark: 88
Name: Bob, Mark: 92
Name: David, Mark: 95

Student marks sorted in descending order by mark:
Name: David, Mark: 95
Name: Bob, Mark: 92
Name: Eve, Mark: 88
Name: Alice, Mark: 85
Name: Charlie, Mark: 78
```

## Code explanation:

This vague prompt resulted in basic sorting without clarity on order. The AI used default sorting behavior, which may not meet specific requirements.

## Prompt used:

**"#Write a Python function to sort student marks in descending order without using built-in sorting functions."**

The screenshot shows a code editor on the left and an AI interface on the right. The code editor contains Python code for a custom sorting function. The AI interface has a 'Gemini' header and a 'Write a Python function to sort student marks in descending order without using built-in sorting functions' section. It includes a message from Gemini, an 'Explain the 'sort\_students\_custom\_desc' function' button, a 'Show how to find the student with the highest mark' button, and an 'Add docstrings to the 'sort\_students\_custom\_desc' function' button.

```
[7] 0s def sort_students_custom_desc(students_list):
    n = len(students_list)
    for i in range(1, n):
        key_student = students_list[i]
        j = i - 1
        # Move elements of students_list[0...i-1], that are smaller than key_student['mark'],
        # to one position ahead of their current position
        while j >= 0 and key_student['mark'] > students_list[j]['mark']:
            students_list[j + 1] = students_list[j]
            j -= 1
        students_list[j + 1] = key_student
    return students_list

# Using the existing 'students' list for demonstration
students_copy = list(students) # Create a copy to avoid modifying the original 'students' list in place
sorted_students_custom = sort_students_custom_desc(students_copy)

print("\nStudent marks sorted in descending order using custom function:")
for student in sorted_students_custom:
    print(f"Name: {student['name']}, Mark: {student['mark']}")

...
Student marks sorted in descending order using custom function:
Name: David, Mark: 95
Name: Bob, Mark: 92
Name: Eve, Mark: 88
Name: Alice, Mark: 85
Name: Charlie, Mark: 78
```

## Code explanation:

With clear constraints, the AI implemented a selection-sort based logic. The refined prompt produced accurate and efficient sorting in descending order.

## 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 used:

**"#Write a Python function to check whether a number is prime with example inputs and outputs."**

The screenshot shows a code editor on the left and a Gemini AI interface on the right. In the code editor, there is a Python script named `is_prime.py` containing the following code:def is\_prime(number):
 """Checks if a given number is prime."""
 if number <= 1:
 return False
 if number <= 3:
 return True
 if number % 2 == 0 or number % 3 == 0:
 return False
 i = 5
 while i \* i <= number:
 if number % i == 0 or number % (i + 2) == 0:
 return False
 i += 6
 return True

Below the code, under the heading "Example Usage of `is_prime` function:", there is a block of test code:print("Is 7 prime? {is\_prime(7)}") # Expected: True
print("Is 10 prime? {is\_prime(10)}") # Expected: False
print("Is 2 prime? {is\_prime(2)}") # Expected: True
print("Is 1 prime? {is\_prime(1)}") # Expected: False
print("Is 0 prime? {is\_prime(0)}") # Expected: False
print("Is -5 prime? {is\_prime(-5)}") # Expected: False
print("Is 13 prime? {is\_prime(13)}") # Expected: True
print("Is 29 prime? {is\_prime(29)}") # Expected: True
print("Is 35 prime? {is\_prime(35)}") # Expected: False
print("Is 97 prime? {is\_prime(97)}") # Expected: True

The Gemini interface on the right shows the prompt: "P Write a Python function to check whether a number is prime with example inputs and outputs". It also shows the AI's response: "Certainly! Here's a Python function to check if a number is prime, complete with example inputs and outputs." The AI has successfully run the provided examples and confirmed their correctness.

## Code Explanation:

The function first checks numbers less than 2 and then tests divisibility up to the square root. Few-shot examples improved correctness and handled edge cases like 0 and 1.

## 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.

### Prompt used:

**"#Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input."**

The screenshot shows a Jupyter Notebook cell with the following code:

```
[14]: display(VBox(subject_marks_widgets + [calculate_button, output_widget]))  
print("Interactive student grading system UI displayed.")
```

Below the code, there is an interactive UI component consisting of five input fields for subject marks (Subject 1 to Subject 5) and a button labeled "Calculate Grade". The output area displays the results:

Total Marks: 410.00 / 500  
Percentage: 82.00%  
Grade: B  
Interactive student grading system UI displayed.

To the right of the notebook, a Gemini AI interface window is open, showing the following text:

Gemini  
total marks, percentage, and grade based on user input

Certainly! I can help you create an interactive user interface for the student grading system using ipywidgets. This will allow you to input marks and see the results directly in the notebook.

- Import Libraries: Import necessary libraries like ipywidgets and IPython.display to create and render the interactive user interface components.
- Define UI Components: Create FloatText widgets for inputting marks for 5 subjects, a Button widget to trigger the calculation, and an Output widget to display the results.
- Implement UI Calculation Logic: Develop a Python function that will be executed when the calculate button is pressed. This function will read marks from the input widgets, perform the total marks, percentage, and grade calculations (reusing the logic from the previously defined calculate\_student\_grade function).

Implementation logic:  
Connect the button click to the calculation function | Show the current UI layout

What can I help you build?

Gemini 2.5 Flash ▶

## Code explanation:

In this stage, the AI generated a basic program that accepts student marks using input statements. The program calculates total marks and percentage, but grading logic is minimal due to limited prompt guidance.

## Prompt used:

**"#Create a Python program that accepts marks of 5 subjects, validates input, calculates total marks, percentage, and displays grade based on percentage."**

## Code Explanation:

With a refined prompt, the AI generated a well-structured program with input validation. The program calculates total marks, percentage, and assigns grades using conditional statements, producing accurate and clear output.

## 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.

## Prompt used:

"#Create a Python program with two functions:

1. Convert kilometers to miles

2. Convert miles to kilometers

Use accurate conversion values and display results clearly."

The screenshot shows a code editor on the left and a Gemini AI interface on the right. The code editor contains two functions: `km_to_miles` and `miles_to_km`. Below the functions, a section titled "Conversion Examples" shows two examples of usage. The first example converts 10 kilometers to miles, and the second converts 5 miles to kilometers. The Gemini interface on the right shows the prompt and the generated response, which includes the code and its output.

```
[16] ✓ 0s
def km_to_miles(kilometers):
    """Converts kilometers to miles using an accurate conversion factor."""
    miles = kilometers * 0.621371 # 1 kilometer = 0.621371 miles
    return miles

def miles_to_km(miles):
    """Converts miles to kilometers using an accurate conversion factor."""
    kilometers = miles * 1.60934 # 1 mile = 1.60934 kilometers
    return kilometers

▼ Conversion Examples:

[16] ✓ 0s
  ➔ # Example 1: Convert Kilometers to Miles
  km_value = 10
  miles_result = km_to_miles(km_value)
  print(f'{km_value} kilometers is equal to {miles_result:.2f} miles.\n')

  km_value_2 = 100
  miles_result_2 = km_to_miles(km_value_2)
  print(f'{km_value_2} kilometers is equal to {miles_result_2:.2f} miles.\n')

  # Example 2: Convert Miles to Kilometers
  miles_value = 5
  km_result = miles_to_km(miles_value)
  print(f'{miles_value} miles is equal to {km_result:.2f} kilometers.\n')

  miles value 2 = 60

Gemini
P Create a Python program with two functions: Convert kilometers to miles Convert miles to kilometers Use accurate conversion values and display results clearly.
◆ Certainly! Here are two Python functions to convert between kilometers and miles, using precise conversion factors, along with clear examples.
The conversion functions have been successfully defined and demonstrated! You can see the results for converting kilometers to miles and miles to kilometers, all calculated using accurate conversion factors.
We have successfully completed several tasks:
Add data persistence features to the student grading system
Explore how to export grades from the student grading system
Show me examples of using the conversion functions
What can I help you build?
Gemini 2.5 Flash ➤
Gemini can make mistakes, so double-check it and use code with caution. Learn more
```

The screenshot shows a code editor on the left and a Gemini AI interface on the right. The code editor contains the same Python program as the previous screenshot. The Gemini interface on the right shows the prompt and the generated response, which includes the code and its output. The response is identical to the one in the previous screenshot.

```
[16] ✓ 0s
  ➔ miles_result = km_to_miles(km_value)
  print(f'{km_value} kilometers is equal to {miles_result:.2f} miles.\n')

  km_value_2 = 100
  miles_result_2 = km_to_miles(km_value_2)
  print(f'{km_value_2} kilometers is equal to {miles_result_2:.2f} miles.\n')

  # Example 2: Convert Miles to Kilometers
  miles_value = 5
  km_result = miles_to_km(miles_value)
  print(f'{miles_value} miles is equal to {km_result:.2f} kilometers.\n')

  miles_value_2 = 60
  km_result_2 = miles_to_km(miles_value_2)
  print(f'{miles_value_2} miles is equal to {km_result_2:.2f} kilometers.\n')

  ...
  10 kilometers is equal to 6.21 miles.
  100 kilometers is equal to 62.14 miles.

  5 miles is equal to 8.05 kilometers.
  60 miles is equal to 96.56 kilometers.

Gemini
P Create a Python program with two functions: Convert kilometers to miles Convert miles to kilometers Use accurate conversion values and display results clearly.
◆ Certainly! Here are two Python functions to convert between kilometers and miles, using precise conversion factors, along with clear examples.
The conversion functions have been successfully defined and demonstrated! You can see the results for converting kilometers to miles and miles to kilometers, all calculated using accurate conversion factors.
We have successfully completed several tasks:
Add data persistence features to the student grading system
Explore how to export grades from the student grading system
Show me examples of using the conversion functions
```

## **Code Explanation:**

The AI generated two separate functions using correct conversion factors. Clear prompts resulted in accurate conversions and well-formatted output.

## **Conclusion:**

This lab demonstrates that clear, specific, and example-based prompts significantly improve the accuracy, structure, and reliability of AI-generated code.