

AI ASSISTED CODING

Name: P.Swaran Raj
Batch: 37

Roll.No: 2303A52247
Assignment-3

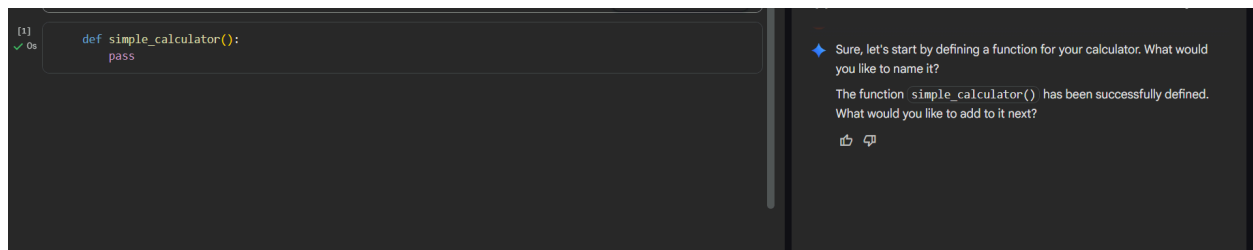
Task Description 1:

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”

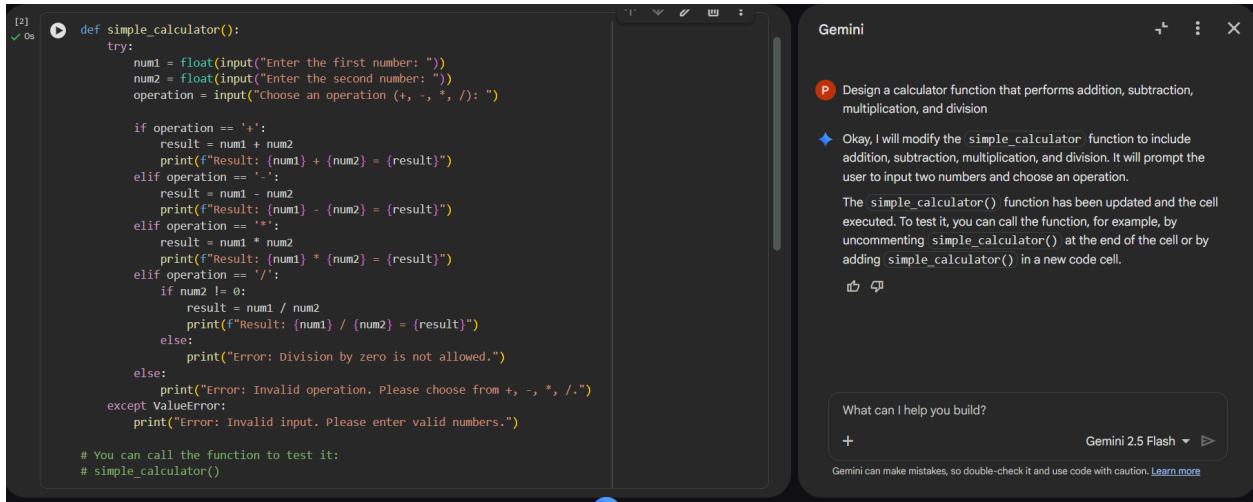


Code Explanation:

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.”

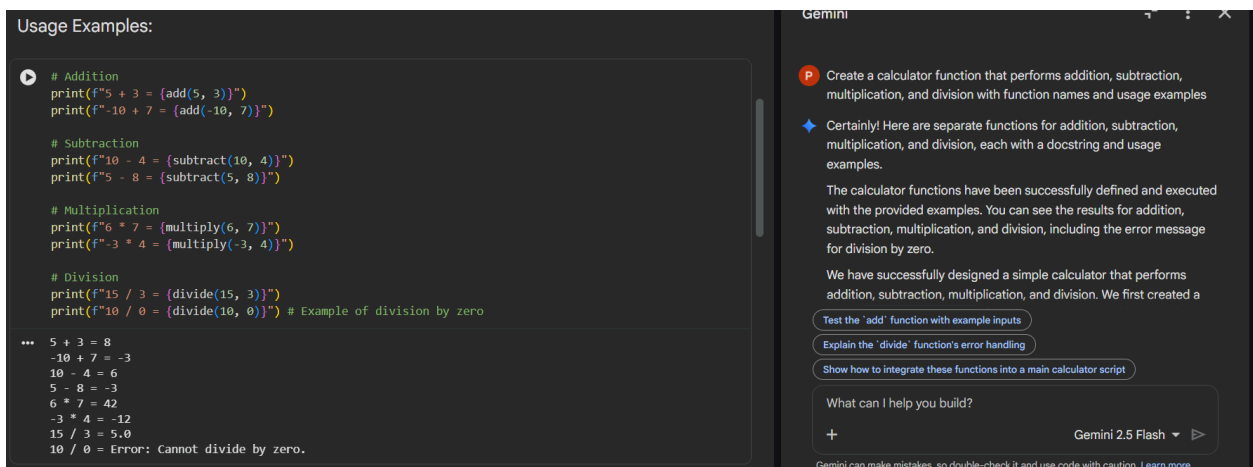


Code Explanation:

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”



Output:

$$10 + 5 = 15$$

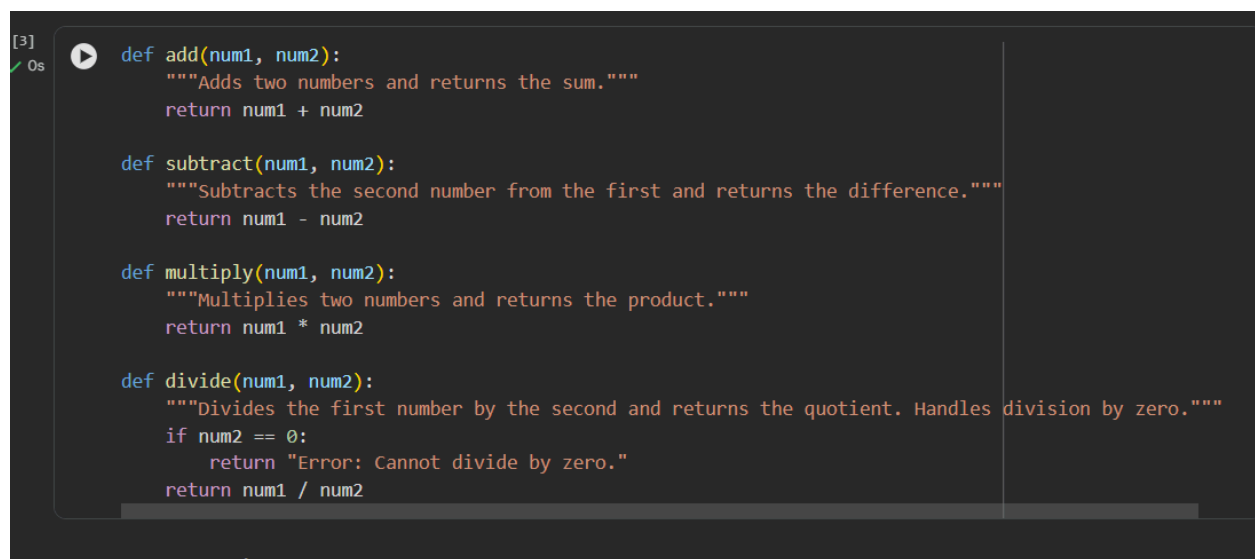
$$10 - 5 = 5$$

$$10 \times 5 = 50$$

$$10 \div 5 = 2$$

Code Explanation:

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

A screenshot of a code editor with a dark background. On the left, there is a sidebar with a play button icon and the text "[3]" and "0s". The main area contains four Python functions:

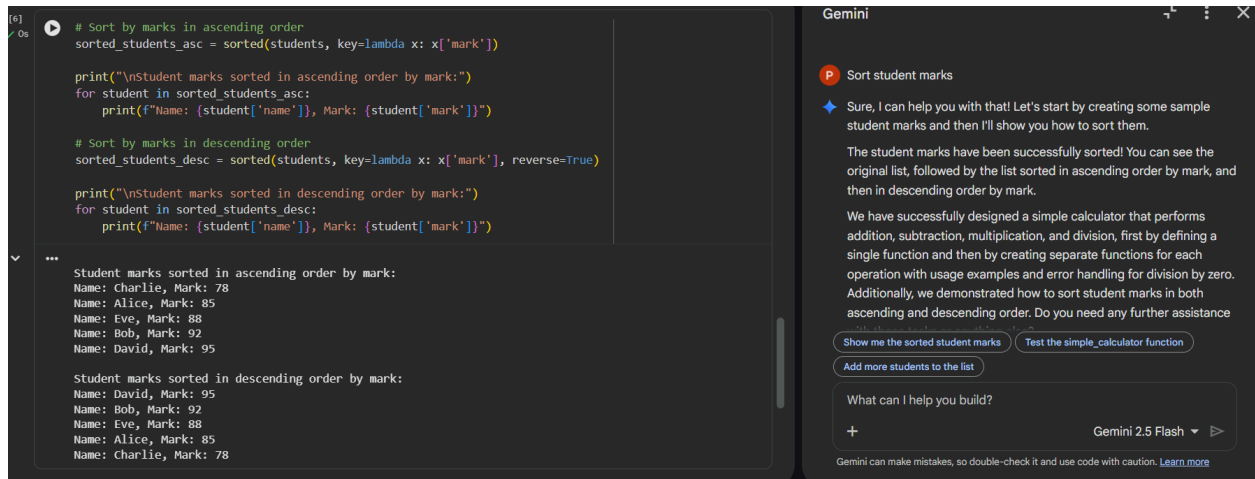
```
def add(num1, num2):  
    """Adds two numbers and returns the sum."""  
    return num1 + num2  
  
def subtract(num1, num2):  
    """Subtracts the second number from the first and returns the difference."""  
    return num1 - num2  
  
def multiply(num1, num2):  
    """Multiplies two numbers 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.”

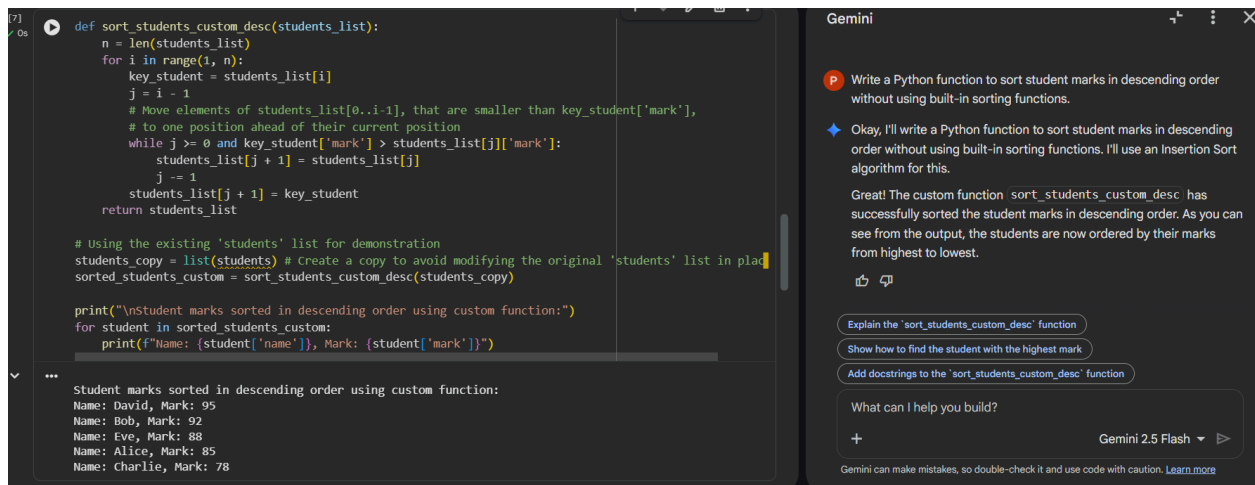


Code explanation:

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.”



Code explanation:

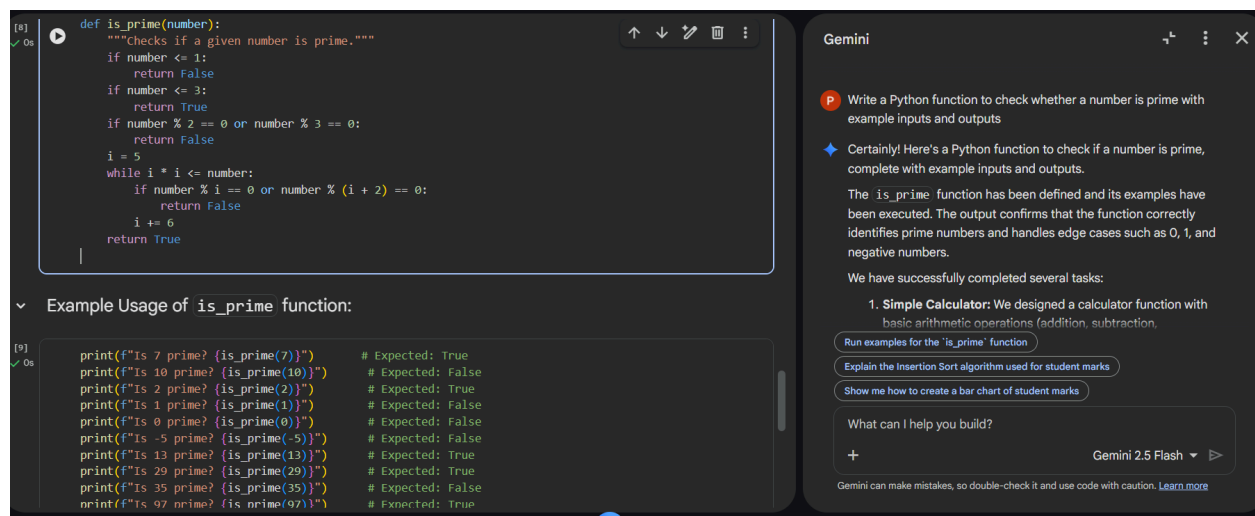
With clear constraints, the AI implemented a selection-sort based logic. The refined prompt produced accurate and efficient sorting in descending order.

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 displays a code editor on the left and a Gemini chat interface on the right. The code editor shows a Python function `is_prime` that checks if a number is prime. It includes comments and handles edge cases like 0, 1, and negative numbers. Below the function, there is an 'Example Usage of is_prime function:' section with a list of test cases and their expected results. The Gemini chat interface on the right shows a prompt asking for a Python function to check if a number is prime with example inputs and outputs. The chat response provides the function and explains its logic, including handling edge cases. It also lists several tasks completed by the function, such as a simple calculator and explaining the Insertion Sort algorithm.

```
[8] 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
```

Example Usage of `is_prime` function:

```
[9] print(f"Is 7 prime? {is_prime(7)}")    # Expected: True  
    print(f"Is 10 prime? {is_prime(10)}") # Expected: False  
    print(f"Is 2 prime? {is_prime(2)}")   # Expected: True  
    print(f"Is 1 prime? {is_prime(1)}")   # Expected: False  
    print(f"Is 0 prime? {is_prime(0)}")   # Expected: False  
    print(f"Is -5 prime? {is_prime(-5)}") # Expected: False  
    print(f"Is 13 prime? {is_prime(13)}") # Expected: True  
    print(f"Is 29 prime? {is_prime(29)}") # Expected: True  
    print(f"Is 35 prime? {is_prime(35)}") # Expected: False  
    print(f"Is 97 prime? {is_prime(97)}") # Expected: True
```

Gemini

P Write a Python function to check whether a number is prime with example inputs and outputs

◆ Certainly! Here's a Python function to check if a number is prime, complete with example inputs and outputs.

The `is_prime` function has been defined and its examples have been executed. The output confirms that the function correctly identifies prime numbers and handles edge cases such as 0, 1, and negative numbers.

We have successfully completed several tasks:

1. **Simple Calculator:** We designed a calculator function with basic arithmetic operations (addition, subtraction).

Run examples for the 'is_prime' function

Explain the Insertion Sort algorithm used for student marks

Show me how to create a bar chart of student marks

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:

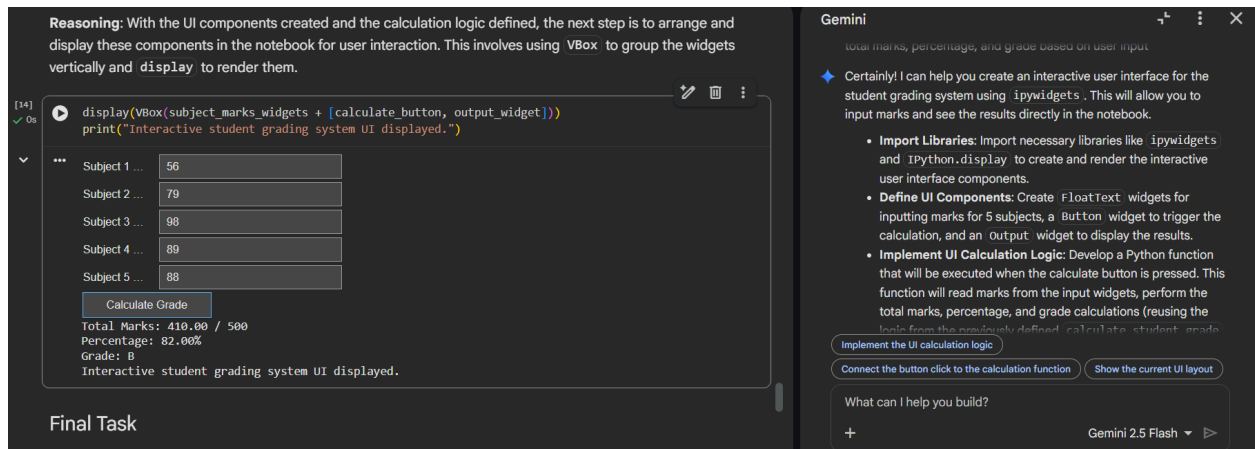
The function first checks numbers less than 2 and then tests divisibility up to the square root. Few-shot examples improved correctness and handled edge cases like 0 and 1.

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.”



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 the Gemini chat interface on the right. The code editor contains the following Python code:

```
[15] ✓ 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.')

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.')

miles_value_2 = 60
```

The Gemini chat interface on the right shows the prompt and the first part of the response:

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

The screenshot shows the same code editor and Gemini chat interface. The code editor now includes the output of the program:

```
[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.')

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.')

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.')

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.
```

The Gemini chat interface on the right shows the second part of the response:

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.