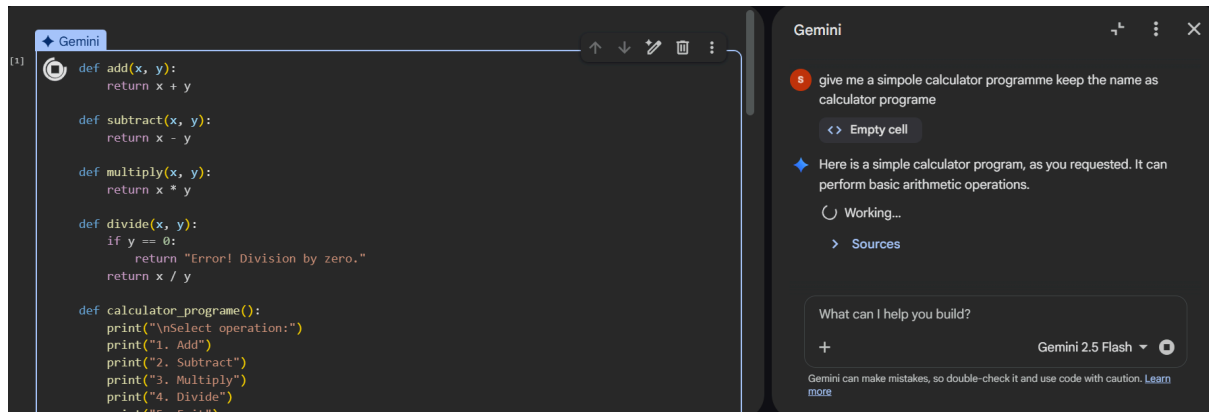


2303a51214

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.



```
if choice in ('1', '2', '3', '4'):
    try:
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))
    except ValueError:
        print("Invalid input. Please enter numbers only.")
        continue

    if choice == '1':
        print(num1, "+", num2, "=", add(num1, num2))

    elif choice == '2':
        print(num1, "-", num2, "=", subtract(num1, num2))

    elif choice == '3':
        print(num1, "*", num2, "=", multiply(num1, num2))

    elif choice == '4':
        print(num1, "/", num2, "=", divide(num1, num2))
elif choice == '5':
    print("Exiting calculator. Goodbye!")
    break
else:
    print("Invalid input. Please enter 1, 2, 3, 4, or 5.")

# To run the calculator, call the function:
calculator_programe()

...
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4/5): 
```

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.

Expected Output-2

• AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.

Sort student marks

To sort student marks, I'll first create a sample dataset using a pandas DataFrame. Then, I'll show you how to sort these marks.

```
Create a sample DataFrame for student marks
data = {
    'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Math': [85, 92, 78, 90, 88],
    'Science': [79, 88, 85, 91, 82],
    'English': [90, 85, 92, 75, 80]
}
df_marks = pd.DataFrame(data)

print("Original Student Marks:")
display(df_marks)
Sort by 'Math' score in descending order
df_sorted_math = df_marks.sort_values(by='Math', ascending=False)

print("Student Marks Sorted by Math Score (Descending):")
display(df_sorted_math)
Sort by 'Science' (descending) and then 'English' (ascending)
df_sorted_multi = df_marks.sort_values(by=['Science', 'English'],
ascending=[False, True])

print("Student Marks Sorted by Science (Desc) and then English (Asc):")
display(df_sorted_multi)
```

Task Description-3

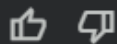
- Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness.

Expected Output-3

- Improved prime-checking function with better edge-case handling.

s Check if a number is prime. Examples: Input: 2 → Output: Prime
Input: 4 → Output: Not Prime Input: 5 → Output: Prime Input: 1
→ Output: Not Prime Write a Python function.

✦ Here's a Python function `is_prime` that checks if a number is prime, along with example usage.



```
import math

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

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

    Returns:
        str: 'Prime' if the number is prime, 'Not Prime' otherwise.
    """
    if num <= 1:
        return "Not Prime"
    if num == 2:
        return "Prime"
    if num % 2 == 0:
        return "Not Prime"
    for i in range(3, int(math.sqrt(num)) + 1, 2):
        if num % i == 0:
            return "Not Prime"
    return "Prime"

# Test cases
print(f"Input: 2 → Output: {is_prime(2)}")
print(f"Input: 4 → Output: {is_prime(4)}")
print(f"Input: 5 → Output: {is_prime(5)}")
print(f"Input: 1 → Output: {is_prime(1)}")
print(f"Input: 7 → Output: {is_prime(7)}")
print(f"Input: 9 → Output: {is_prime(9)}")
```

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.

Expected Output-4

- Well-structured UI code with accurate calculations and clear output display.

S Create a Python program with a user interface that: - Takes student name - Takes marks of 5 subjects - Calculates total, percentage - Displays grade

 Working...

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](#)

```
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()

```

```

def km_to_miles(kilometers):
    "

```

Task Description-5

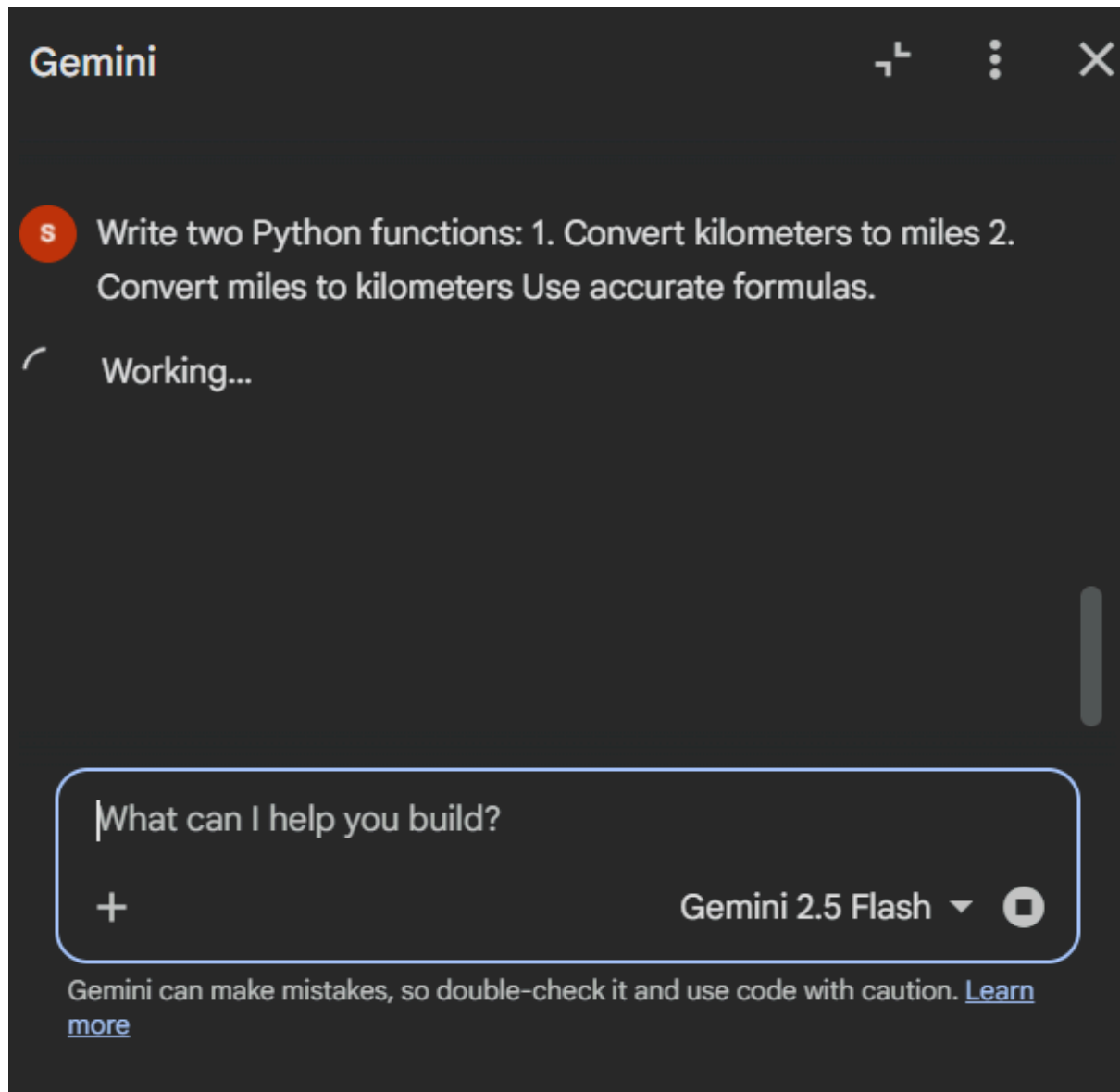
- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit

Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear

Instructions.

Expected Output-5

- Analysis of code quality and accuracy differences across multiple prompt variations.



Certainly! Here are two Python functions to convert between kilometers and miles, along with examples to demonstrate their use.

The conversion functions have run successfully! Here are the results:

- 10 kilometers is equal to 6.21 miles.
- 5 miles is equal to 8.05 kilometers.
- 1 kilometer is equal to 0.62 miles.

- 1 mile is equal to 1.61 kilometers.

```
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.")
```