

## AI ASSISTED CODING-ASSIGNMENT -3

NAME : R.RONITH REDDY  
HTNO: 2303A52280

**TASK- 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 Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code editor contains the following Python code:

```
[4] parameters.  
[4] def add(x, y):  
[4]     # Implementation for addition will go here  
[4]     pass  
  
[4]     def subtract(x, y):  
[4]         # Implementation for subtraction will go here  
[4]         pass  
  
[4]     def multiply(x, y):  
[4]         # Implementation for multiplication will go here  
[4]         pass  
  
[4]     def divide(x, y):  
[4]         # Implementation for division will go here  
[4]         pass
```

A sidebar on the right is titled "Gemini" and shows the following conversation:

- R Prompt used: "#design a simple calculator by initially providing only the function name"
- ◆ My apologies for the previous attempt! It seems there was an issue and the code was not added. Let's try again with your refined request. I will start by outlining the function names first, and then I'll provide the implementation for each.
- ◇ Working...

The code editor then shows the completed logic:

```
[5] def add(x, y):  
[5]     return x + y  
  
[5] def subtract(x, y):  
[5]     return x - y  
  
[5] def multiply(x, y):  
[5]     return x * y  
  
[5] def divide(x, y):  
[5]     if y == 0:  
[5]         raise ValueError("Division by zero")  
[5]     else:  
[5]         return x / y
```

The "Variables" and "Terminal" tabs are visible at the bottom left, and the status bar indicates "Executing (40s)" and "Python 3".

**EXPLANATION :**  
**THE FUNCTION NAME IS USED WHERE IT DEFINED ALL THE ARITHMETICA FUNCTION ALONG WITH PROCESS , WHERE THE ARITHMETIC OPERATIONS CAN BE DONE , WHERE IT IS COMPLETELY GENERATED THROUGH GEMINI AI .**

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```

[5] 0s
def calculate(operation, num1, num2):
    """
    Performs an arithmetic operation based on the given string.

    Args:
        operation (str): The name of the operation ('add', 'subtract', 'multiply', 'divide').
        num1 (float): The first number.
        num2 (float): The second number.

    Returns:
        float or str: The result of the operation or an error message.
    """
    if operation == 'add':
        return add(num1, num2)
    elif operation == 'subtract':
        return subtract(num1, num2)
    elif operation == 'multiply':
        return multiply(num1, num2)
    elif operation == 'divide':
        return divide(num1, num2)
    else:
        return "Error: Invalid operation specified."
    
```

Finally, run the `calculator()` function to start the interactive calculator.

**Gemini**

R Prompt used: "#design a simple calculator by initially providing only the function name"

◆ My apologies for the previous attempt! It seems there was an issue and the code was not added. Let's try again with your refined request. I will start by outlining the function names first, and then I'll provide the implementation for each.

Working...

**PROMPT USED :**  
**CREATE A CALCULATION FUNCTION THAT PERFORMS ARITHMETIC OPERATION WITH FUNCTION NAMES AND USAGE EXAMPLES**

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```

[8] 0s
def calculation(operation, num1, num2):
    """
    Performs an arithmetic operation based on the given string.

    Args:
        operation (str): The name of the operation ('add', 'subtract', 'multiply', 'divide').
        num1 (float): The first number.
        num2 (float): The second number.

    Returns:
        float or str: The result of the operation or an error message.
    """
    if operation == 'add':
        return add(num1, num2)
    elif operation == 'subtract':
        return subtract(num1, num2)
    elif operation == 'multiply':
        return multiply(num1, num2)
    elif operation == 'divide':
        return divide(num1, num2)
    else:
        return "Error: Invalid operation specified."
    
```

Here are some examples of how to use the `calculation` function:

**Gemini**

R create a calculation function that performs arithmetic operation with function names and usage examples

◆ Certainly! I'll create a `calculation` function that takes an operation name and two numbers, then performs the requested arithmetic. It will also show you how to use it with various examples.

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](#)

Variables Terminal

10:35PM Python 3

```

print(f"Result: {calculation('add', 20, 10)}\n")
print(f"Result: {calculation('subtract', 20, 10)}\n")
print(f"Result: {calculation('multiply', 5, 4)}\n")
print(f"Result: {calculation('divide', 100, 20)}\n")
print(f"Result: {calculation('divide', 10, 0)}\n")
print(f"Result: {calculation('power', 2, 3)}\n") # Example of an invalid operation

... Performing add operation with 20 and 10:
Result: 30

Performing subtract operation with 20 and 10:
Result: 10

Performing multiply operation with 5 and 4:
Result: 20

Performing divide operation with 100 and 20:
Result: 5.0

Performing divide operation with 10 and 0:
Result: Error: Cannot divide by zero!

Performing power operation with 2 and 3:
Result: Error: Invalid operation specified.

```

**Gemini**

- R create a calculation function that performs arithmetic operation with function names and usage examples
- Blue diamond Certainly! I'll create a `calculation` function that takes an operation name and two numbers, then performs the requested arithmetic. It will also show you how to use it with various examples.
- Working...

What can I help you build? Gemini

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

## CODE EXPLANATION :

**THE FUNCTION DEFINES THE ARITHMETIC OPERATION WHERE ADDITION , SUBTRACTION, MULTIPLICATION GIVE THE EXPECTED OUTPUT , WHERE AS IN DIVISION OPERATION CORRECTLY HANDLED BY ZERO AND RETURN TO ERROR , INVALID OPERATION IS ALSO BEING SPECIFIED , AND THE FUNCTION RETURNED TO ERROR.**

## TASK 2 : PROMPTS FOR SORTING THE MARKS USING SORTING FUNCTION

**PROMPT :**  
**GENERATE PYTHON CODE FOR THE SORTING THE STUDENT MARKS**

```

# Sample student data
students = [
    {'name': 'Alice', 'mark': 85},
    {'name': 'Bob', 'mark': 92},
    {'name': 'Charlie', 'mark': 78},
    {'name': 'David', 'mark': 92},
    {'name': 'Eve', 'mark': 85}
]

print("Original student list:")
for student in students:
    print(student)

```

1. Sorting by Mark (Ascending Order)

We can use the `.sorted()` function with a `lambda` key to sort the list of dictionaries by the 'mark' value in ascending order

```

# Sort by 'mark' in ascending order

```

**Gemini**

- I generate python code for the sorting the student marks
- Blue diamond Okay, I'll generate Python code to sort student marks. I'll assume your student data is in a list of dictionaries, where each dictionary contains a student's name and their mark. This approach is flexible and widely applicable. I will show you how to sort by mark (ascending and descending) and by name.
- 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](#)

The screenshot shows a Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell [11] contains Python code to sort a list of student dictionaries by mark in ascending order:

```
# Sort by 'mark' in ascending order
sorted_by_mark_asc = sorted(students, key=lambda student: student['mark'])

print("\nSorted by mark (ascending):")
for student in sorted_by_mark_asc:
    print(student)
```

The output shows the students sorted by mark in ascending order:

```
Sorted by mark (ascending):
{'name': 'Charlie', 'mark': 78}
{'name': 'Alice', 'mark': 85}
{'name': 'Eve', 'mark': 85}
{'name': 'Bob', 'mark': 92}
{'name': 'David', 'mark': 92}
```

Cell [12] contains code to sort the same list by mark in descending order:

```
# Sort by 'mark' in descending order
sorted_by_mark_desc = sorted(students, key=lambda student: student['mark'], reverse=True)

print("\nSorted by mark (descending):")
for student in sorted_by_mark_desc:
    print(student)
```

The output shows the students sorted by mark in descending order:

```
Sorted by mark (descending):
{'name': 'Bob', 'mark': 92}
{'name': 'David', 'mark': 92}
{'name': 'Alice', 'mark': 85}
{'name': 'Eve', 'mark': 85}
{'name': 'Charlie', 'mark': 78}
```

The Gemini sidebar on the right is active, showing a prompt to generate Python code for sorting student marks. It provides a detailed response explaining the approach and includes links to calculate average marks, visualize distributions, and add marks using a calculator.

## EXPLANATION :

**THE PROMPT GENERATED THE CODE IN TERMS OF ALL ASPECTS OF ASCENDING AND DESCENDING ORDER BY USING SORTING FUNCTION.**

## TASK - 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 :

**CREATE A USER INTERFACE FOR STUDENT GRADING SYSTEM THAT CALCULATES , TOTAL BASED ON USER INPUT**

The screenshot shows a Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell [23] contains Python code for a function named `display_student_report`:

```
def display_student_report(student_data, total_marks, percentage, grade):
    """
    Displays a formatted report for a student.

    Args:
        student_data (dict): A dictionary containing the student's name and their marks.
            Example: {'name': 'Alice', 'marks': {'Math': 85, 'Science': 90}}
        total_marks (int): The student's total marks.
        percentage (float): The student's overall percentage.
        grade (str): The student's assigned letter grade.

    Prints:
        ("n" + "-" * 30)
        f"Student Report: {student_data['name']}"
        ("-" * 30)

        ("Subject Marks:")
        for subject, mark in student_data['marks'].items():
            print(f" {subject}: {mark}")

        print("nTotal Marks: {total_marks}")
        print(f"Percentage: {percentage:.2f}%")
        print(f"Grade: {grade}")
        print("-" * 30)

    Prints:
        "Function 'display_student_report' defined successfully."
    """

    print("Function 'display_student_report' defined successfully.")

    ... Function 'display_student_report' defined successfully.
```

The Gemini sidebar on the right is active, providing a step-by-step guide to creating a student grading system. It includes tasks like preparing, defining grading criteria, creating an input interface, calculating total marks, assigning grades, displaying reports, and finally summarizing the system.

## EXPLANATION :

THE GIVEN GEMINI GIVEN IS DEFINED COMPLETELY ABOUT THE STUDENT MARKS , PERCENTAGE AND GRADE ACCORDING TO THE MARKS ACHIEVED BY THE STUDENTS , IT CALCULATES THE TOTAL MARKS OF STUDENTS ALONG WITH PERCENTAGE AND GRADE .

## PROMPT :

CREATE A PYTHON PROGRAM THAT ACCEPTS MARKS OF 5 SUBJECTS, VALIDATES INPUT, CALCULATES TOTAL MARKS, PERCENTAGE, AND DISPLAY: GRADE BASED ON PERCENTAGE.

The screenshot shows a Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell contains:

```
[28] display_student_report(student_info, total_marks, percentage, grade)
print("Student report displayed successfully.")
```

The output cell shows the generated report:

```
Student Report: Bren
Subject Marks:
  Math: 67
  Science: 90
  History: 99
  English: 89
  Art: 32

Total Marks: 377
Percentage: 75.40%
Grade: C

Student report displayed successfully.
```

The sidebar on the right has a "Gemini" section with a "Final Task" subtask:

- Final Task: Summarize the complete student grading program, explaining its functionality, the interactive steps, and how it meets all the specified requirements.

Below the sidebar, there are several buttons for Gemini tasks:

- Assign Letter Grade
- Display Final Report
- Review Grading Criteria
- What can I help you build?

At the bottom, the status bar shows "11:55 PM" and "Python 3".

The screenshot shows a Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell contains:

```
[26] max_marks_per_subject = 100
total_marks, percentage = calculate_total_and_percentage(student_info['marks'], max_marks_per_subject)

print(f"Total marks for {student_info['name']}: {total_marks}")
print(f"Percentage for {student_info['name']}: {percentage:.2f}%")
print("Total marks and percentage calculated successfully.")
```

The output cell shows the results:

```
... Total marks for Bren : 377
Percentage for Bren : 75.40%
Total marks and percentage calculated successfully.
```

The sidebar on the right has a "Gemini" section with a "Final Task" subtask:

- Final Task: Summarize the complete student grading program, explaining its functionality, the interactive steps, and how it meets all the specified requirements.

Below the sidebar, there are several buttons for Gemini tasks:

- Assign Letter Grade
- Display Final Report
- Review Grading Criteria
- What can I help you build?

At the bottom, the status bar shows "11:55 PM" and "Python 3".

## **EXPLANATION :**

**THE PROMPTS IS VERY MUCH REDEFINED AND DEFINED EVERY FUNCTION THROUGH GEMINI AI , WHERE IT VALIDATES THE SUBJECTS , MARKS OBTAINED , PERCENTAGE CALCULATION AND ASSIGN GRADES , WHERE IT GIVES THE PROPER OUTPUT BY FOLLOWING THE CONDITIONS ACCURATELY.**

## **TASK - 5**

**ANALYZING PROMPT SPECIFICITY IN UNIT CONVERSION FUNCTIONS: IMPROVING A UNIT CONVERSION FUNCTION (KILOMETERS TO MILES AND MILES TO KILOMETERS) USING CLEAR INSTRUCTIONS.**

### **PROMPT :**

**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 Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell contains two functions: `kilometers_to_miles` and `miles_to_kilometers`. The `kilometers_to_miles` function converts kilometers to miles using a conversion factor of 0.621371. The `miles_to_kilometers` function converts miles to kilometers using a conversion factor of 1.60934. Both functions print a success message. Below the code, a note says "Next, let's define the `miles_to_kilometers` function. We know that 1 mile is approximately 1.60934 kilometers." The Gemini AI interface on the right shows a prompt asking for a Python program with two functions: one to convert kilometers to miles and another to convert miles to kilometers, using precise conversion values. Gemini responds by creating the two functions. The AI interface also includes a sidebar with options like "Ask to review the student grading system code", "Test the kilometers to miles converter", and "Start a new task".

```
[29] ✓ Os
def kilometers_to_miles(kilometers):
    """
    Converts a distance from kilometers to miles.

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

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

print("Function `kilometers_to_miles` defined successfully.")

Function `kilometers_to_miles` defined successfully.

Next, let's define the `miles_to_kilometers` function. We know that 1 mile is approximately 1.60934 kilometers.

[30] ✓ Os
def miles_to_kilometers(miles):
    """
    Converts a distance from miles to kilometers.

    Args:
        miles (float): The distance in miles.

    Returns:
        float: The distance in kilometers.
    """
    conversion_factor = 1.60934 # 1 mile = 1.60934 kilometers
    kilometers = miles * conversion_factor
    return kilometers
```

Gemini

B "#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."

◆ Certainly! I'll create two Python functions for you: one to convert kilometers to miles and another to convert miles to kilometers, using precise conversion values. I will then show examples of how to use them.

The conversion functions have been successfully defined and tested!

- `kilometers_to_miles` correctly converted 10 kilometers

Ask to review the student grading system code  
Test the kilometers to miles converter  
Start a new task

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](#)

12:06 AM Python 3

The screenshot shows a Google Colab notebook titled "AI&AC\_ASSINMT\_3 - Colab". The code cell contains the following Python function:

```

Args:
    miles (float): The distance in miles.

Returns:
    float: The distance in kilometers.
    """
conversion_factor = 1.60934 # 1 mile = 1.60934 Kilometers
kilometers = miles * conversion_factor
return kilometers

print("Function `miles_to_kilometers` defined successfully.")

```

The output of the cell shows the function definition:

Function `miles\_to\_kilometers` defined successfully.

Below the code cell, a note says "Now, let's see these functions in action with some examples!"

The code cell below contains examples of using the function:

```

# Example 1: Convert 10 kilometers to miles
k_val_1 = 10
m_val_1 = kilometers_to_miles(k_val_1)
print(f"{k_val_1} kilometers is equal to {m_val_1:.2f} miles")

# Example 2: Convert 50 miles to kilometers
m_val_2 = 50
k_val_2 = miles_to_kilometers(m_val_2)
print(f"{m_val_2} miles is equal to {k_val_2:.2f} kilometers")

# Example 3: Convert a decimal value from kilometers to miles
k_val_3 = 24.5
m_val_3 = kilometers_to_miles(k_val_3)
print(f"{k_val_3} kilometers is equal to {m_val_3:.3f} miles")

```

The output of the code cell shows the results of the examples:

10 kilometers is equal to 6.21 miles  
50 miles is equal to 80.47 kilometers  
24.5 kilometers is equal to 15.224 miles

At the bottom of the screen, there is a sidebar for "Gemini" which includes a message from Gemini, a "Ask to review the student grading system code" button, a "Test the kilometers to miles converter" button, and a "Start a new task" button.

## EXPLANATION :

**GEMINI AI GENERATED FUNCTIONS WHERE CONVERSIONS CAN BE TURNED INTO KILOMETRES TO MILES AND MILES TO KILOMETERS , THIS PROMPT RESULTED APPROXIMATE CONVERSION AND STRUCTURED OUTPUT**

## CONCLUSION :

**FROM THE ABOVE WE CAN LEARN THAT AI GENERATED PROMPTS CAN GIVE IMPROVED CODE AND ACCURACY OF THE OUTPUT .**