

AI ASSISTED CODING LAB

ASSIGNMENT 2.4

ENROLLMENT NO :2503A51L15

BATCH NO: 19

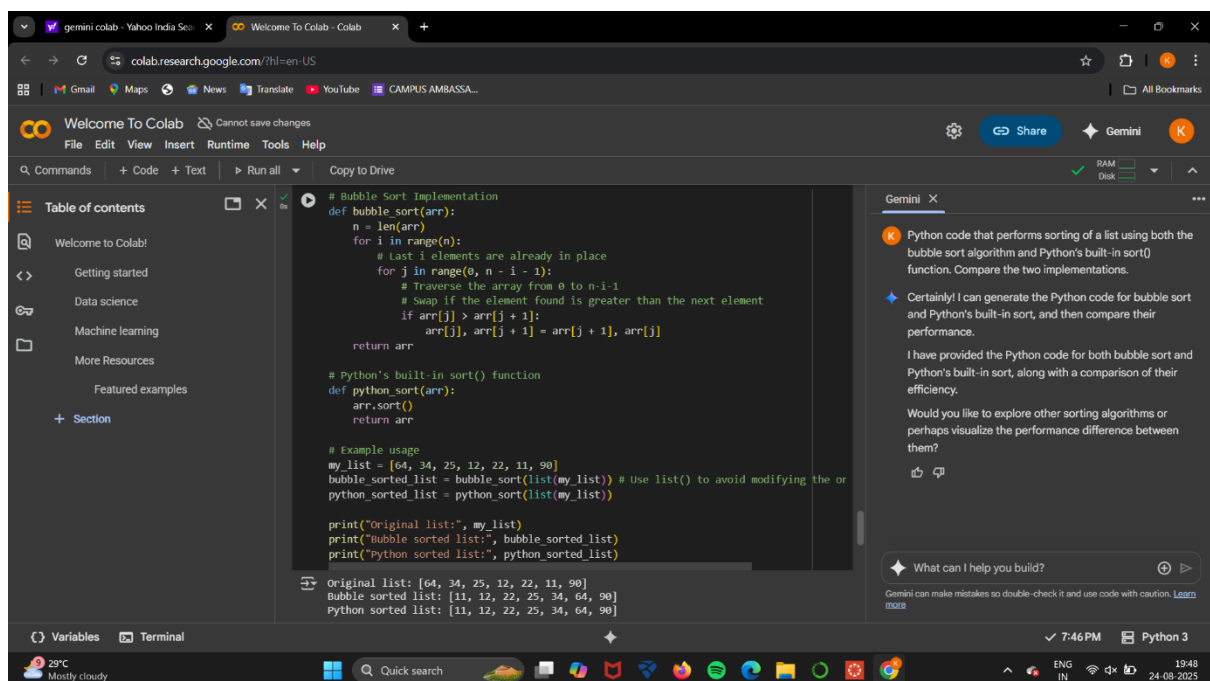
NAME: MOHAMMAD KHAJA AFZALUDDIN

TASK1

TASK1 DESCRIPTION:- Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations.

PROMPT:-Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations.

CODE:-



The screenshot shows the Google Colab web interface. The main code cell contains the following Python code:

```
# Bubble Sort Implementation
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n - i - 1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Python's built-in sort() function
def python_sort(arr):
    arr.sort()
    return arr

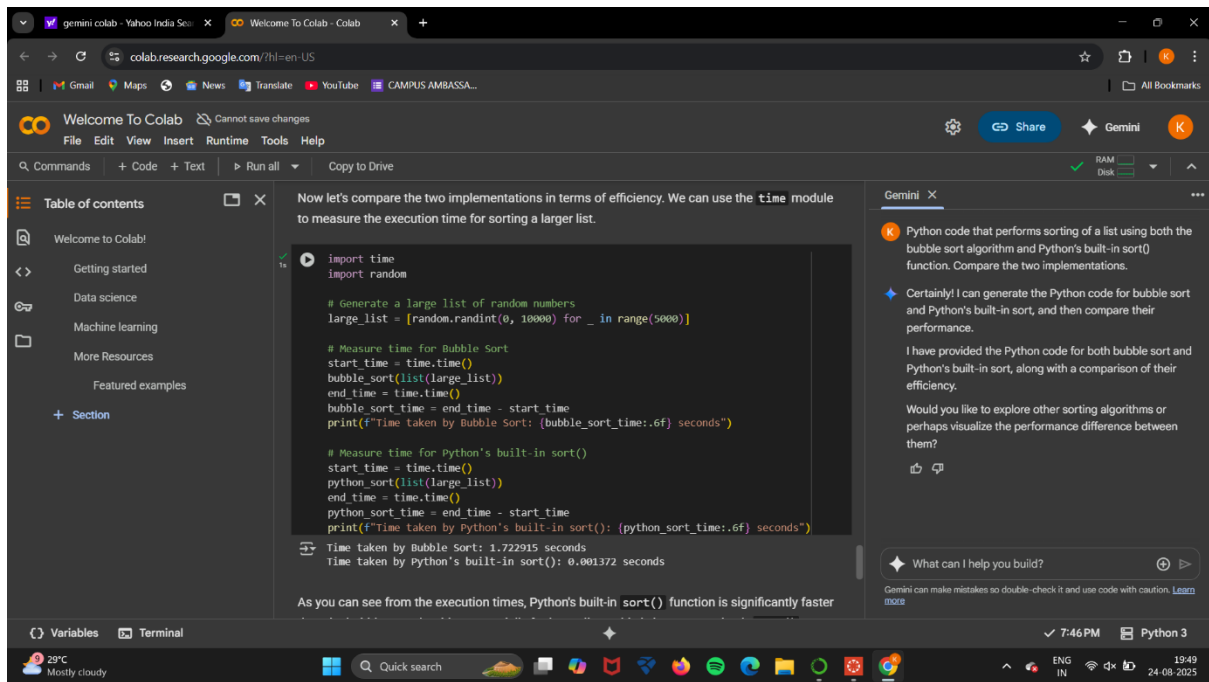
# Example usage
my_list = [64, 34, 25, 12, 22, 11, 90]
bubble_sorted_list = bubble_sort(list(my_list)) # Use list() to avoid modifying the or
python_sorted_list = python_sort(list(my_list))

print("Original list:", my_list)
print("Bubble sorted list:", bubble_sorted_list)
print("Python sorted list:", python_sorted_list)
```

The output of the code is displayed at the bottom:

```
Original list: [64, 34, 25, 12, 22, 11, 90]
Bubble sorted list: [11, 12, 22, 25, 34, 64, 90]
Python sorted list: [11, 12, 22, 25, 34, 64, 90]
```

On the right side, the Gemini AI chat interface is open. It shows a prompt: "Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations." and a response from Gemini: "Certainly! I can generate the Python code for bubble sort and Python's built-in sort, and then compare their performance. I have provided the Python code for both bubble sort and Python's built-in sort, along with a comparison of their efficiency. Would you like to explore other sorting algorithms or perhaps visualize the performance difference between them?"



TASK2

TASK2 DESCRIPTION:- In Colab, use Google Gemini to generate a Python function that takes a string and returns The number of vowels, The number of consonants, The number of digits in the string

PROMPT:-Generate a Python function that takes a string and returns: The number of vowels, The number of consonants, The number of digits in the string

CODE:-

The screenshot shows the Google Colab interface. On the left is a 'Table of contents' sidebar. The main code editor contains a Python function named `analyze_string` that takes an `input_string` and returns a tuple of counts for vowels, consonants, and digits. The function uses three sets: `vowels = "aeiouAEIOU"`, `consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"`, and `digits = "0123456789"`. It iterates through each character in the input string and increments the respective counter. The right sidebar shows the Gemini chat interface with a prompt: 'generate a Python function that takes a string and returns: The number of vowels, The number of consonants, The number of digits in the string'. The system response is visible, showing the generated function and a confirmation message.

```
def analyze_string(input_string):  
    """  
    Analyzes a string to count vowels, consonants, and digits.  
    Args:  
        input_string: The string to analyze.  
    Returns:  
        A tuple containing the counts of vowels, consonants, and digits.  
    """  
    vowels = "aeiouAEIOU"  
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"  
    digits = "0123456789"  
  
    vowel_count = 0  
    consonant_count = 0  
    digit_count = 0  
  
    for char in input_string:  
        if char in vowels:  
            vowel_count += 1  
        elif char in consonants:  
            consonant_count += 1  
        elif char in digits:  
            digit_count += 1  
  
    return vowel_count, consonant_count, digit_count
```

This screenshot shows the same Google Colab interface after the code has been executed. The code editor now includes an example usage section at the bottom: `# Example usage`, `test_string = "Hello World 123!"`, `v, c, d = analyze_string(test_string)`, and `print` statements for each count. Below the code, the output of the execution is displayed: 'Original string: Hello World 123!', 'Number of vowels: 3', 'Number of consonants: 7', and 'Number of digits: 3'. The Gemini chat interface on the right remains the same as in the first screenshot.

```
digits = "0123456789"  
  
vowel_count = 0  
consonant_count = 0  
digit_count = 0  
  
for char in input_string:  
    if char in vowels:  
        vowel_count += 1  
    elif char in consonants:  
        consonant_count += 1  
    elif char in digits:  
        digit_count += 1  
  
    return vowel_count, consonant_count, digit_count  
  
# Example usage  
test_string = "Hello World 123!"  
v, c, d = analyze_string(test_string)  
  
print(f"Original string: {test_string}")  
print(f"Number of vowels: {v}")  
print(f"Number of consonants: {c}")  
print(f"Number of digits: {d}")
```

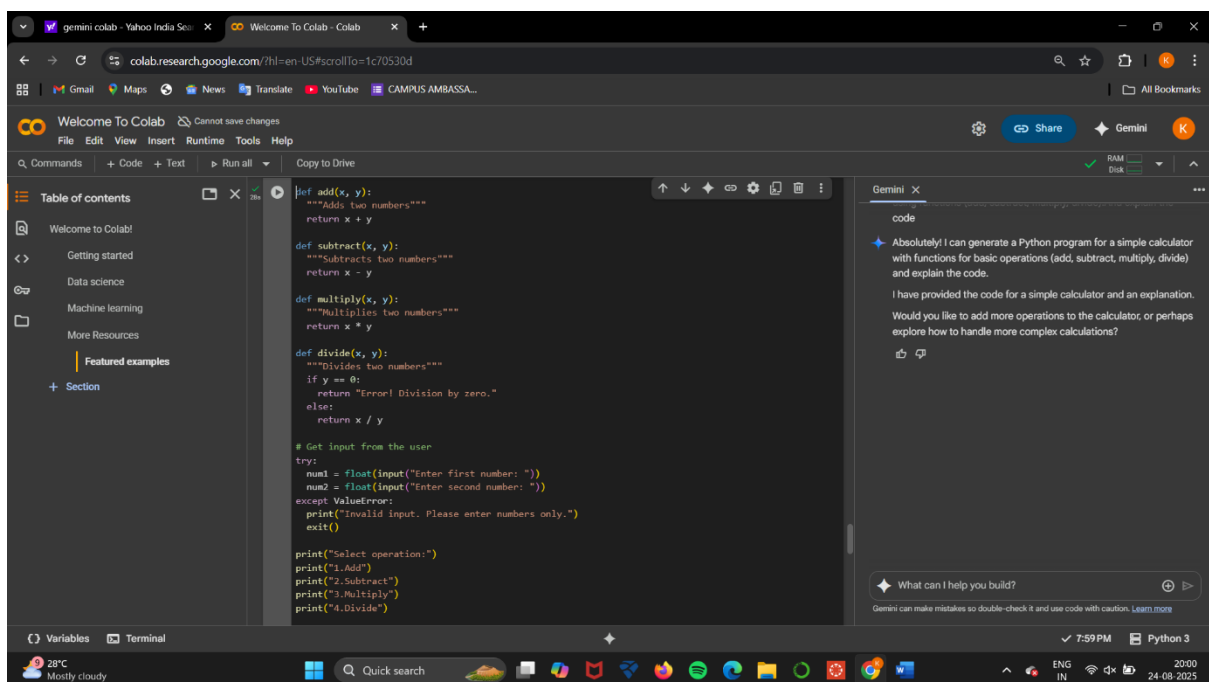
Original string: Hello World 123!
Number of vowels: 3
Number of consonants: 7
Number of digits: 3

TASK4

TASK4 DESCRIPTION:- Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.

PROMPT:-Generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide).And explain the code.

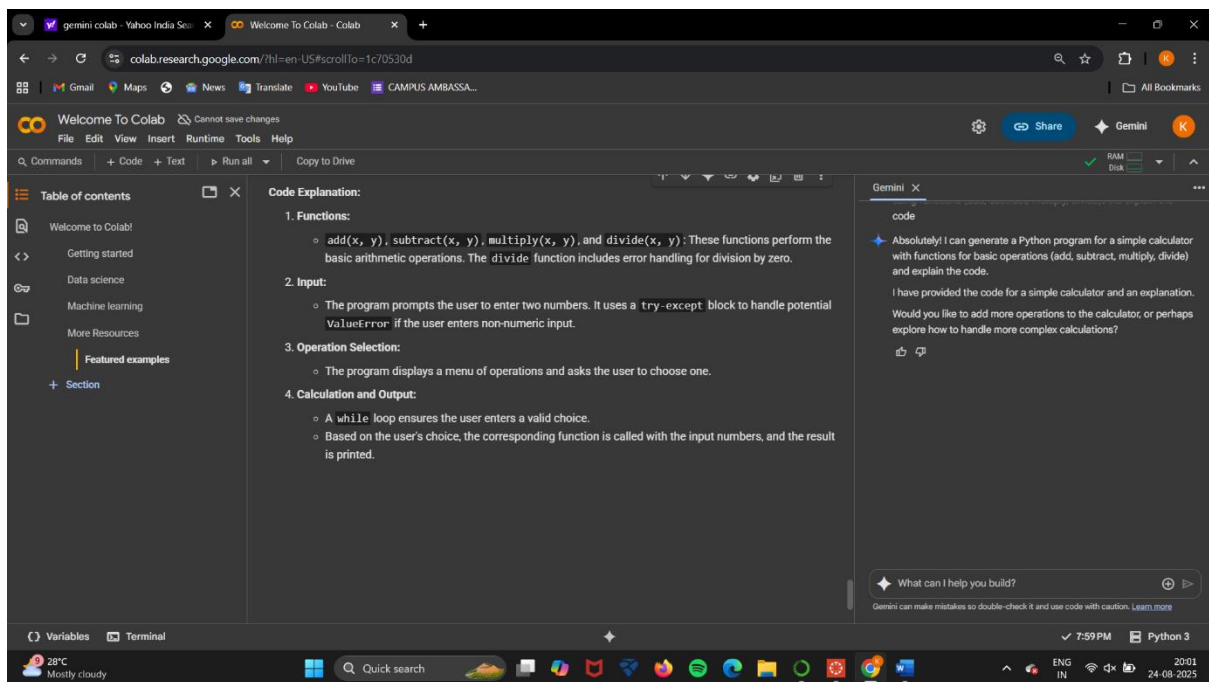
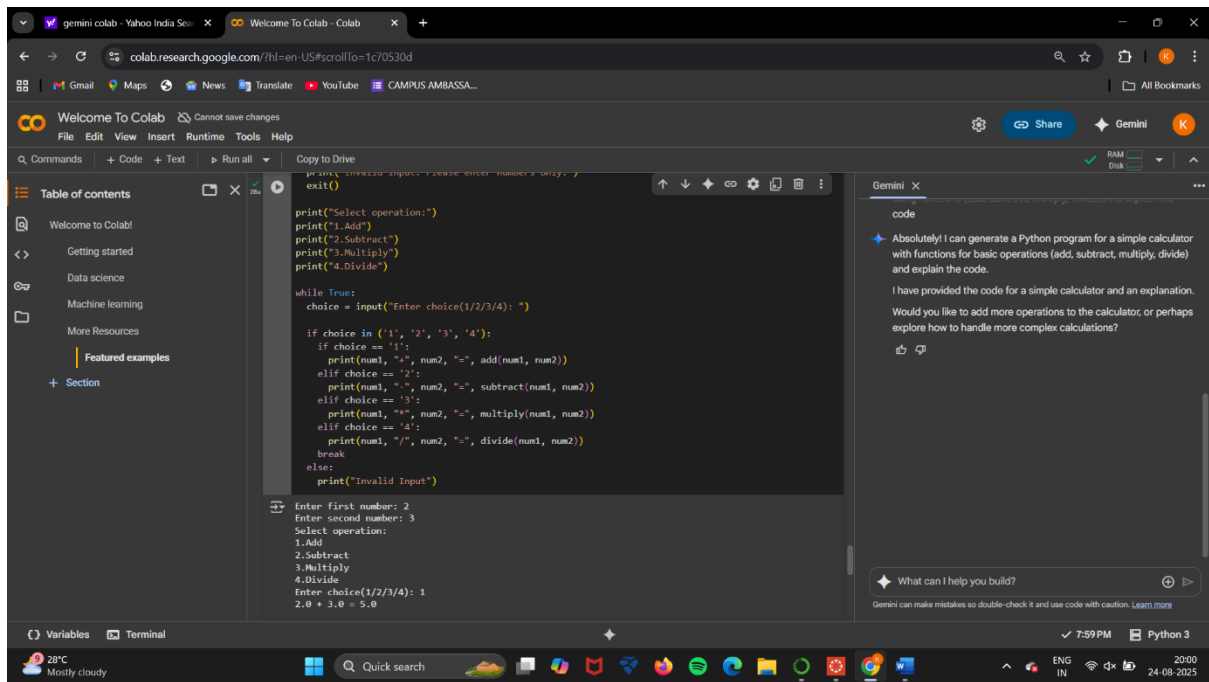
CODE:-



The screenshot shows a Google Colab notebook with a Python program for a simple calculator. The code defines four functions: `add`, `subtract`, `multiply`, and `divide`. It then prompts the user to enter two numbers and select an operation. The Gemini chat interface on the right shows a response confirming the code's functionality and offering to explain it.

```
def add(x, y):  
    """Adds two numbers"""  
    return x + y  
  
def subtract(x, y):  
    """Subtracts two numbers"""  
    return x - y  
  
def multiply(x, y):  
    """Multiplies two numbers"""  
    return x * y  
  
def divide(x, y):  
    """Divides two numbers"""  
    if y == 0:  
        return "Error! Division by zero."  
    else:  
        return x / y  
  
# Get input from the user  
try:  
    num1 = float(input("Enter first number: "))  
    num2 = float(input("Enter second number: "))  
except ValueError:  
    print("Invalid input. Please enter numbers only.")  
    exit()  
  
print("Select operation:")  
print("1.Add")  
print("2.Subtract")  
print("3.Multiply")  
print("4.Divide")
```

Gemini X
code
Absolutely! I can generate a Python program for a simple calculator with functions for basic operations (add, subtract, multiply, divide) and explain the code.
I have provided the code for a simple calculator and an explanation. Would you like to add more operations to the calculator, or perhaps explore how to handle more complex calculations?



OBSERVATION:-

I observed how Google Gemini can generate Python programs when provided with clear prompts and how different problem-solving approaches can be compared and analyzed.

- In Task 1, Gemini generated two different sorting implementations: one using the manual Bubble Sort algorithm and the other using Python's built-in `sort()` function. This highlighted the difference between a step-by-step algorithmic approach and an optimized built-in method, making it clear that while algorithms are useful for learning, built-in functions provide efficiency in practical use.

- In Task 2, I observed that Gemini was able to implement a function that processes a string and accurately counts vowels, consonants, and digits. This task demonstrated the AI's ability to handle string manipulation and conditional logic effectively.
- In Task 4, Gemini successfully generated a program for a simple calculator using functions for addition, subtraction, multiplication, and division. More importantly, when asked to explain the code, it provided a step-by-step breakdown, showing how AI can assist not only in code generation but also in teaching and concept explanation.