

**AI ASSISTED CODING**

**LAB-2:** ***Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI***

**Roll No:** 2503A51L34

**Name:** Dhangar Nagamrutha

**Batch:** 24BTCAICSB20

**Task #1:**

**Prompt:** 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.

A screen shot of a computer program

AI-generated content may be incorrect.**Code Generated: Output:**

A screen shot of a computer screen

AI-generated content may be incorrect.

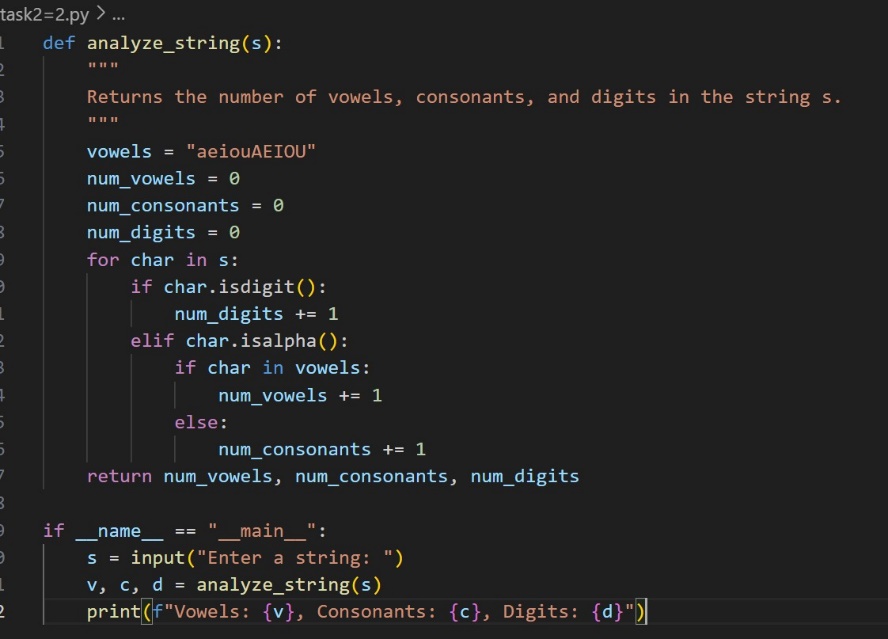
**Observation:**

The program demonstrates how Bubble Sort can be implemented by repeatedly swapping adjacent elements until the list is sorted. It also verifies correctness by comparing the result with Python’s built-in sorted() function. This highlights the difference between a manual sorting algorithm and optimized library functions, showing how both approaches can achieve the same output but with different efficiency.

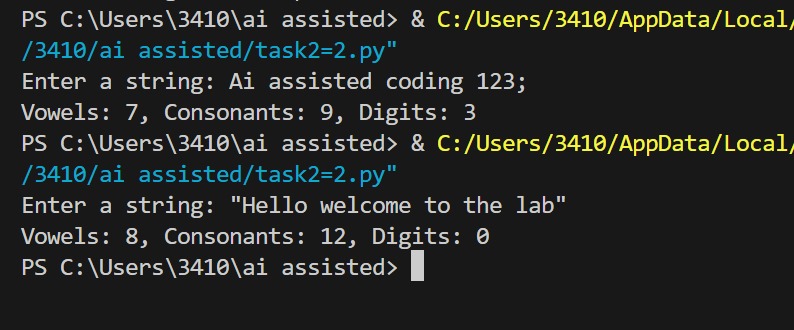
**Task #2:**

**Prompt:** 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.

**Code Generated:**



**Output:**



**Observation:**

The program analyzes a given string to count vowels, consonants, and digits. It uses character checks like isdigit() and isalpha() along with conditional logic to classify each character. This approach shows how string manipulation and conditional statements can be combined to extract meaningful information from user input.

**Task #3:**

**Prompt:** Install and set up Cursor AI. Use it to generate a Python program that performs file handling:

1. Create a text file
2. Write sample text
3. Read and display the content.

**Code Generated:**

A screen shot of a computer

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Observation:**

The program demonstrates basic file handling in Python by creating, writing, and reading a text file. It writes multiple lines into sample.txt and then reads back the content for display. This highlights how file operations such as open(), write(), and read() can be used to manage data storage and retrieval. Copilot generated the structure automatically, showing how it simplifies repetitive coding tasks while still producing correct results.

**Task #4:** 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.

A screen shot of a computer program

AI-generated content may be incorrect.**Code Generated:**

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Observation:**

The program works as a simple calculator with functions for addition, subtraction, multiplication, and division. It uses user input to take numbers and the operation type, then calls the respective function to compute the result. Division by zero is handled with an error message, showing the importance of validation in arithmetic operations. This demonstrates the use of functions, conditional logic, and user interaction in Python.

**Task #5:**

**Prompt:** Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

**Code Generated:**

A computer screen with text on it

AI-generated content may be incorrect.

**Output:**

A computer screen with text and numbers

AI-generated content may be incorrect.

**Observation:**

The program determines whether a given year is a leap year or not by applying divisibility rules. It checks if the year is divisible by 4 but not by 100, or divisible by 400, which is the correct logic for leap year calculation. This illustrates the use of conditional expressions, modular arithmetic, and function-based design in Python.