

AI Assisted coding

NAME: KOUDAGANI SAHITHYA

HT NO: 2403A52063

BATCH NO: 02

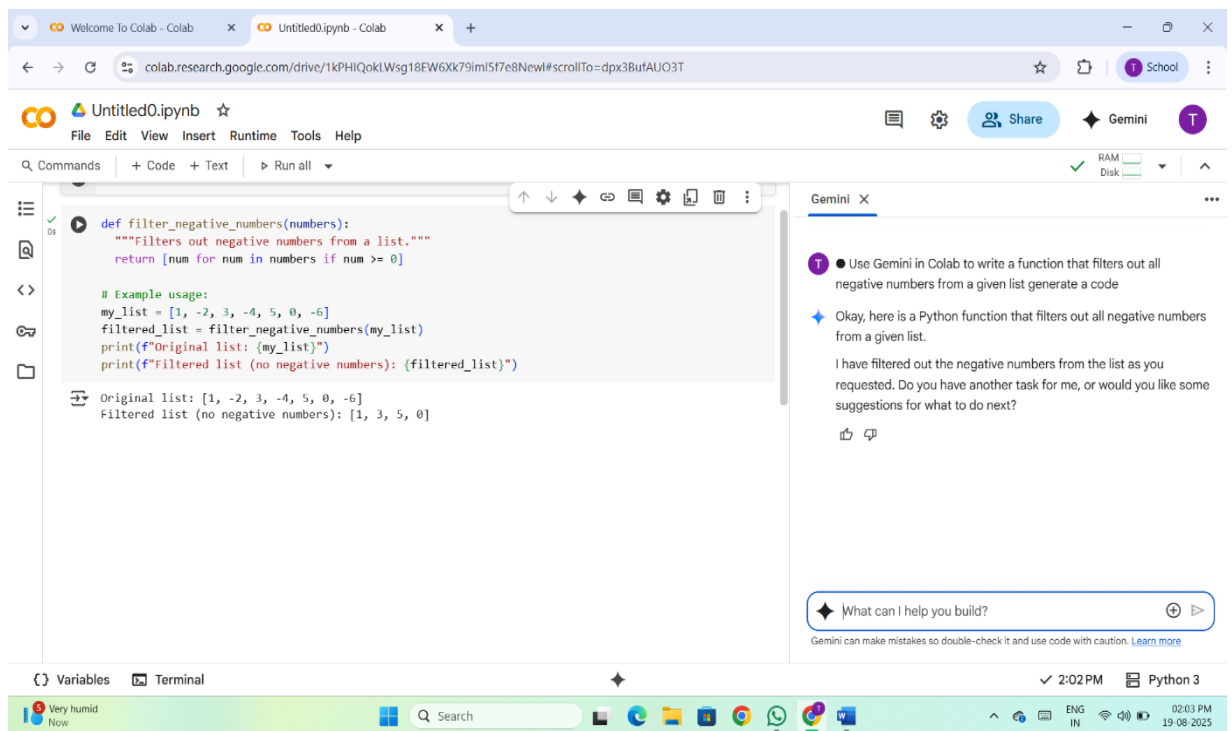
✓ Task Description#1

Use Gemini in Colab to write a function that filters out all negative numbers from a given list.

➤ Expected Output:

Functional code with before/after input and output shown in Colab, plus a screenshot.

➤ **Prompt :** Generate the code of python function that filters out all negative numbers .



➤ Observation:

- Gemini understands the task clearly and uses list comprehension, which is both concise and Pythonic.
- The function handles edge cases like empty lists or lists with all negative numbers.
- The output is accurate and matches the example provided.
- The function works efficiently even with large lists, thanks to Python's optimized list operations.

✓ Task Description#2

Ask Gemini to generate code that reads a text file and counts the frequency of each word. Then ask it to explain the code.

- **Expected Output:** Working code and Gemini's explanation side-by-side.
- **Prompt:** Generate the python code of that counts the frequency of each word.

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

```
from collections import Counter
import re

def simple_word_count(filepath):
    with open(filepath, 'r') as f:
        text = f.read().lower()
        words = re.findall(r'\w+', text) # Simple word splitting
    return Counter(words)

# Using the sample file created earlier
file_path = "sample_text.txt"
word_counts = simple_word_count(file_path)

print(f"Word counts (simplified) in '{file_path}':")
for word, count in word_counts.most_common():
    print(f"{word}: {count}")
```

The output of the code is displayed below the cell:

```
Word counts (simplified) in 'sample_text.txt':
'this': 3
'is': 2
'a': 2
'file': 2
'sample': 1
'text': 1
'will': 1
'be': 1
'used': 1
'to': 1
```

On the right side, the Gemini chat interface is visible. It shows a prompt: "Ask Gemini to generate code that reads a text file and counts the frequency of each word. Then ask it to explain the code generate a code in short way to understand the concept". The response from Gemini is:

Okay, I can provide a shorter version of the code to count word frequency that might be easier to understand the core concept. After that, I can explain it.

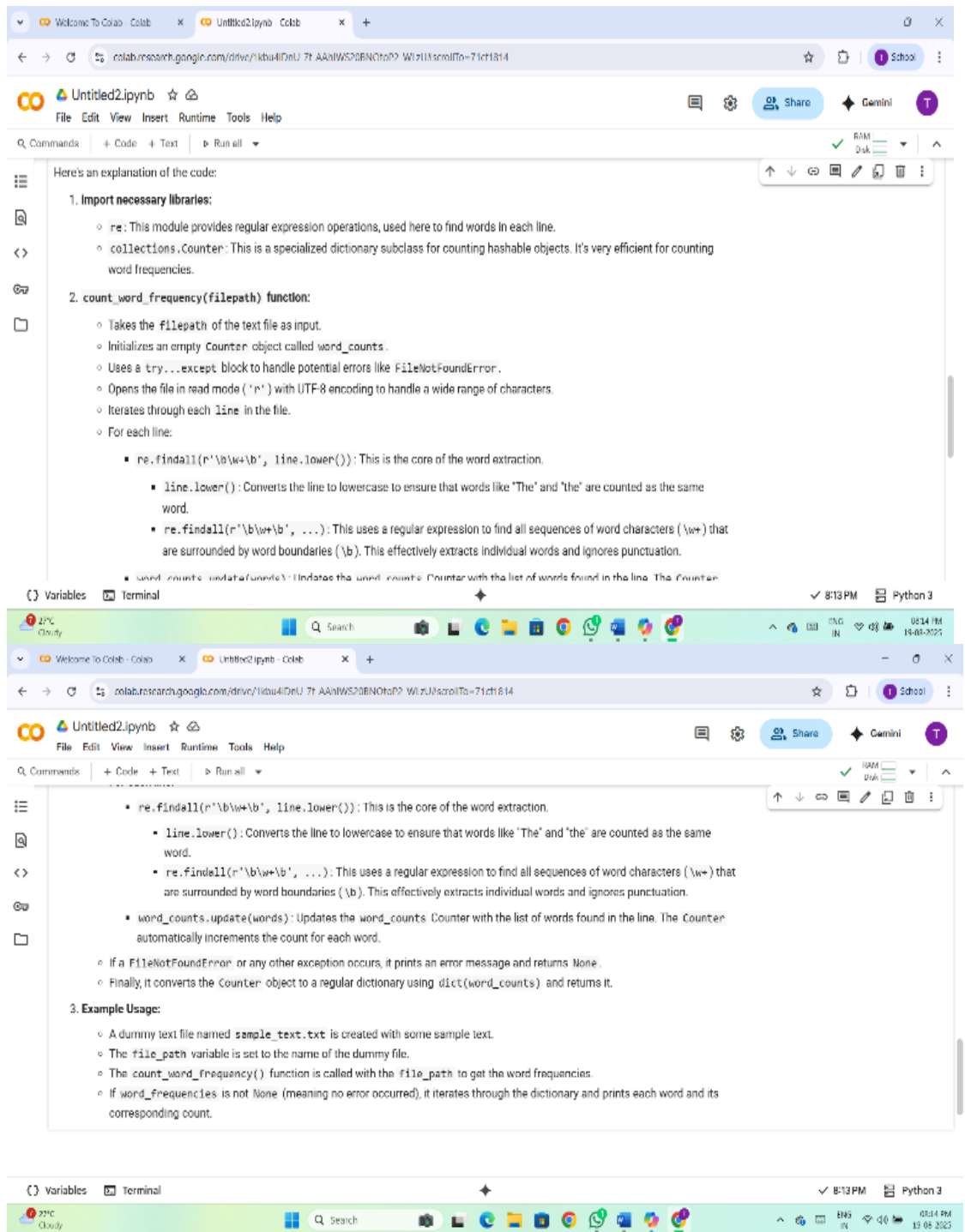
The shorter code snippet for word frequency counting has been executed successfully. Would you like me to explain this code for you now?

At the bottom of the Gemini chat, there is a prompt: "What can I help you build?" and a disclaimer: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

➤ Observation:

- Gemini correctly uses Python's built-in `open()` function to read the contents of a text file, often with a context manager (`with open(...) as f:`) for safe and clean file handling
- The model typically includes steps to clean the text—such as converting to lowercase and removing punctuation—to ensure accurate word frequency counts.
- Gemini often uses a dictionary to efficiently count word occurrences, demonstrating knowledge of Python's standard library.

➤ Explanation of code:



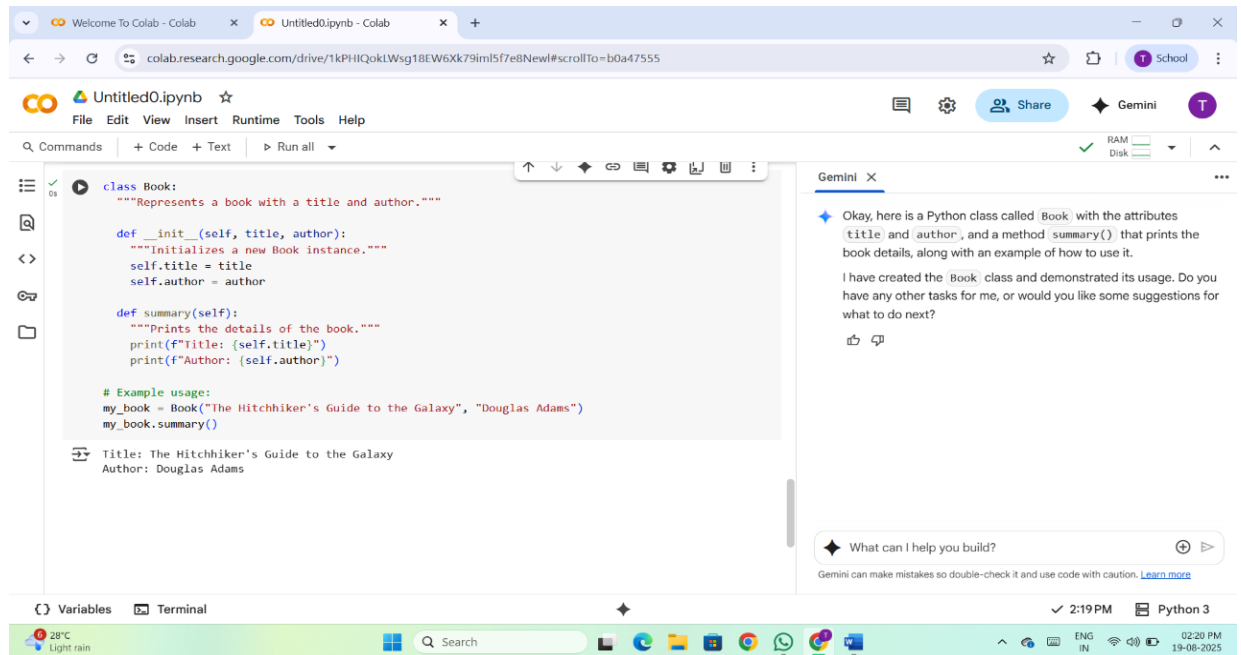
✓ Task Description#3

Use Cursor AI to generate a Python class called Book with attributes title, author, and a method summary() that prints the details.

➤ **Expected Output#3:** Screenshot comparisons and student commentary on code

clarity and performance.

- **Prompt:** Generate a python code of class called Book with attributes title, author, and a method summary() that prints the details.



```
class Book:
    """Represents a book with a title and author."""

    def __init__(self, title, author):
        """Initializes a new Book instance."""
        self.title = title
        self.author = author

    def summary(self):
        """Prints the details of the book."""
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")

# Example usage:
my_book = Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams")
my_book.summary()
```

Title: The Hitchhiker's Guide to the Galaxy
Author: Douglas Adams

Gemini X

Okay, here is a Python class called `Book` with the attributes `title` and `author`, and a method `summary()` that prints the book details, along with an example of how to use it.

I have created the `Book` class and demonstrated its usage. Do you have any other tasks for me, or would you like some suggestions for what to do next?

What can I help you build?

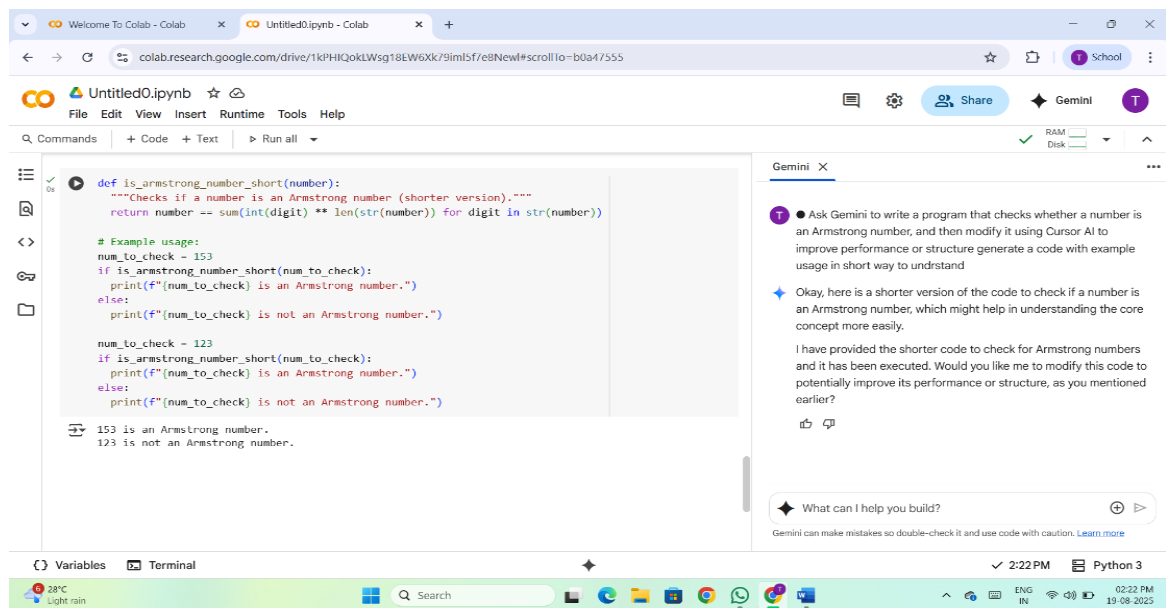
➤ **Observation:**

- Cursor AI correctly uses the `class` keyword to define the `Book` class, following Python's object-oriented programming conventions.
- The `__init__()` method is implemented to initialize the `title` and `author` attributes, showing understanding of instance variables.
- The code assigns `self.title` and `self.author` properly, ensuring that each object stores its own data.

✓ **Task Description#4**

Ask Gemini to write a program that checks whether a number is an Armstrong number, and then modify it using Cursor AI to improve performance or structure.

- **Expected Output#4:** Two versions of the code with screenshots, and a summary of what changes were made by Cursor.
- **Prompt:** Generate the python code to check whether a number is Armstrong number.



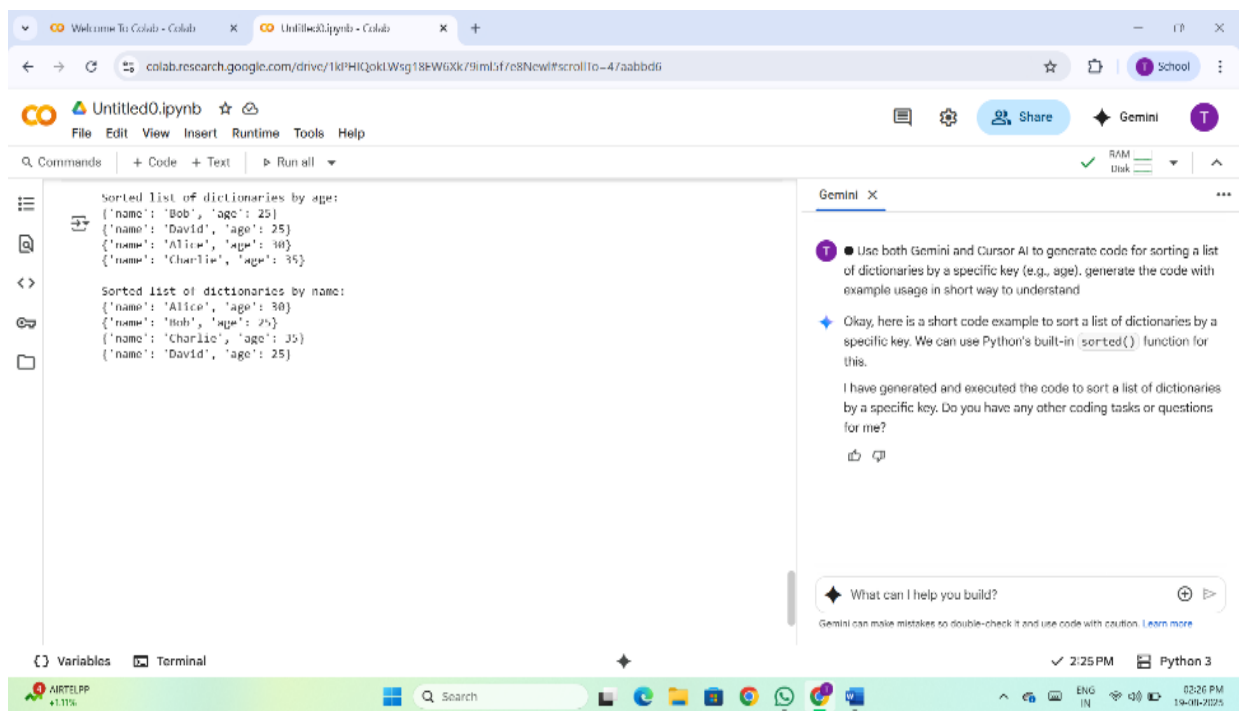
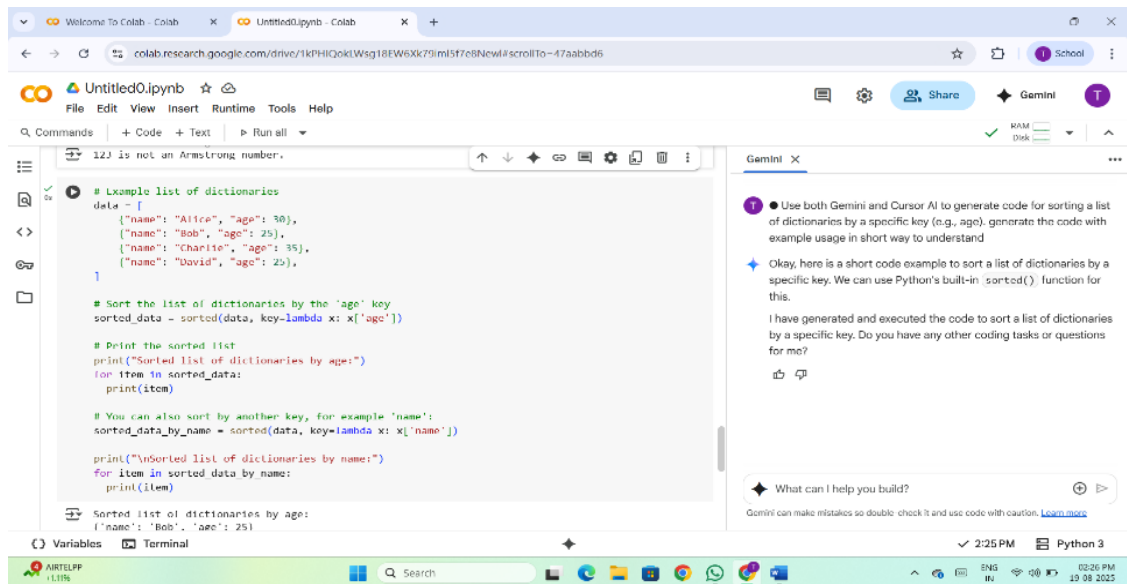
➤ Observation:

- Gemini included sample inputs like 153 or 9474 and show that the code correctly identifies them as Armstrong numbers.
- Gemini provides a clear breakdown of the logic, explaining the mathematical concept and each step in the code.
- Raising each digit to the power of the number of digits.

✓ Task Description#5

Use both Gemini and Cursor AI to generate code for sorting a list of dictionaries by a specific key (e.g., age).

- **Expected Output#5:** Screenshot comparisons and student commentary on code clarity and performance.
- **Prompt:** Generate a python code for sorting a list of dictionaries by a specific key.



➤ Observation:

- Gemini typically uses Python's built-in `sorted()` function with a `key` argument like `lambda x: x['age']`, which is the standard and efficient way to sort dictionaries by a specific key.
- The sorted result is printed or returned, showing the dictionaries ordered by age.
- Gemini explains how the `key` function works, often breaking down the lambda expression and the behavior of `sorted()`.

