ASSIGNMENT – 2

NAME : M.TANVI

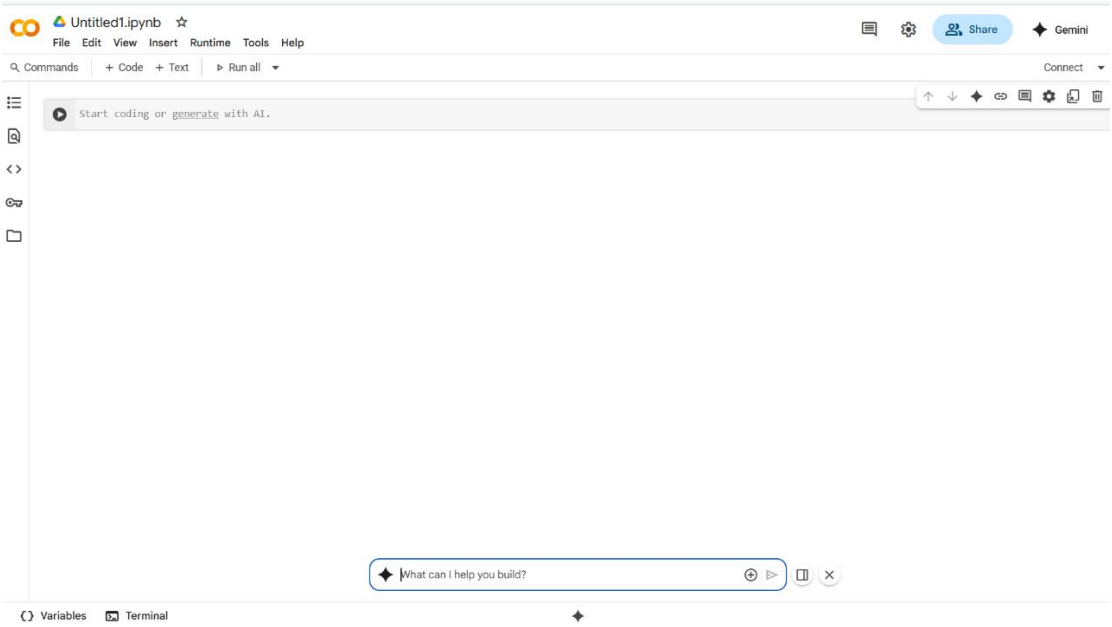
COURSE : AI ASSISTED CODING

ROLL NO : 2403A510A4

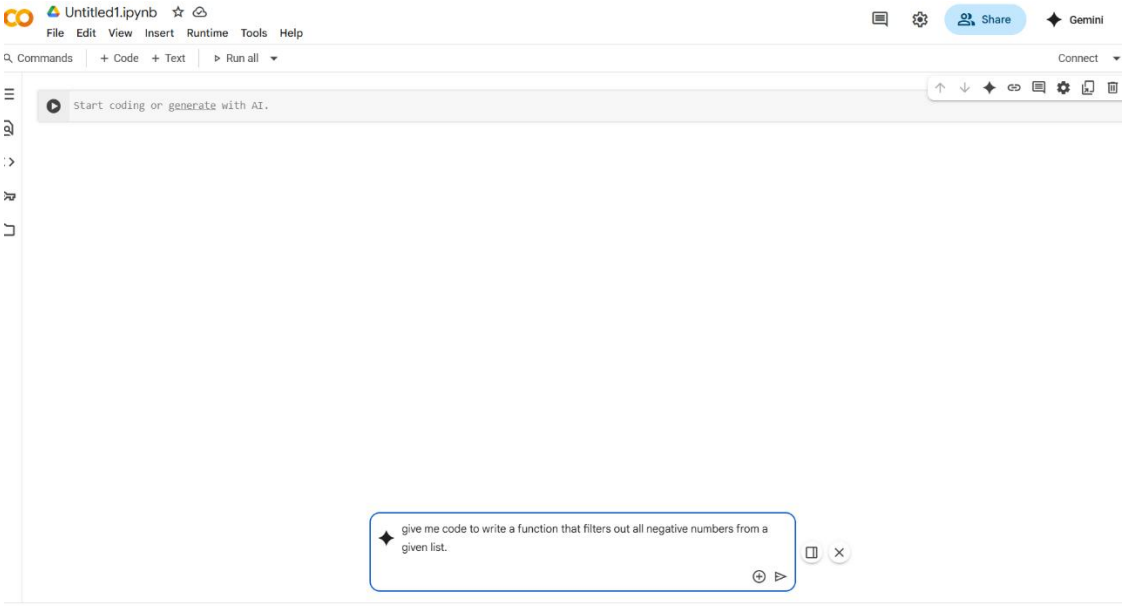
DATE : 19 AUG 2025

TASK1:

OPEN GOOGLE COLAB:



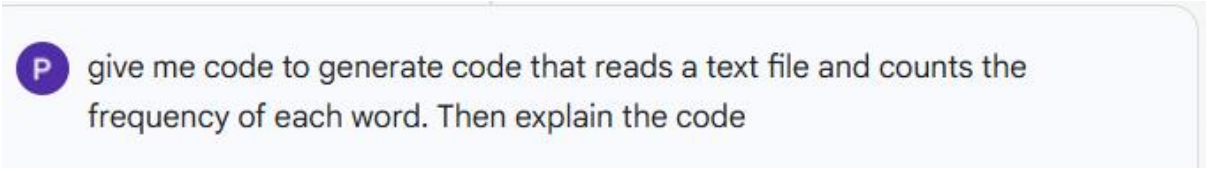
INPUT:



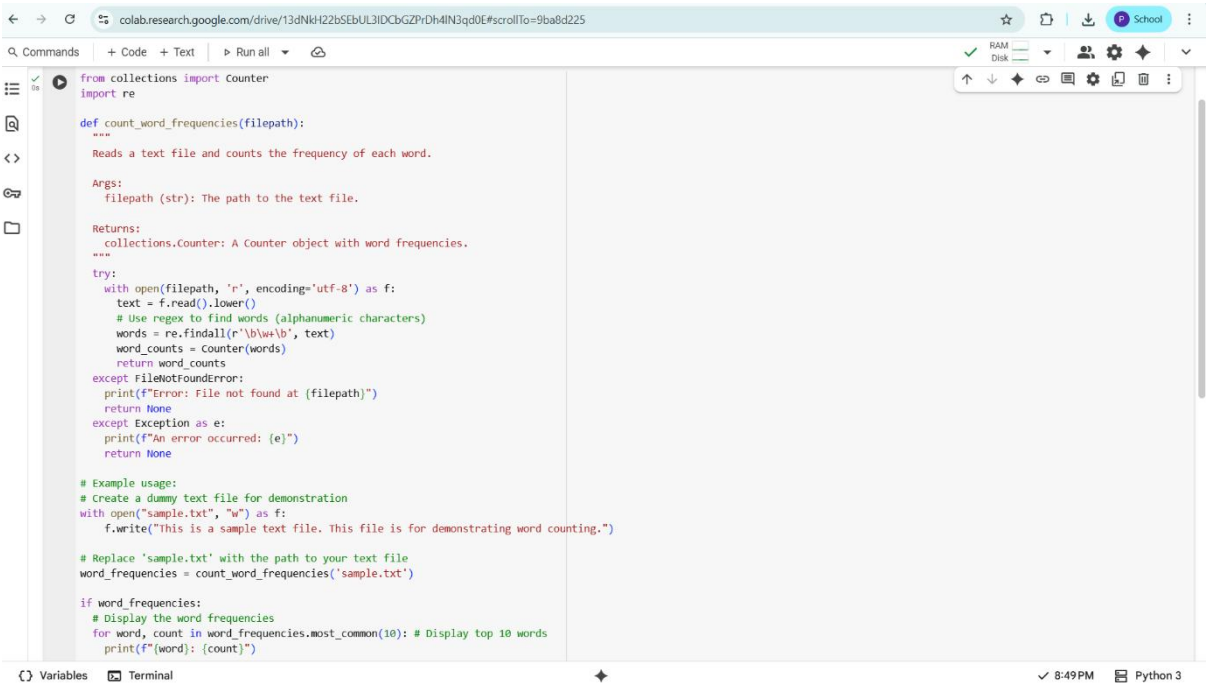
CODE & OUTPUT:

TASK 2:

INPUT:



CODE :



OUTPUT:

```
if word_frequencies:
  # Display the word frequencies
  for word, count in word_frequencies.most_common(10): # Display top 10 words
    print(f"{word}: {count}")
```

```
this: 2
is: 2
file: 2
a: 1
sample: 1
text: 1
for: 1
demonstrating: 1
word: 1
counting: 1
```

EXPLANATION OF CODE BY GEMINI:

Gemini

P give me code to generate code that reads a text file and counts the frequency of each word. Then explain the code

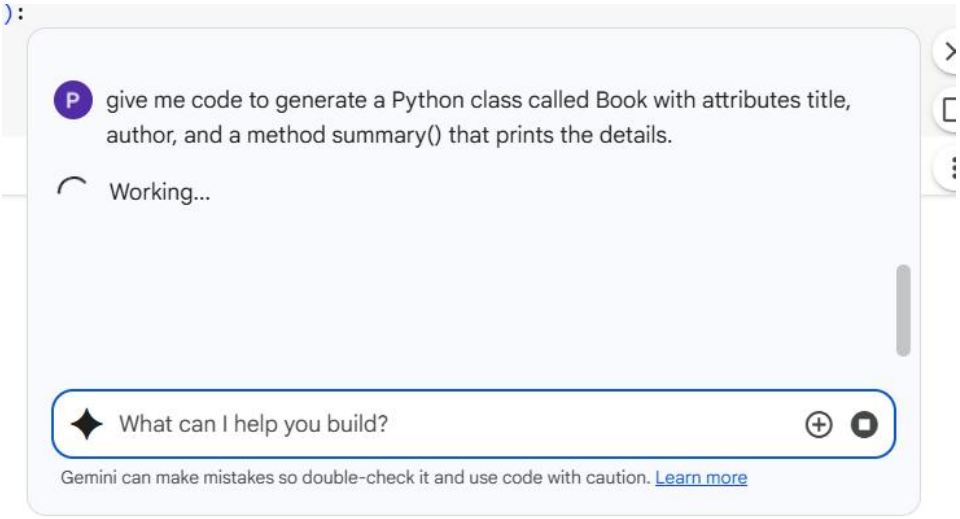
Happy to help! I'll generate the code to read a text file and count word frequencies, and then explain it to you.

This code defines a function `count_word_frequencies` that takes a file path as input. It reads the file, converts the text to lowercase, and uses regular expressions to find all the words. It then uses the `Counter` class from the `collections` module to count the frequency of each word. Finally, it returns a `Counter` object containing the word frequencies. The example usage shows how to create a sample file, call the function, and print the top 10 most frequent words.

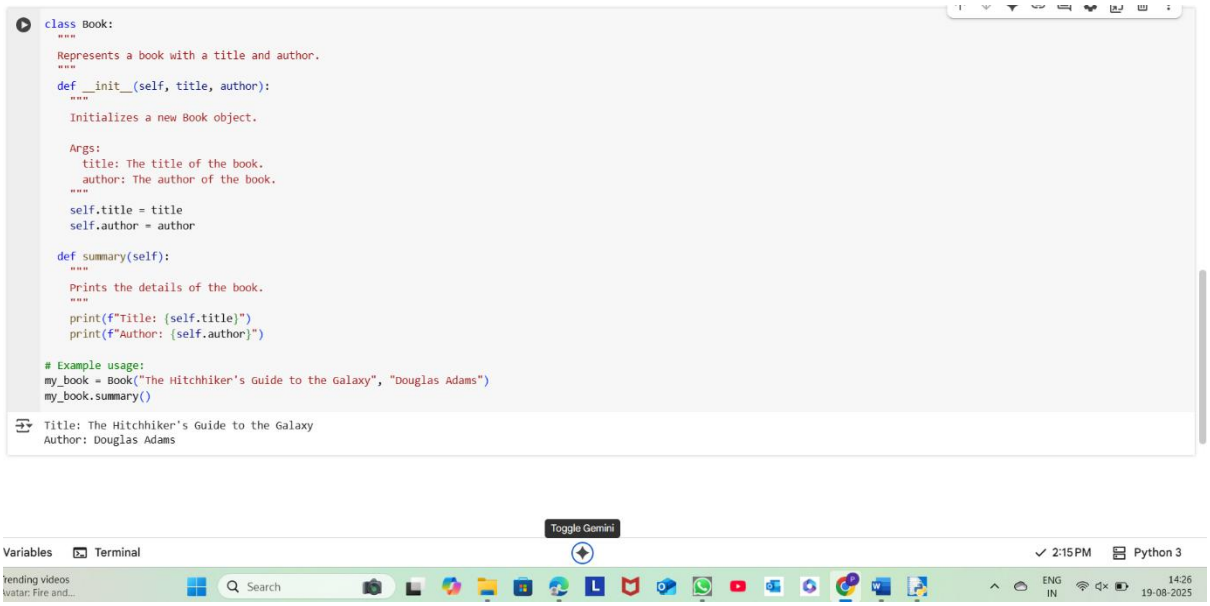
Would you like me to show you how to visualize the word frequencies?

TASK 3:

INPUT IN GEMINI:



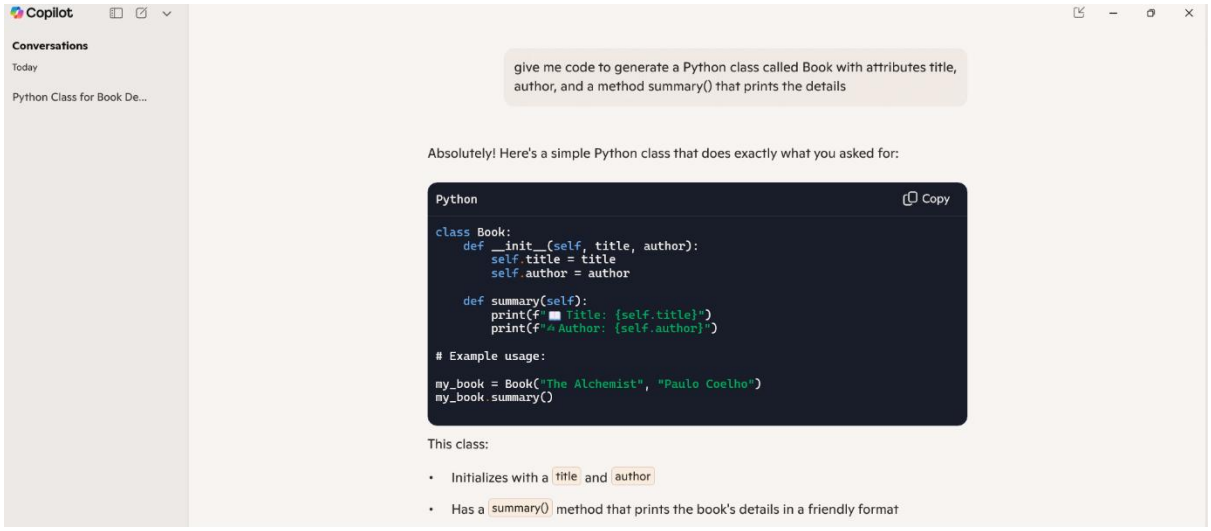
CODE & OUTPUT IN GEMINI:



```
class Book:
  """
  Represents a book with a title and author.
  """
  def __init__(self, title, author):
    """
    Initializes a new Book object.

    Args:
      title: The title of the book.
      author: The author of the book.
    """
    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
```


INPUT & CODE IN COPILOT:

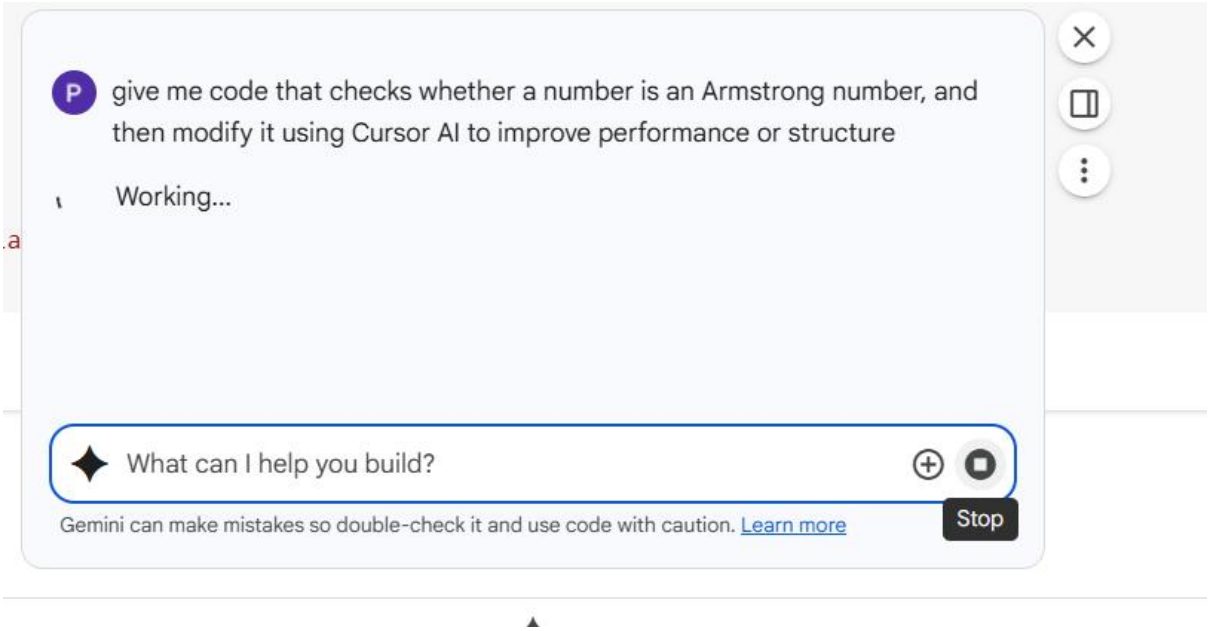


COMMENTING ON CODE CLARITY AND PERFORMANCE:

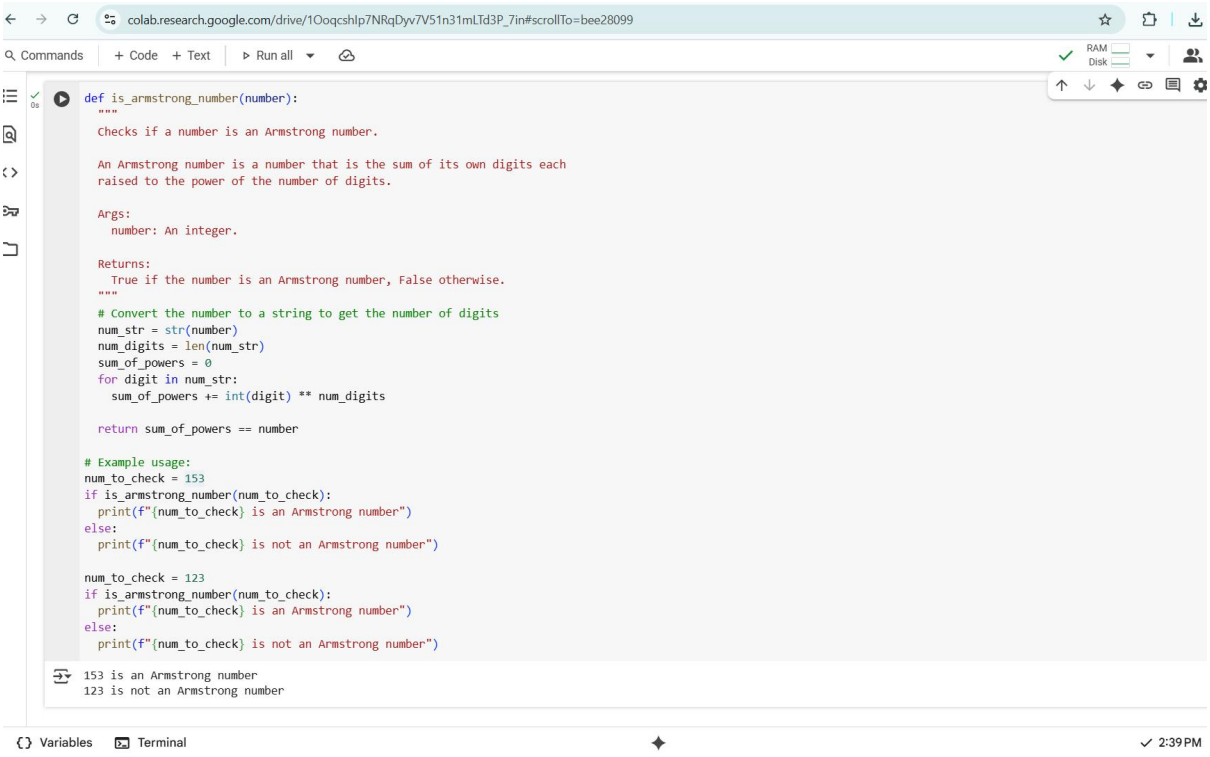
- ➔ IN GOOGLE COLAB WE CAN ABLE TO GENERATE THE CODE AND RUN THE CODE TO GET THE OUTPUT WITHOUT DOWNLOADING ANY PYTHON EXTENSION.
- ➔ IN COPILOT WE CAN ONLY ABLE TO GENERATE THE CODE WE CANNOT RUN AND GET THE OUTPUT LIKE GOOGLE COLAB.

TASK 4:

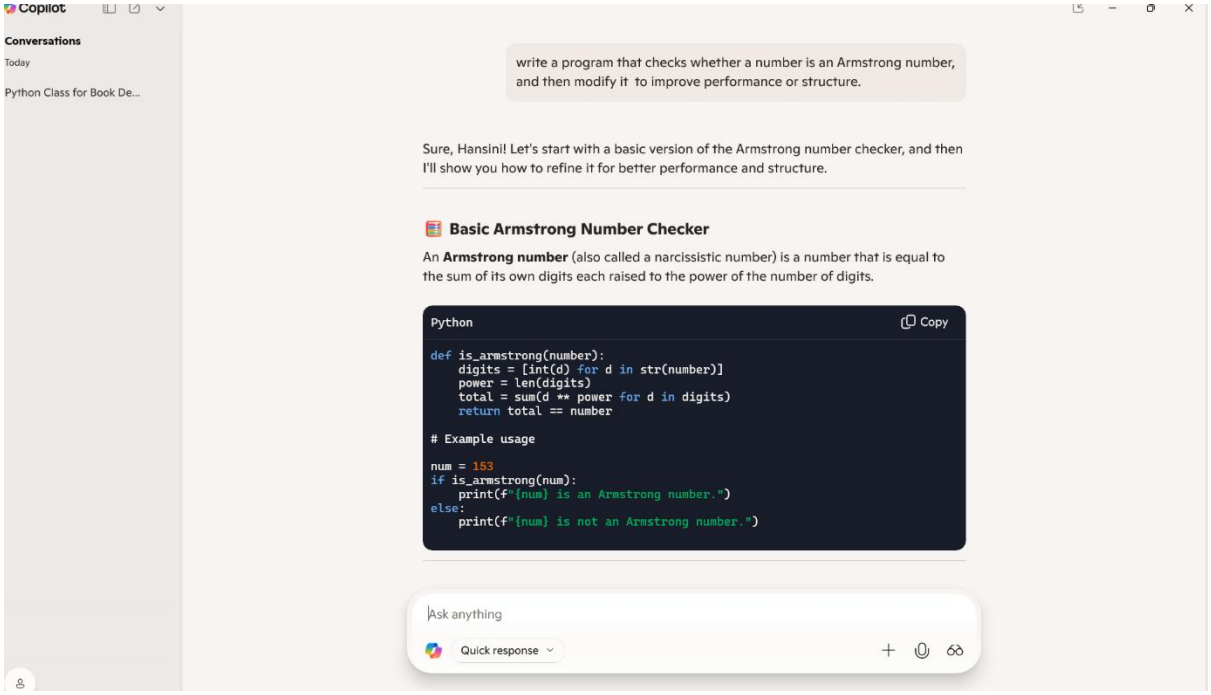
INPUT IN GEMINI:



CODE & OUTPUT:



INPUT & CODE IN COPILOT:

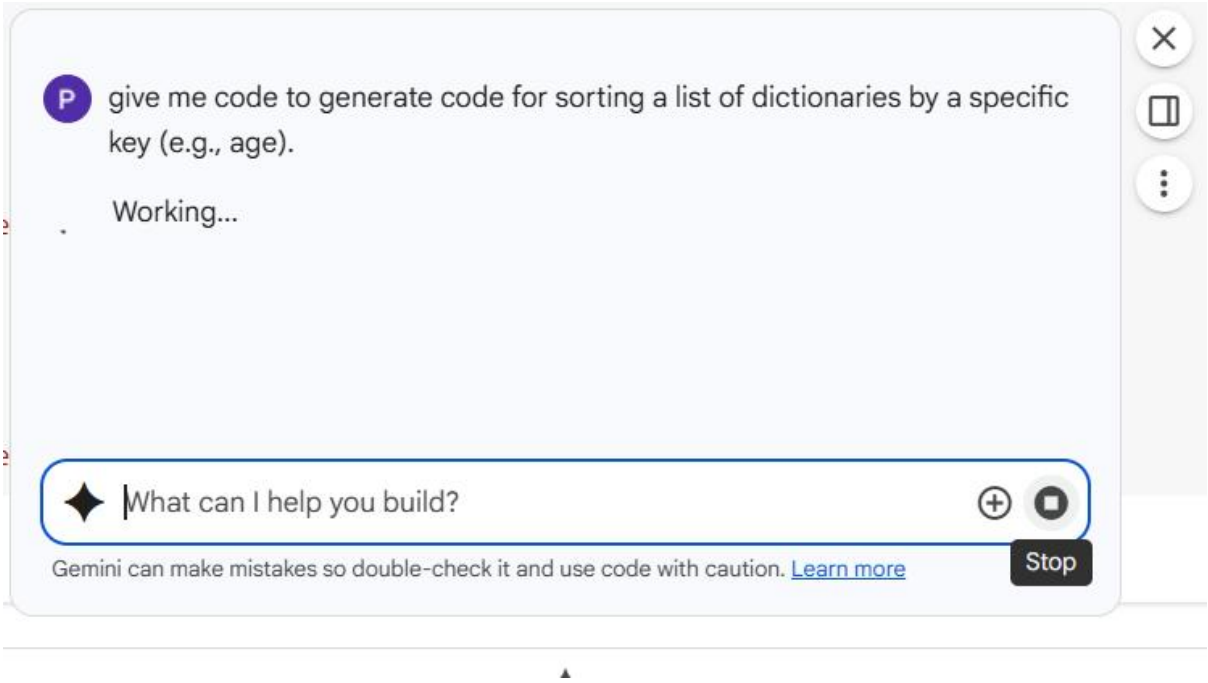


CHANGES IN COMPARISION OF GEMINI & COPILOT:

- ➔ IN GOOGLE COLAB WE CAN ABLE TO GENERATE THE CODE AND RUN THE CODE TO GET THE OUTPUT WITHOUT DOWNLOADING ANY PYTHON EXTENSION.
- ➔ IN COPILOT WE CAN ONLY ABLE TO GENERATE THE CODE WE CANNOT RUN AND GET THE OUTPUT LIKE GOOGLE COLAB.

TASK 5:

INPUT IN GEMINI:

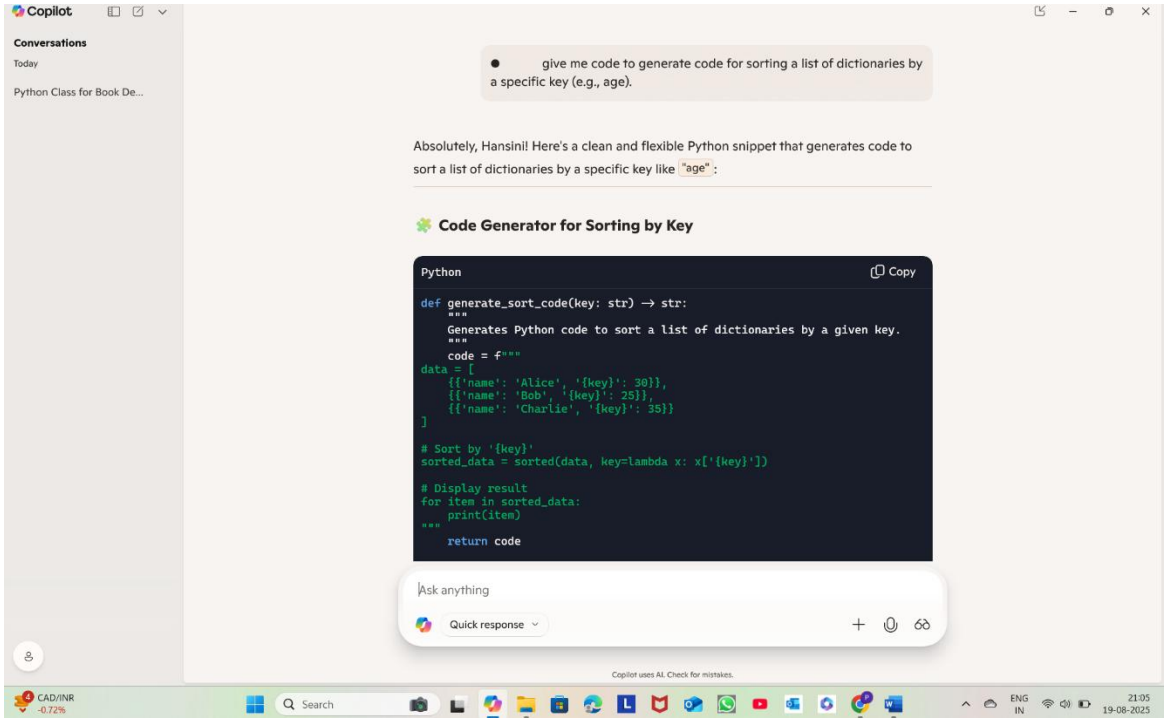


CODE & OUTPUT:

```python
def sort_list_of_dicts(list_of_dicts, sort_key):
  """
  Sorts a list of dictionaries by a specific key.

  Args:
    list_of_dicts: A list of dictionaries.
    sort_key: The key to sort by.

  Returns:
    A new list of dictionaries sorted by the specified key.
  """
  return sorted(list_of_dicts, key=lambda x: x[sort_key])

# Example usage:
people = [
    {'name': 'Alice', 'age': 30, 'city': 'New York'},
    {'name': 'Bob', 'age': 25, 'city': 'London'},
    {'name': 'Charlie', 'age': 35, 'city': 'Paris'},
    {'name': 'David', 'age': 25, 'city': 'Tokyo'}
]

sorted_by_age = sort_list_of_dicts(people, 'age')
print("Sorted by age:")
for person in sorted_by_age:
  print(person)

sorted_by_name = sort_list_of_dicts(people, 'name')
print("\nSorted by name:")
for person in sorted_by_name:
  print(person)
```

```
Sorted by age:
{'name': 'Bob', 'age': 25, 'city': 'London'}
{'name': 'David', 'age': 25, 'city': 'Tokyo'}
{'name': 'Alice', 'age': 30, 'city': 'New York'}
{'name': 'Charlie', 'age': 35, 'city': 'Paris'}

Sorted by name:
{'name': 'Alice', 'age': 30, 'city': 'New York'}
{'name': 'Bob', 'age': 25, 'city': 'London'}
{'name': 'Charlie', 'age': 35, 'city': 'Paris'}
{'name': 'David', 'age': 25, 'city': 'Tokyo'}
```

INPUT & CODE IN COPILOT:



CHANGES IN COMPARISION OF GEMINI & COPILOT:

- ➔ IN GOOGLE COLAB WE CAN ABLE TO GENERATE THE CODE AND RUN THE CODE TO GET THE OUTPUT WITHOUT DOWNLOADING ANY PYTHON EXTENSION.
- ➔ IN COPILOT WE CAN ONLY ABLE TO GENERATE THE CODE WE CANNOT RUN AND GET THE OUTPUT LIKE GOOGLE COLAB.