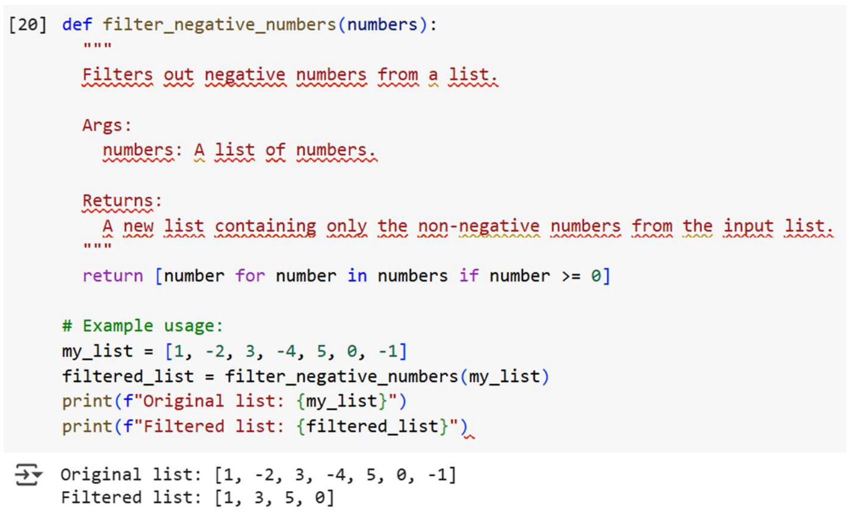
AI ASSITED CODING

Task Description#1

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

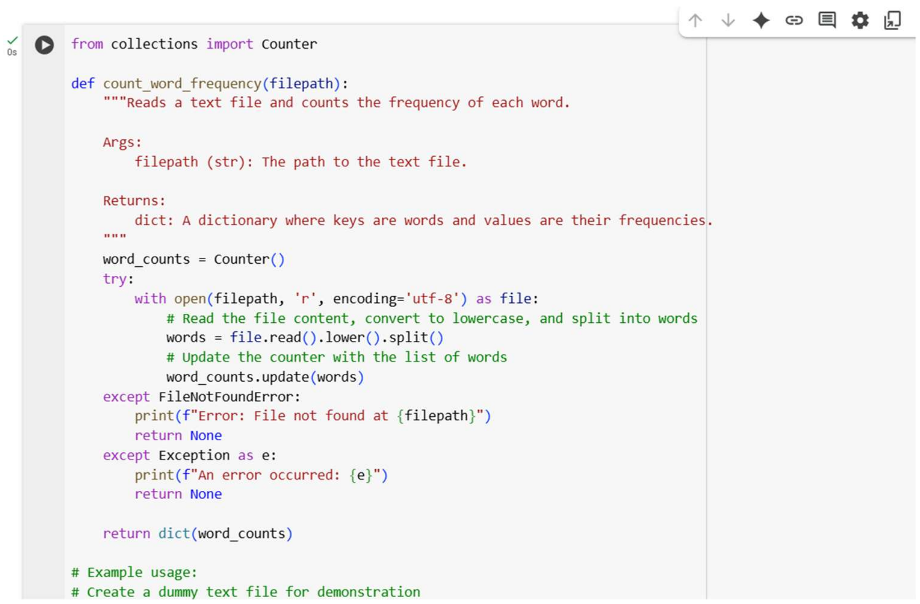


```
[20] def filter_negative_numbers(numbers):
       """
       Filters out negative numbers from a list.

       Args:
         numbers: A list of numbers.

       Returns:
         A new list containing only the non-negative numbers from the input list.
       """
       return [number for number in numbers if number >= 0]

     # Example usage:
     my_list = [1, -2, 3, -4, 5, 0, -1]
     filtered_list = filter_negative_numbers(my_list)
     print(f"Original list: {my_list}")
     print(f"Filtered list: {filtered_list}")
```

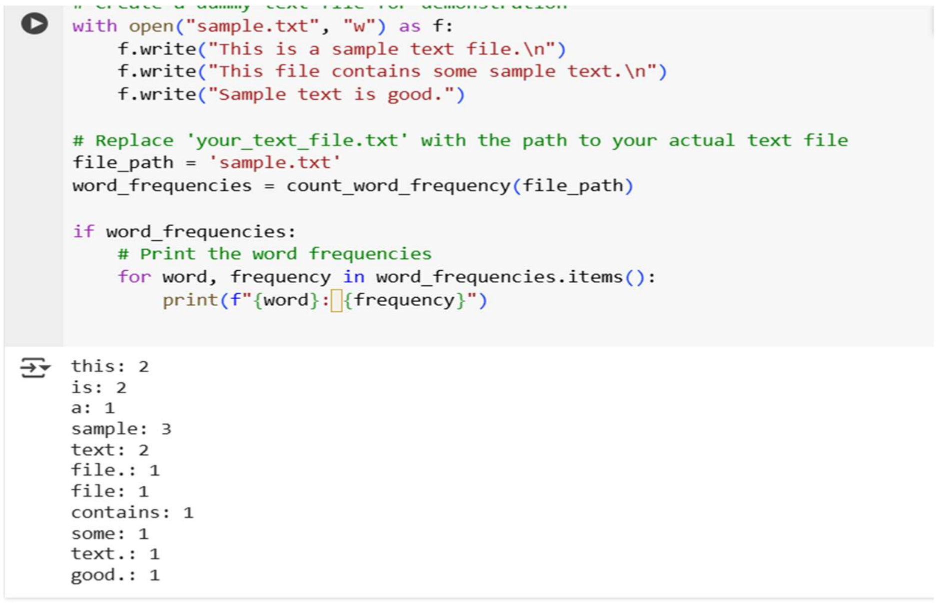
```
Original list: [1, -2, 3, -4, 5, 0, -1]
Filtered list: [1, 3, 5, 0]
```

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.



```
from collections import Counter

def count_word_frequency(filepath):
    """Reads a text file and counts the frequency of each word.

    Args:
        filepath (str): The path to the text file.

    Returns:
        dict: A dictionary where keys are words and values are their frequencies.
    """
    word_counts = Counter()
    try:
        with open(filepath, 'r', encoding='utf-8') as file:
            # Read the file content, convert to lowercase, and split into words
            words = file.read().lower().split()
            # Update the counter with the list of words
            word_counts.update(words)
    except FileNotFoundError:
        print(f"Error: File not found at {filepath}")
        return None
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

    return dict(word_counts)

# Example usage:
# Create a dummy text file for demonstration
```

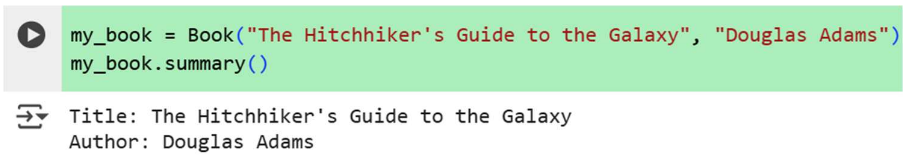


```
# Create a dummy text file for demonstration
with open("sample.txt", "w") as f:
    f.write("This is a sample text file.\n")
    f.write("This file contains some sample text.\n")
    f.write("Sample text is good.")

# Replace 'your_text_file.txt' with the path to your actual text file
file_path = 'sample.txt'
word_frequencies = count_word_frequency(file_path)

if word_frequencies:
    # Print the word frequencies
    for word, frequency in word_frequencies.items():
        print(f"{word}: {frequency}")
```

```
this: 2
is: 2
a: 1
sample: 3
text: 2
file.: 1
file: 1
contains: 1
some: 1
text.: 1
good.: 1
```

Task Description#3

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

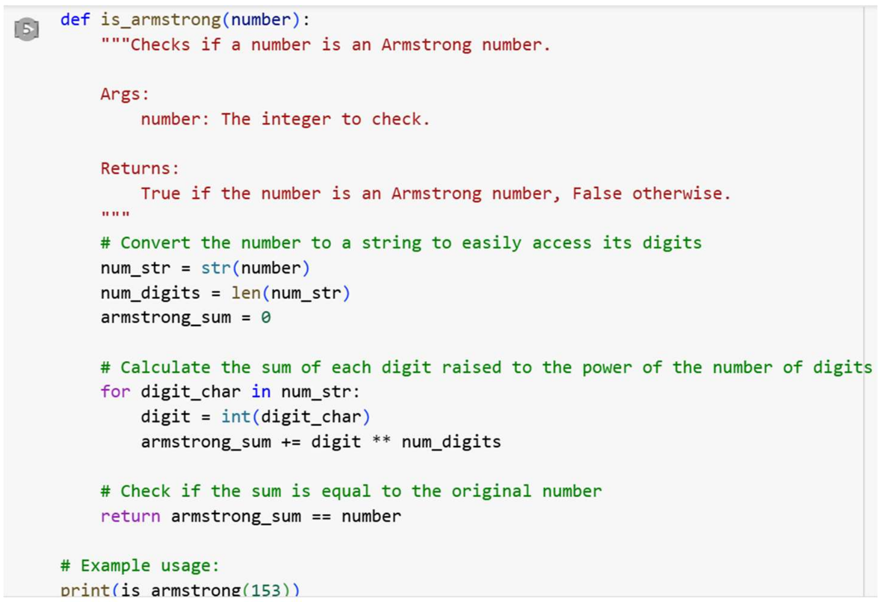


```
my_book = Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams")
my_book.summary()
```

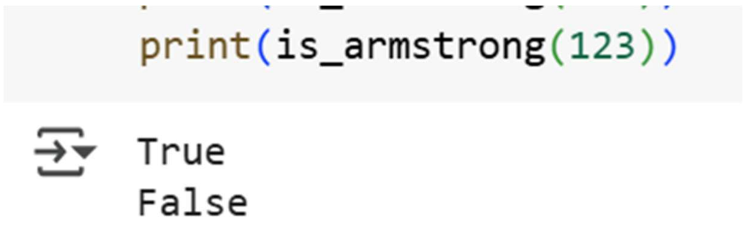
```
Title: The Hitchhiker's Guide to the Galaxy
Author: Douglas Adams
```

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



```
def is_armstrong(number):
    """Checks if a number is an Armstrong number.

    Args:
        number: The integer to check.

    Returns:
        True if the number is an Armstrong number, False otherwise.
    """
    # Convert the number to a string to easily access its digits
    num_str = str(number)
    num_digits = len(num_str)
    armstrong_sum = 0

    # Calculate the sum of each digit raised to the power of the number of digits
    for digit_char in num_str:
        digit = int(digit_char)
        armstrong_sum += digit ** num_digits

    # Check if the sum is equal to the original number
    return armstrong_sum == number

# Example usage:
print(is_armstrong(153))
```

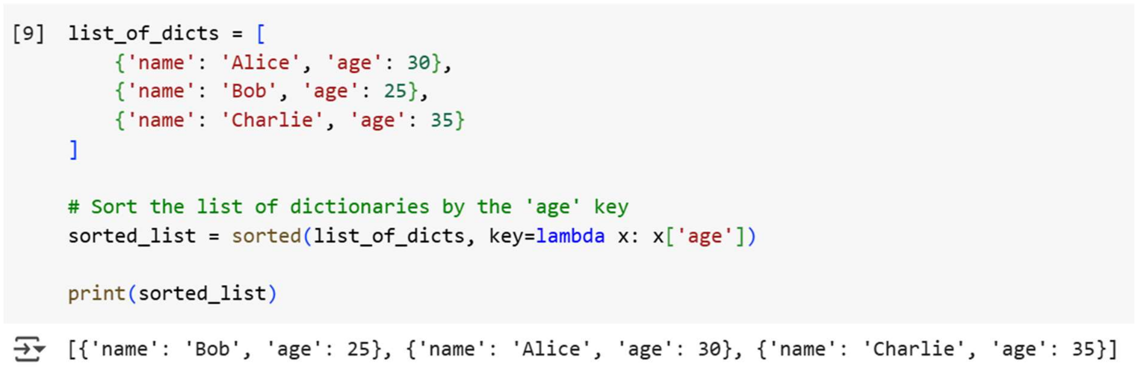


```
print(is_armstrong(123))
```

```
True
False
```

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).



```
[9] list_of_dicts = [
        {'name': 'Alice', 'age': 30},
        {'name': 'Bob', 'age': 25},
        {'name': 'Charlie', 'age': 35}
    ]

    # Sort the list of dictionaries by the 'age' key
    sorted_list = sorted(list_of_dicts, key=lambda x: x['age'])

    print(sorted_list)
```

```
[{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'age': 35}]
```