LAB ASSIGNMENT(2403A52399):

PROMPT1: Generate a python code with a function that filters out all negative numbers from a given list.

Code: def filter\_negative\_numbers(numbers):

  return [number for number in numbers if number >= 0]

my\_list = [1, -2, 3, -4, 5, 0, -6]

filtered\_list = filter\_negative\_numbers(my\_list)

print(filtered\_list)

# output:

[1, 3, 5, 0]

# Prompt2: generate a python code that reads a text file and counts the frequency of each word.

Code: from collections import Counter

import re

def count\_word\_frequency(file\_path):

  word\_counts = Counter()

  with open(file\_path, 'r', encoding='utf-8') as file:

    for line in file:

      words = re.findall(r'\b\w+\b', line.lower())

      word\_counts.update(words)

  return word\_counts

with open('sample.txt', 'w', encoding='utf-8') as f:

    f.write("This is a sample text file.\n")

    f.write("This file is for demonstrating word frequency counting.\n")

    f.write("Sample text, sample file.")

file\_path = 'sample.txt'

word\_frequencies = count\_word\_frequency(file\_path)

print(word\_frequencies)

# output: Counter({'sample': 3, 'file': 3, 'this': 2, 'is': 2, 'text': 2, 'a': 1, 'for': 1, 'demonstrating': 1, 'word': 1, 'frequency': 1, 'counting': 1})

# prompt3: generate a Python class called Book with attributes title, author, and a method summary() that prints the details

Code:

def \_\_init\_\_(*self*, *title*: str, *author*: str) -> None:

*self*.title = *title*

*self*.author = *author*

def summary(*self*) -> None:

        """Print the book's details to stdout."""

        print(f"Title: {*self*.title} | Author: {*self*.author}")

*if* \_\_name\_\_ == "\_\_main\_\_":

    book = Book("1984", "George Orwell")

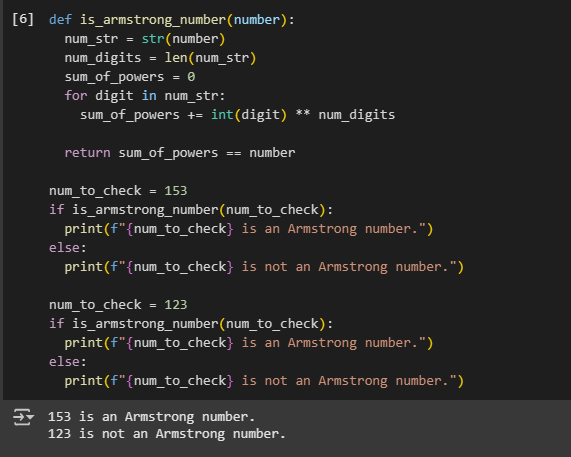
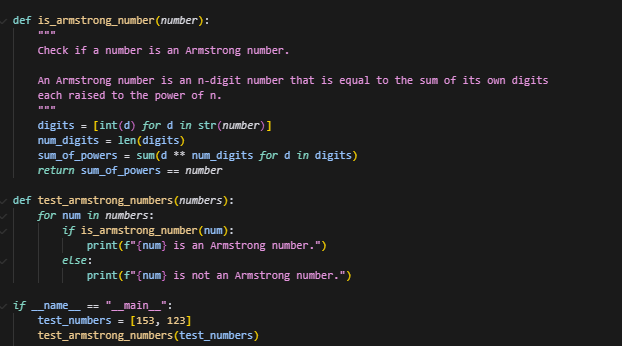
    book.summary()

# output:

Title: 1984 | Author: George Orwell

Prompt4: write a program that checks whether a number is an Armstrong number, and then modify it using Cursor AI to improve performance or structure.

# Code:



# Prompt5: generate python code for sorting a list of dictionaries by a specific key (e.g., age).

# Code:



directories = [

        {"name": "dir1", "age": 5},

        {"name": "dir2", "age": 2},

        {"name": "dir3", "age": 8}

    ]

*# Sort directories by 'age'*

    sorted\_directories = sorted(directories, *key*=lambda *d*: *d*['age'])

    print("Directories sorted by age:")

*for* directory *in* sorted\_directories:

        print(directory)

*# Example: Sorting a list of dictionaries by the 'age' key in descending order*

        sorted\_by\_age\_desc = sorted(directories, *key*=lambda *d*: *d*['age'], *reverse*=True)

        print("Directories sorted by age (descending):")

*for* dir\_desc *in* sorted\_by\_age\_desc:

            print(dir\_desc)