***Python Assignment Questions***

1. **File Analysis: Count Words, Lines, and Characters**

**Objective**:

Write a Python script that reads a text file and computes:

1. The total number of lines.
2. The total number of words.
3. The total number of characters.

**Requirements**:

1. Accept the file path as input.
2. Handle cases where the file does not exist gracefully.

**Program:**

# 1. File Analysis: Count Words, Lines, and Characters

# Function to analyze the file

def analyze\_file(file\_path):

    try:

        #Reading the text file

        # Open the file in read mode ('r')

        with open(file\_path, 'r') as file:

            lines = file.readlines()  # Read all lines from the file

        #Computing required information

            # A. Total number of lines in the file (by calculating the length of the list of lines)

            no\_of\_lines = len(lines)

            # B. Total number of words (by splitting each line into words and summing up the lengths of lists of these words)

            no\_of\_words = sum(len(line.split()) for line in lines)

            # C. Total number of characters (by summing up the lengths of all lines, including spaces and punctuation)

            no\_of\_char = sum(len(line) for line in lines)

        # Displaying the results

        print(f"Total Lines: {no\_of\_lines}")

        print(f"Total Words: {no\_of\_words}")

        print(f"Total Characters: {no\_of\_char}")

    except FileNotFoundError:

        # File does not exist

        print(f"Error: The file at '{file\_path}' does not exist. Please check the path and try again.")

# Main program

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

    print("Solution to Python assignment task 1")

    file\_path = input("Enter the path to the text file: ") #Accepting the file path as input.

    analyze\_file(file\_path)

1. **Generate Fibonacci Sequence up to N and Find Primes**

**Objective**:

Write a Python function that:

1. Generates the Fibonacci sequence up to a given number N.
2. Filters and prints only the prime numbers from the sequence.

**Program:**

# 2. Generate Fibonacci Sequence up to N and Find Primes

# Function to generate Fibonacci sequence

def generate\_fibonacci(n):

    fibonacci = [0, 1]  # Starting values

    while True:

        next\_num = fibonacci[-1] + fibonacci[-2]  # Sum of last two numbers

        if next\_num > n:

            break

        fibonacci.append(next\_num)

    return fibonacci

# Function to check if a number is prime

def is\_prime(num):

    if num < 2:  # Numbers less than or equal to 1 are not prime

        return False

    for i in range(2, int(num \*\* 0.5) + 1):  # Check divisors up to square root of num

        if num % i == 0:

            return False  # Not prime if divisible

    return True

# Main function to select primes from the Fibonacci sequence

def primes\_in\_fibonacci(n):

    fibonacci\_sequence = generate\_fibonacci(n)  # Generate Fibonacci sequence

    primes = [num for num in fibonacci\_sequence if is\_prime(num)]  # Filter primes

    print("Fibonacci Sequence up to", n, ":", fibonacci\_sequence)

    print("Prime Numbers in Fibonacci Sequence:", primes)

N = int(input("Enter a number: "))

primes\_in\_fibonacci(N)

**3. Find the Second Largest Number in a List**  
  
 **Objective**:

Write a Python function that takes a list of integers as input and returns the second largest number in the list.

**Requirements**:

1. Handle cases where the list has duplicates.
2. If the list has fewer than two unique numbers, return None.

**Program:**

# 3.Find the Second Largest Number in a List

def second\_largest(nums):

    unique\_nums = list(set(nums))  # Remove duplicates

    if len(unique\_nums) < 2:

        return None

    unique\_nums.sort(reverse=True)  # Sort in descending order

    return unique\_nums[1]

nums = list(map(int, input("Enter numbers separated by spaces: ").split()))

print(f"The second largest number is: {second\_largest(nums)}")

4. **Check If a String is a Valid Anagram**  
**Objective**: Write a Python function that checks whether two strings are anagrams of each other.

**Requirements**:

1. Ignore case sensitivity.
2. Consider only alphanumeric characters (ignore spaces and punctuation)

**Program:**

#4. Check If a String is a Valid Anagram

import string

def is\_anagram(str1, str2):

    # Remove spaces and punctuation, and make everything lowercase to ignore case sensitivity

    str1 = ''.join(char.lower() for char in str1 if char.isalnum())

    str2 = ''.join(char.lower() for char in str2 if char.isalnum())

    # Check if the sorted characters in both strings are the same

    return sorted(str1) == sorted(str2)

#For example

str1 = "Listen!"

str2 = "Silent"

if is\_anagram(str1, str2):

    print("Yes, the strings are anagrams!")

else:

    print("No, the strings are not anagrams.")