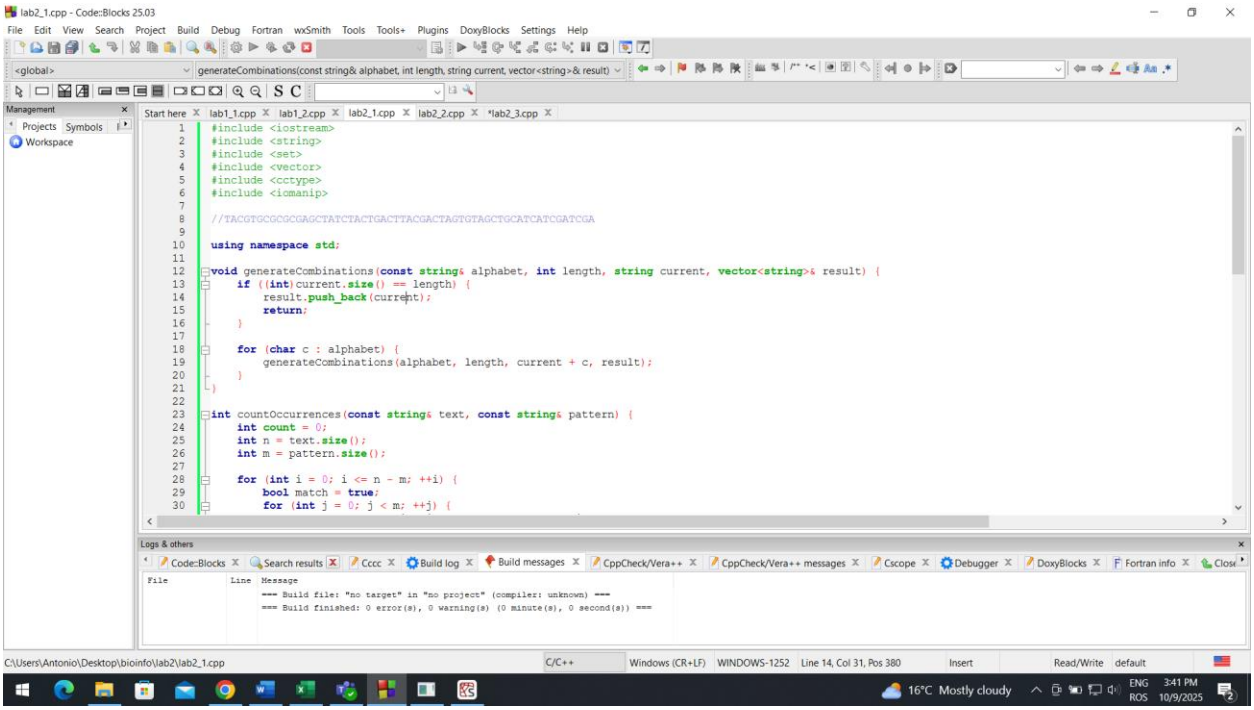
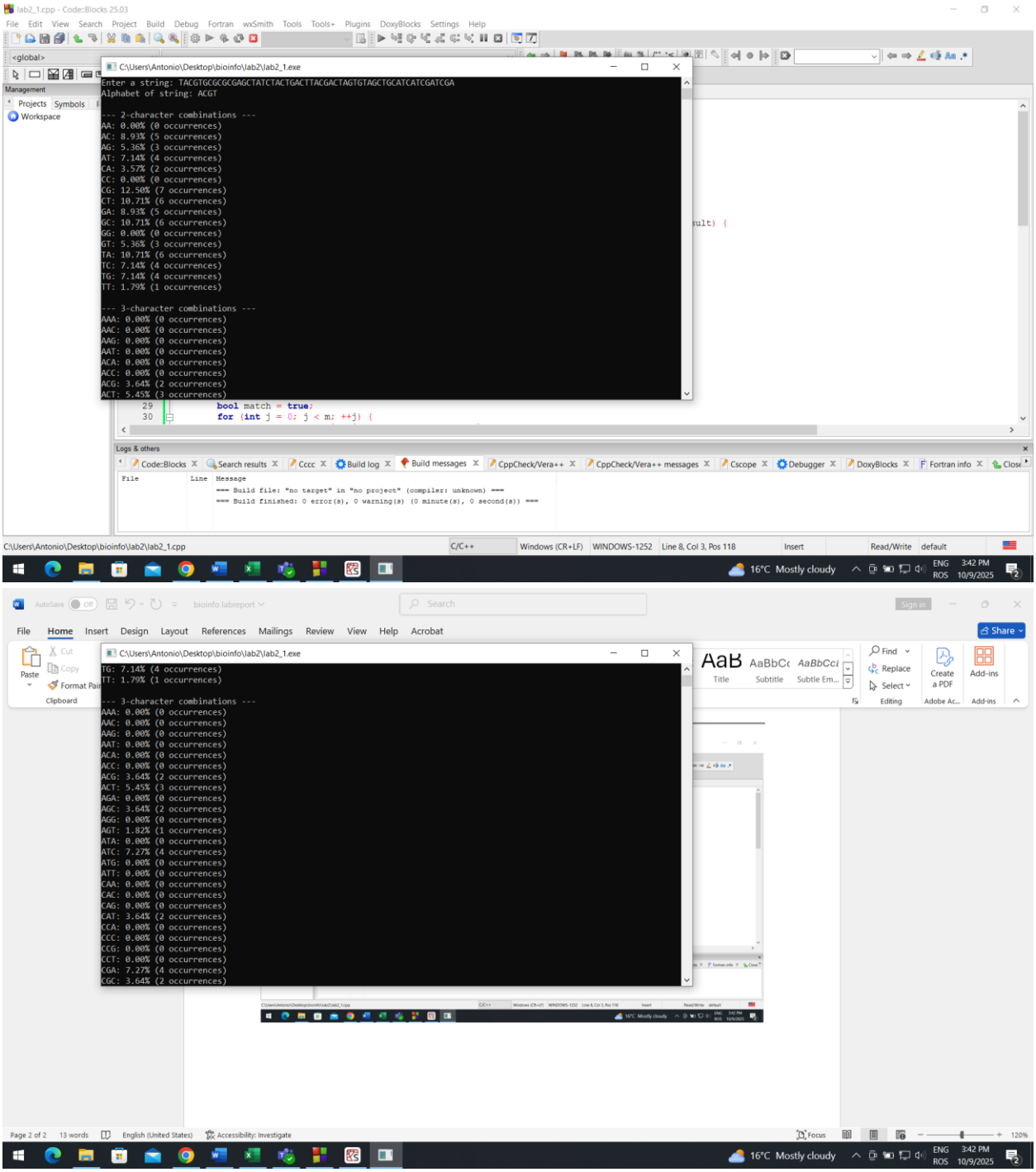
LABORATORY REPORT #2

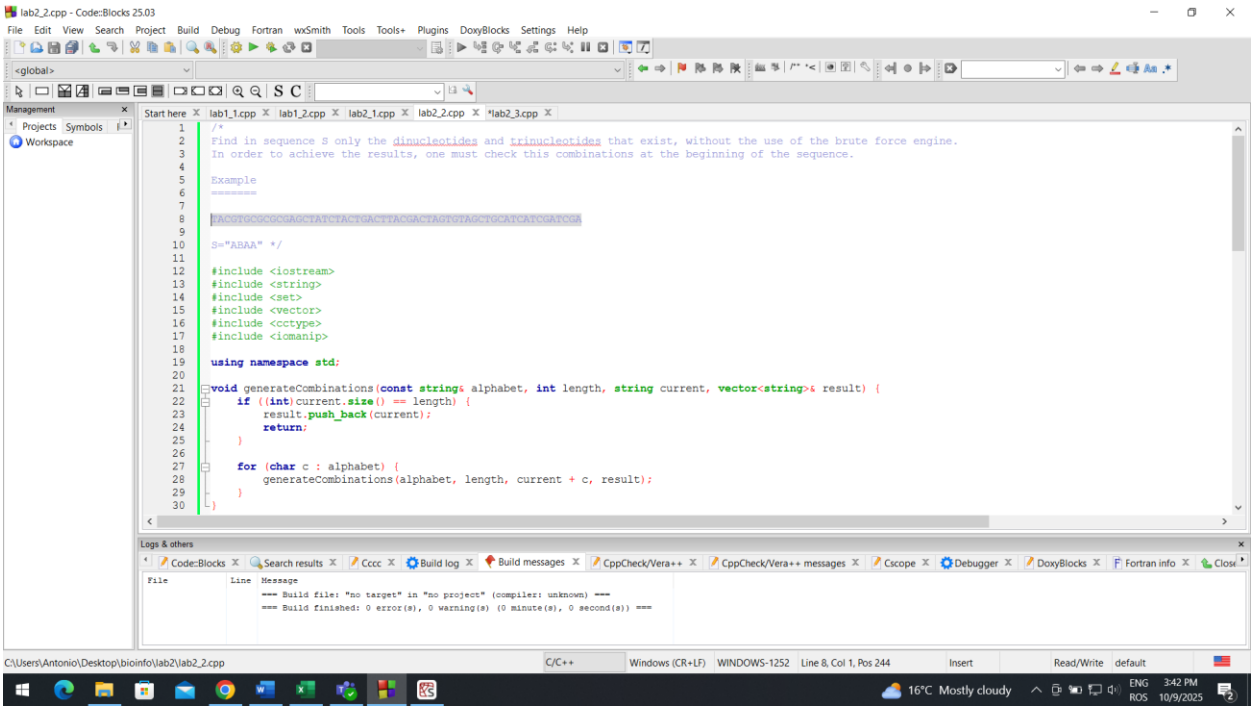
GEORGESCU Mihai-Antonio, 1242EA
Bioinformatics, 4th year 1st semester, 2025-2026

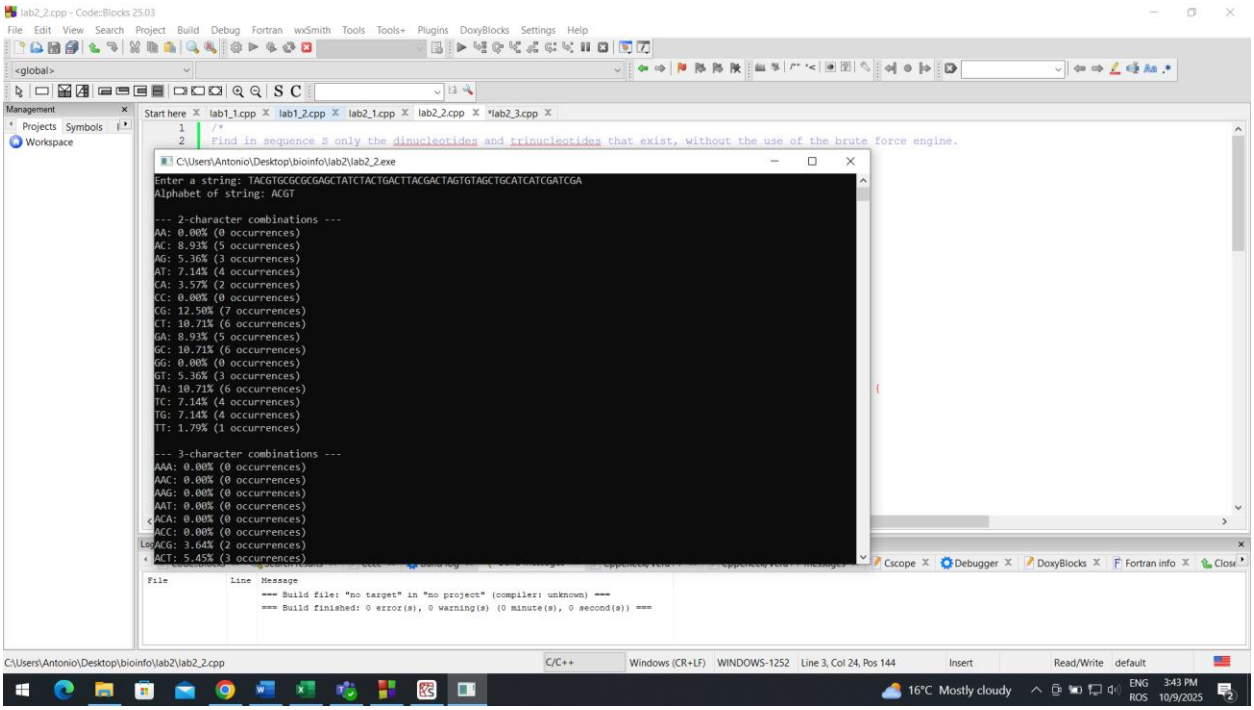
Lab2_1.cpp



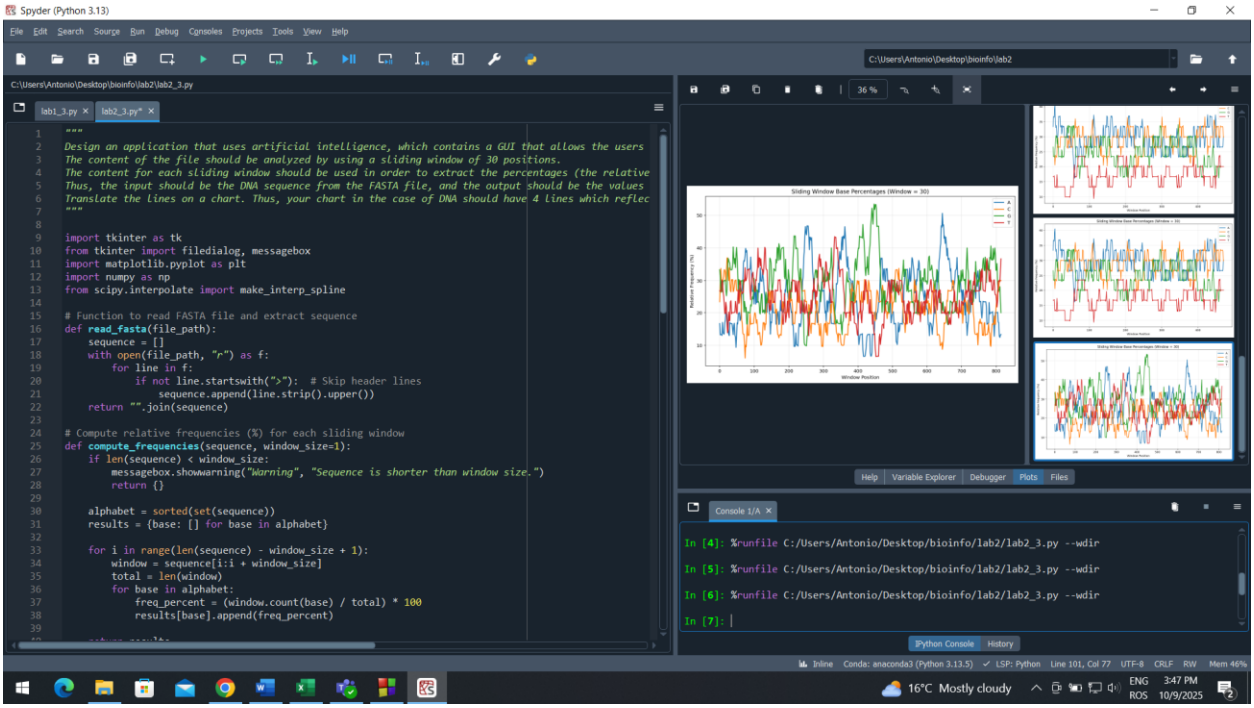

C:\Users\Antonio\Desktop\bioinfo\lab2\lab2_1.exe
Enter a string: TACGTGCGCGCGAGCTATCTACTGACTTACGACTAGTGTAGCTGCATCATCGATCGA
Alphabet of string: ACGT
--- 2-character combinations ---
AA: 0.00% (0 occurrences)
AC: 8.93% (5 occurrences)
AG: 5.36% (3 occurrences)
AT: 7.14% (4 occurrences)
CA: 3.57% (2 occurrences)
CC: 0.00% (0 occurrences)
CG: 12.50% (7 occurrences)
CT: 10.71% (6 occurrences)
GA: 8.93% (5 occurrences)
GC: 10.71% (6 occurrences)
GG: 0.00% (0 occurrences)
GT: 5.36% (3 occurrences)
TA: 10.71% (6 occurrences)
TC: 7.14% (4 occurrences)
TG: 7.14% (4 occurrences)
TT: 1.79% (1 occurrences)
--- 3-character combinations ---
AAA: 0.00% (0 occurrences)
AAC: 0.00% (0 occurrences)
AAG: 0.00% (0 occurrences)
AAT: 0.00% (0 occurrences)
ACA: 0.00% (0 occurrences)
ACC: 0.00% (0 occurrences)
ACG: 3.64% (2 occurrences)
ACT: 5.45% (3 occurrences)
AGA: 0.00% (0 occurrences)
AGC: 3.64% (2 occurrences)
AGG: 0.00% (0 occurrences)
AGT: 1.82% (1 occurrences)
ATA: 0.00% (0 occurrences)
ATC: 7.27% (4 occurrences)
ATG: 0.00% (0 occurrences)
ATT: 0.00% (0 occurrences)
CAA: 0.00% (0 occurrences)
CAC: 0.00% (0 occurrences)
CAG: 0.00% (0 occurrences)
CAT: 3.64% (2 occurrences)
CCA: 0.00% (0 occurrences)
CCC: 0.00% (0 occurrences)
CCG: 0.00% (0 occurrences)
CCT: 0.00% (0 occurrences)
CGA: 7.27% (4 occurrences)
CGC: 3.64% (2 occurrences)

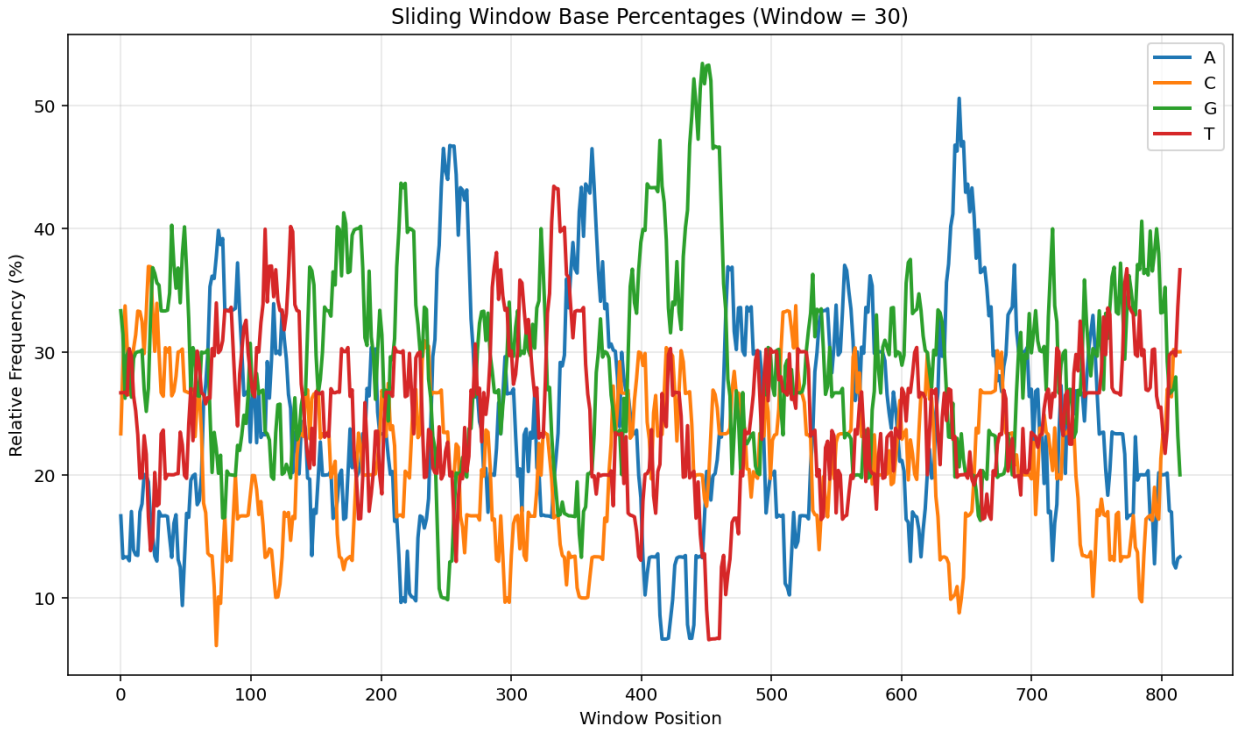
lab2_2.cpp


lab2_2.cpp - Code::Blocks 25.03
/*
Find in sequence S only the dinucleotides and trinucleotides that exist, without the use of the brute force engine.
In order to achieve the results, one must check this combinations at the beginning of the sequence.
Example
=======
TACGTGCGCGCGAGCTATCTACTGACTTACGACTAGTGTAGCTGCATCATCGATCGA
S="ABAA" */
#include <iostream>
#include <string>
#include <set>
#include <vector>
#include <cctype>
#include <iomanip>
using namespace std;
void generateCombinations(const string& alphabet, int length, string current, vector<string>& result) {
if ((int)current.size() == length) {
result.push_back(current);
return;
}
for (char c : alphabet) {
generateCombinations(alphabet, length, current + c, result);
}
}
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===


C:\Users\Antonio\Desktop\bioinfo\lab2\lab2_2.exe
Enter a string: TACGTGCGCGCGAGCTATCTACTGACTTACGACTAGTGTAGCTGCATCATCGATCGA
Alphabet of string: ACGT
--- 2-character combinations ---
AA: 0.00% (0 occurrences)
AC: 8.93% (5 occurrences)
AG: 5.36% (3 occurrences)
AT: 7.14% (4 occurrences)
CA: 3.57% (2 occurrences)
CC: 0.00% (0 occurrences)
CG: 12.50% (7 occurrences)
CT: 10.71% (6 occurrences)
GA: 8.93% (5 occurrences)
GC: 10.71% (6 occurrences)
GG: 0.00% (0 occurrences)
GT: 5.36% (3 occurrences)
TA: 10.71% (6 occurrences)
TC: 7.14% (4 occurrences)
TG: 7.14% (4 occurrences)
TT: 1.79% (1 occurrences)
--- 3-character combinations ---
AAA: 0.00% (0 occurrences)
AAC: 0.00% (0 occurrences)
AAG: 0.00% (0 occurrences)
AAT: 0.00% (0 occurrences)
ACA: 0.00% (0 occurrences)
ACC: 0.00% (0 occurrences)
ACG: 3.64% (2 occurrences)
ACT: 5.45% (3 occurrences)

lab2_3.py


Spyder (Python 3.13)
lab2_3.py
"""
Design an application that uses artificial intelligence, which contains a GUI that allows the users
The content of the file should be analyzed by using a sliding window of 30 positions.
The content for each sliding window should be used in order to extract the percentages (the relative
Thus, the input should be the DNA sequence from the FASTA file, and the output should be the values
Translate the lines on a chart. Thus, your chart in the case of DNA should have 4 lines which reflec
"""
import tkinter as tk
from tkinter import filedialog, messagebox
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import make_interp_spline
# Function to read FASTA file and extract sequence
def read_fasta(file_path):
sequence = []
with open(file_path, "r") as f:
for line in f:
if not line.startswith(">"): # Skip header lines
sequence.append(line.strip().upper())
return "".join(sequence)
# Compute relative frequencies (%) for each sliding window
def compute_frequencies(sequence, window_size=1):
if len(sequence) < window_size:
messagebox.showwarning("Warning", "Sequence is shorter than window size.")
return {}
alphabet = sorted(set(sequence))
results = {base: [] for base in alphabet}
for i in range(len(sequence) - window_size + 1):
window = sequence[i:i + window_size]
total = len(window)
for base in alphabet:
freq_percent = (window.count(base) / total) * 100
results[base].append(freq_percent)
Console 1/A
In [4]: %runfile C:/Users/Antonio/Desktop/bioinfo/lab2/lab2_3.py --wdir
In [5]: %runfile C:/Users/Antonio/Desktop/bioinfo/lab2/lab2_3.py --wdir
In [6]: %runfile C:/Users/Antonio/Desktop/bioinfo/lab2/lab2_3.py --wdir
In [7]:
16°C Mostly cloudy
ENG
ROS
3:47 PM
10/9/2025


Sliding Window Base Percentages (Window = 30)
A
C
G
T
Relative Frequency (%)
Window Position
10
20
30
40
50
0
100
200
300
400
500
600
700
800