Name : Devkumar Biswas

Class: BE(AI&DS)

Div: B

Subject : BIO(CL-II Lab)

Roll no. : BEAD21267

**Assignment No. - 1**

**Problem Statement** : DNA Sequence Analysis.

Task: Analyze a given DNA sequence and perform basic sequence manipulation, including finding motifs, calculating GC content, and identifying coding regions.

CODE:

import tkinter as tk

from tkinter import messagebox

import matplotlib.pyplot as plt

def find\_pattern(sequence, pattern):

"""

Finds occurrences of a pattern in a DNA sequence.

Returns a list of starting indices of the pattern matches.

"""

matches = []

pattern\_length = len(pattern)

sequence\_length = len(sequence)

for i in range(sequence\_length - pattern\_length + 1):

if sequence[i:i + pattern\_length] == pattern:

matches.append(i)

return matches

def calculate\_gc\_content(sequence):

"""

Calculates the GC content of a DNA sequence.

Returns the GC content as a percentage.

"""

gc\_count = 0

total\_count = 0

for nucleotide in sequence:

if nucleotide == 'G' or nucleotide == 'C':

gc\_count += 1

total\_count += 1

gc\_content = (gc\_count / total\_count) \* 100

return gc\_content

def find\_motifs(sequence, motifs):

"""

Finds motifs (subsequences) in a DNA sequence.

Returns a list of starting indices of the motif matches.

"""

matches = []

for motif in motifs:

pattern\_length = len(motif)

sequence\_length = len(sequence)

for i in range(sequence\_length - pattern\_length + 1):

if sequence[i:i + pattern\_length] == motif:

matches.append(i)

return matches

def predict\_protein\_coding\_regions(sequence):

"""

Predicts protein coding regions in a DNA sequence.

Returns a list of protein coding regions as tuples of start and end indices.

"""

regions = []

in\_coding\_region = False

sequence\_length = len(sequence)

for i in range(sequence\_length - 2):

codon = sequence[i:i + 3]

if codon == 'ATG' and not in\_coding\_region:

start\_index = i

in\_coding\_region = True

elif (codon == 'TAA' or codon == 'TAG' or codon == 'TGA') and in\_coding\_region:

end\_index = i + 2

regions.append((start\_index, end\_index))

in\_coding\_region = False

return regions

def create\_dna\_sequence\_visualization(sequence, highlights=None):

"""

Creates a visualization of a DNA sequence.

Optionally, highlights specific regions of interest.

"""

print("Highlights:", highlights)

fig, ax = plt.subplots()

ax.set\_xlim(0, len(sequence))

ax.set\_ylim(0, 1)

ax.set\_xticks(range(len(sequence)))

ax.set\_xticklabels(list(sequence), fontsize=8)

# Highlight regions of interest

if highlights:

for highlight in highlights:

if isinstance(highlight, int):

start = end = highlight

else:

start, end = highlight

ax.axvspan(start, end + 1, facecolor='yellow', alpha=0.3)

ax.set\_title("DNA Sequence Visualization")

plt.show()

def analyze\_sequence():

sequence = entry\_sequence.get()

pattern = entry\_pattern.get()

# Validate the input

if not sequence.isalpha() or set(sequence) - {'A', 'T', 'G', 'C'}:

messagebox.showerror("Error", "Invalid input! Please enter a valid DNA sequence.")

return

pattern\_matches = find\_pattern(sequence, pattern)

gc\_content = calculate\_gc\_content(sequence)

motifs = find\_motifs(sequence, ["ATG", "TAG"])

coding\_regions = predict\_protein\_coding\_regions(sequence)

result = "DNA Sequence: {}\n".format(sequence)

if len(pattern\_matches) > 0:

result += "Pattern '{}' Matches: {}\n".format(pattern, pattern\_matches)

else:

result += "Pattern '{}' not found in the sequence.\n".format(pattern)

result += "GC Content: {:.2f}\n".format(gc\_content)

if len(motifs) > 0:

result += "Motif Matches: {}\n".format(motifs)

else:

result += "No motifs found in the sequence.\n"

if len(coding\_regions) > 0:

result += "Coding Regions: {}\n".format(coding\_regions)

else:

result += "No protein coding regions found in the sequence.\n"

# Create DNA sequence visualization

highlights = pattern\_matches + coding\_regions

print("Highlights:", highlights)

create\_dna\_sequence\_visualization(sequence, highlights)

# Show the result in a message box

messagebox.showinfo("Sequence Analysis Result", result)

# Create the main window

window = tk.Tk()

window.title("DNA Sequence Analyzer")

# Create a label and entry field for the sequence input

label\_sequence = tk.Label(window, text="Enter a DNA sequence:")

label\_sequence.pack(pady=5)

entry\_sequence = tk.Entry(window)

entry\_sequence.pack()

# Create a label and entry field for the pattern input

label\_pattern = tk.Label(window, text="Enter a pattern:")

label\_pattern.pack(pady=5)

entry\_pattern = tk.Entry(window)

entry\_pattern.pack()

# Create a button to trigger the analysis

button = tk.Button(window, text="Analyze", command=analyze\_sequence)

button.pack(pady=10)

# Start the GUI event loop

window.mainloop()

dna\_sequence = "ATGGTACCCTAAATGTAGCTAGCTAAAGTCCCATG"

pattern = "TAA"

motifs = ["ATG", "TAG"]

coding\_regions = predict\_protein\_coding\_regions(dna\_sequence)

print("DNA Sequence:", dna\_sequence)

print("Pattern Matches:", find\_pattern(dna\_sequence, pattern))

print("GC Content:", calculate\_gc\_content(dna\_sequence))

print("Motif Matches:", find\_motifs(dna\_sequence, motifs))

print("Coding Regions:", coding\_regions)

OUTPUT:





