BIOTECHNOLOGY -FOCUSED DNA ANALYSIS TOOL

Purpose: This project is designed to help biotechnology students and professionals analyze DNA sequences in a simple but effective way using Python programming. It simulates some key bioinformatics tasks that are commonly done to understand DNA structure, function, and to translate genetic information into meaningful biological insights.—

Features Explained:

1. Input and Validation of DNA Sequence:- The program allows the user to enter a DNA sequence manually. - It checks if the sequence contains only valid nucleotide characters (A, T, G, C, and N for unknown). - This ensures data correctness and avoids incorrect downstream analysis.

2. Nucleotide Composition Calculation:- It calculates the percentage of each base (Adenine, Thymine, Guanine, Cytosine) in the sequence. - Knowing these percentages helps understand the chemical makeup of the DNA and informs about sequence bias or mutations. - The composition is visually represented using ASCII progress bars in the terminal for easy understanding.

3. GC Content Analysis:- GC content is the percentage of Guanines (G) and Cytosines (C) in the DNA. - This metric is biologically important as regions with high GC content tend to be more stable. - It’s widely used in gene prediction, genome characterization, and primer design.

4. Motif Finding:- This feature finds all occurrences (positions) of a specific short DNA pattern (motif) within the sequence. - Motifs can represent important biological signals like restriction enzyme recognition sites or transcription factor binding sites. - The program highlights where these motifs occur, helping users locate functional regions.

5. Transcription (DNA to RNA):- Transcription is the biological process where DNA is converted into RNA. - The tool simulates this by replacing all thymine (T) nucleotides in DNA with uracil (U), thus producing an RNA sequence. - This helps in understanding gene expression and RNA analysis.

6. Translation (RNA to Protein):- Translation converts RNA sequences into proteins by reading codons (triplets of RNA bases). - The program uses the standard genetic code to convert RNA codons into amino acids. - It stops translation at stop codons and outputs the synthesized protein sequence. This is fundamental in biotechnology for studying gene function and protein engineering.

7. Sample DNA Sequences:- To facilitate quick testing, the tool provides sample DNA sequences from different organisms: - Human mitochondrial DNA - Bacterial 16S rRNA gene segment - Plant chloroplast DNA fragment - Phage lambda fragment - Users can load these samples to see how the tool works without needing to enter their own sequences.

8. Interactive Menu-Driven Interface:- The program runs as a terminal application with clear menus to guide the user. - Users select what analysis they want to perform step-by-step. - This interactive approach makes the tool accessible and user friendly even for beginners.—

Why This Project is Unique and Valuable:--

Combines Python with Biotechnology Concepts: It applies programming skills to real biological data and problems, demonstrating interdisciplinary knowledge. Educational Value: It replicates fundamental bioinformatics analyses used in research and industry, making it a practical learning tool. Fully Functional: Instead of just a prototype, it includes many core DNA analysis features integrated into one script. Modular and Extendable: The project’s structure and clear functions allow adding more features in the future, such as graphical visualization or large sequence handling. Ready for Demonstration: The tool is self-contained, easy to run, and interact with perfect for an internship interview demo.

How It Works in Practice:

1. When started, it welcomes the user and presents main options: enter a new sequence, load a sample, or exit.

2. Once a sequence is selected, the user can choose from several analyses: composition, GC content, motif search, transcription, translation, or full summary.

3. The user receives immediate printed results with numeric values and visual aids

4. They can go back and forth to explore different features or enter new sequences conveniently.

5. Each biological concept is demonstrated with real or sample data to reinforce understanding.

**Conclusion:** "In summary, the **Biotechnology DNA Analysis Tool** is a unique and practical application that combines programming with biological analysis. It serves as an educational resource for students and professionals in the field of biotechnology. I believe this project showcases my ability to apply programming skills to real-world biological problems."

**Demo:** "Now, I would like to demonstrate the tool in action. [At this point, you can run the script and show how to input a sequence, perform analyses, and display results.]"

**Closing:** "Thank you for your attention! I'm happy to answer any questions you may have about the project."

**Running Instructions:**

1. **Ensure Python is Installed:** Make sure you have Python installed on your machine (preferably Python 3.x).
2. **Save the Code:** Copy the provided code into a file named **biotech\_dna\_analysis.py**.
3. **Run the Script:** Open your terminal or command prompt, navigate to the directory where the file is saved, and run the command:

bash

RunCopy code

1python biotech\_dna\_analysis.py

1. **Follow the Prompts:** Interact with the tool by following the on-screen prompts to input sequences or select sample data.

"""

Biotechnology-focused DNA Analysis Tool

Features:

- Input DNA sequences

- Validate sequences

- Calculate nucleotide composition

- Calculate GC content

- Find motif occurrences

- Transcription (DNA to RNA)

- Translation (RNA to Protein)

- Display statistics and simple ASCII visualizations

- Dummy sample sequences for testing

Run this script in the terminal. Follow interactive menu prompts.

"""

import re

# Genetic code dictionary for translation

GENETIC\_CODE = {

'UUU':'F', 'UUC':'F', 'UUA':'L', 'UUG':'L',

'UCU':'S', 'UCC':'S', 'UCA':'S', 'UCG':'S',

'UAU':'Y', 'UAC':'Y', 'UAA':'\*', 'UAG':'\*',

'UGU':'C', 'UGC':'C', 'UGA':'\*', 'UGG':'W',

'CUU':'L', 'CUC':'L', 'CUA':'L', 'CUG':'L',

'CCU':'P', 'CCC':'P', 'CCA':'P', 'CCG':'P',

'CAU':'H', 'CAC':'H', 'CAA':'Q', 'CAG':'Q',

'CGU':'R', 'CGC':'R', 'CGA':'R', 'CGG':'R',

'AUU':'I', 'AUC':'I', 'AUA':'I', 'AUG':'M',

'ACU':'T', 'ACC':'T', 'ACA':'T', 'ACG':'T',

'AAU':'N', 'AAC':'N', 'AAA':'K', 'AAG':'K',

'AGU':'S', 'AGC':'S', 'AGA':'R', 'AGG':'R',

'GUU':'V', 'GUC':'V', 'GUA':'V', 'GUG':'V',

'GCU':'A', 'GCC':'A', 'GCA':'A', 'GCG':'A',

'GAU':'D', 'GAC':'D', 'GAA':'E', 'GAG':'E',

'GGU':'G', 'GGC':'G', 'GGA':'G', 'GGG':'G'

}

def validate\_dna(seq):

"""Return True if the sequence contains only DNA nucleotides."""

return re.fullmatch(r'[ATGCNatgcn]+', seq) is not None

def dna\_to\_rna(seq):

"""Transcribe DNA sequence to RNA."""

return seq.upper().replace('T', 'U')

def rna\_to\_protein(rna\_seq):

"""Translate RNA sequence into protein (using standard genetic code). Stop at stop codon."""

protein = []

seq = rna\_seq.upper()

for i in range(0, len(seq)-2, 3):

codon = seq[i:i+3]

aa = GENETIC\_CODE.get(codon, '?')

if aa == '\*': # stop codon

break

protein.append(aa)

return ''.join(protein)

def nucleotide\_composition(seq):

"""Calculate nucleotide composition percentages."""

seq = seq.upper()

length = len(seq)

comp = {

'A': seq.count('A') / length \* 100,

'T': seq.count('T') / length \* 100,

'G': seq.count('G') / length \* 100,

'C': seq.count('C') / length \* 100,

'N': seq.count('N') / length \* 100

}

return comp

def gc\_content(seq):

"""Calculate GC content percentage."""

seq = seq.upper()

g = seq.count('G')

c = seq.count('C')

length = len(seq)

return (g + c) / length \* 100

def find\_motif(seq, motif):

"""Find all start indices of motif in sequence."""

seq = seq.upper()

motif = motif.upper()

indices = [m.start() + 1 for m in re.finditer(f'(?={motif})', seq)]

return indices

def print\_progress\_bar(prefix, percent, size=30):

"""Print a simple ASCII progress bar."""

filled = int(size \* percent / 100)

bar = '█' \* filled + '-' \* (size - filled)

print(f"{prefix}: |{bar}| {percent:.2f}%")

def display\_composition(comp):

print("Nucleotide composition:")

for base, perc in comp.items():

print(f" {base}: {perc:.2f}%")

print()

# Simple bar visualization

for base in ['A', 'T', 'G', 'C']:

print\_progress\_bar(base, comp.get(base, 0))

def main\_menu():

print("="\*60)

print("Welcome to the Biotechnology DNA Analysis Tool")

print("="\*60)

print("Choose an option:")

print("1. Input and analyze new DNA sequence")

print("2. Load sample DNA sequences")

print("3. Exit")

choice = input("Enter choice (1-3): ")

return choice.strip()

def analysis\_menu():

print("\nDNA Analysis Options:")

print("1. Show nucleotide composition")

print("2. Calculate GC content")

print("3. Find motif in sequence")

print("4. Transcribe DNA to RNA")

print("5. Translate RNA to Protein")

print("6. Show summary")

print("7. Back to main menu")

choice = input("Choose analysis to perform (1-7): ")

return choice.strip()

def sample\_sequences():

# Some dummy biologically relevant samples

samples = {

'Human mitochondrial DNA fragment':

'ATGCCACTGACCGGTAGCTAGGCTACGATCGGTAGCTAGCTAGGCTAGCTAGCTGATCG',

'Bacterial 16S rRNA gene fragment':

'AGAGTTTGATCCTGGCTCAGGATGAACGCTGGCGGCGTGCCTAATACATGCAAGTCGA',

'Plant chloroplast DNA fragment':

'ATGCGCAACCGGAGCTCCTATAGAGGCTTGTACTGATGTTGCGGGTACTAG',

'Phage lambda DNA fragment':

'GGGCGGCGACCTTAACCTGACGTTAGGCTAAGGCGTAGTAACTGAA'

}

return samples

def run():

current\_seq = None

while True:

choice = main\_menu()

if choice == '1':

seq = input("\nEnter DNA sequence (A,T,G,C,N only):\n").strip()

if not validate\_dna(seq):

print("Invalid DNA sequence! Only A, T, G, C, N allowed.")

continue

current\_seq = seq.upper()

print("Sequence accepted.")

while True:

if current\_seq is None: # Safety check

break

analysis\_choice = analysis\_menu()

if analysis\_choice == '1':

comp = nucleotide\_composition(current\_seq)

display\_composition(comp)

elif analysis\_choice == '2':

gc = gc\_content(current\_seq)

print(f"GC content: {gc:.2f}%\n")

elif analysis\_choice == '3':

motif = input("Enter motif sequence to find (e.g., ATG): ").strip()

if not validate\_dna(motif):

print("Invalid motif sequence!")

continue

indices = find\_motif(current\_seq, motif)

if indices:

print(f"Motif '{motif}' found at positions: {indices}\n")

else:

print(f"Motif '{motif}' not found in sequence.\n")

elif analysis\_choice == '4':

rna\_seq = dna\_to\_rna(current\_seq)

print(f"Transcribed RNA sequence:\n{rna\_seq}\n")

elif analysis\_choice == '5':

rna\_seq = dna\_to\_rna(current\_seq)

protein = rna\_to\_protein(rna\_seq)

print(f"Translated Protein Sequence:\n{protein}\n")

elif analysis\_choice == '6':

comp = nucleotide\_composition(current\_seq)

gc = gc\_content(current\_seq)

print("\nSequence Summary:")

print(f"Length: {len(current\_seq)}")

display\_composition(comp)

print(f"GC content: {gc:.2f}%")

rna\_seq = dna\_to\_rna(current\_seq)

protein = rna\_to\_protein(rna\_seq)

print(f"Protein translation (first ORF): {protein}\n")

elif analysis\_choice == '7':

break

else:

print("Invalid option, please choose 1-7.")

elif choice == '2':

samples = sample\_sequences()

print("\nSample DNA Sequences:")

for i, (desc, seq) in enumerate(samples.items(), 1):

print(f" {i}. {desc} ({len(seq)} bp)")

selected = input("Select a sample to load (1-4): ")

try:

selected = int(selected)

if 1 <= selected <= len(samples):

current\_seq = list(samples.values())[selected -1]

print(f"Loaded sample: {list(samples.keys())[selected -1]}")

print(f"Sequence: {current\_seq}\n")

# Enter analysis menu directly

while True:

analysis\_choice = analysis\_menu()

if analysis\_choice == '1':

comp = nucleotide\_composition(current\_seq)

display\_composition(comp)

elif analysis\_choice == '2':

gc = gc\_content(current\_seq)

print(f"GC content: {gc:.2f}%\n")

elif analysis\_choice == '3':

motif = input("Enter motif sequence to find (e.g., ATG): ").strip()

if not validate\_dna(motif):

print("Invalid motif sequence!")

continue

indices = find\_motif(current\_seq, motif)

if indices:

print(f"Motif '{motif}' found at positions: {indices}\n")

else:

print(f"Motif '{motif}' not found in sequence.\n")

elif analysis\_choice == '4':

rna\_seq = dna\_to\_rna(current\_seq)

print(f"Transcribed RNA sequence:\n{rna\_seq}\n")

elif analysis\_choice == '5':

rna\_seq = dna\_to\_rna(current\_seq)

protein = rna\_to\_protein(rna\_seq)

print(f"Translated Protein Sequence:\n{protein}\n")

elif analysis\_choice == '6':

comp = nucleotide\_composition(current\_seq)

gc = gc\_content(current\_seq)

print("\nSequence Summary:")

print(f"Length: {len(current\_seq)}")

display\_composition(comp)

print(f"GC content: {gc:.2f}%")

rna\_seq = dna\_to\_rna(current\_seq)

protein = rna\_to\_protein(rna\_seq)

print(f"Protein translation (first ORF): {protein}\n")

elif analysis\_choice == '7':

break

else:

print("Invalid option, please choose 1-7.")

else:

print("Invalid selection.")

except ValueError:

print("Please enter a number.")

elif choice == '3':

print("Exiting. Thank you for using the tool!")

break

else:

print("Invalid option! Enter 1, 2, or 3.")

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

run()