Lab sheet1: Information retrieval and sequence analysis

Q1

COX-2 (prostaglandin H2 synthase-2 (PTGS2)) gene

COX-2 has been thoroughly studied because of its role in prostaglandin synthesis. Prostaglandins have a wide range of roles in our body from aiding in digestion to propagating pain and inflammation.

Aspirin is a general inhibitor of prostaglandin synthesis and therefore, helps reduce pain.

However, aspirin also inhibits the synthesis of prostaglandins that aid in digestion. Therefore, aspirin is a poor choice for pain and inflammation management for those with ulcers or other digestion problems.

Recent advances in targeting specific prostaglandin-synthesizing enzymes have lead to the development of Celebrex, which is marketed as an arthritis therapy. Celebrex is a potent and specific inhibitor of COX-2. Celebrex is considered specific because it doesn't inhibit COX-1, which is involved in synthesizing prostaglandins that aid in digestion.

This is a remarkable accomplishment given the great similarity between COX-1 and COX-2.

This achievement has paved the way for developing new therapies that bind more specifically to their target and therefore have fewer side effects. Understanding the enzyme structures of COX-1 and COX-2 helped researchers develop a drug that would only bind and inhibit COX-2. Many of the types of information and tools used by researchers for these types of studies are freely available on the web .

GenBank, SwissProt, Sequence Manipulation suite are some of the websites.

i. Access the entries for Human PTGS1 and PTGS2 in the "Gene" database at the NCBI (https://www.ncbi.nlm.nih.gov/) Website.

   a. PTGS1 and PTGS2 are isozymes. Isozymes catalyze the same reaction but are separate genes. What types of reactions do PTGS enzymes catalyze? Also, what pathway are these enzymes a part of?
      - Catalyze the conversion of arachidonate to prostaglandin
      - PTGS1 is a part of the arachidonic acid metabolic pathway

   b. How is the expression of PTGS1 and PTGS2 different?

PTGS1	PTSG2
Biased expression in skin (RPKM 36.8), esophagus (RPKM 21.4) and 12 other tissues	Biased expression in bone marrow (RPKM 59.3), urinary bladder (RPKM 41.1) and 11 other tissues

   c. Which isozyme (PTGS1 or PTGS2) is required to inhibit inflammation?
      - PTGS2

   d. The drug Celebrex selectively inhibits PTGS2 while aspirin and other NSAID's inhibit both PTGS1 and PTGS2 in the same way. Why do you think researchers wanted to discover a selective inhibitor to PTGS2?

- To discover a selective inhibitor for PTGS2 to avoid inhibiting PTGS1, which is involved in synthesizing prostaglandins for digestion
- Reduces side effects

e. Describe how studying 3-D structures of PTGS1 and PTGS2 could help researchers design a drug that binds to PTGS1, but not to PTGS2.
   - Understanding structural nuances, such as flexibility and molecular interactions, enables the development of drugs with high specificity, reducing the risk of off target effects

ii. Considering the Homo sapiens PTGS2 gene entry in NCBI gene https://www.ncbi.nlm.nih.gov/gene/ database,

a. What is the gene name?
   - Prostaglandin-endoperoxide synthase 2

b. What is the GeneID number?
   - 5743

c. Where in the human genome is this gene located?
   - The gene PTGS2 is located at complement (186,671,791..186,680,423) on chromosome NC_000001.11.

d. What is the RefSeq accession number for the mRNA sequence of Homo sapiens prostaglandin-endoperoxide synthase 2?
   - NM_000963.4

e. Download the prostaglandin-endoperoxide synthase 2 Reference mRNA sequence in "FASTA" format.

f. What is the RefSeq accession number for the Homo sapiens PTGS2 protein sequence? Download the sequence in "FASTA" format.
   - NP_000954.1

iii. Search for the UniProt entry for PTGS2 in Expasy https://www.expasy.org/website.

a. What are the alternate names for this protein.
   - Prostaglandin G/H synthase 2

b. What types of drugs target this protein?
   - Nonsteroidal anti-inflammatory drugs (NSAIDs) including aspirin and ibuprofen

c. What amino acid is acetylated by aspirin (amino acid type)?
   - Serine

iv. Translate the mRNA sequence of PTGS2 into Protein. Use "Translate " tool in ExPASy. Explain the output.
   - Every DNA sequence can be read in six possible frames - three in the forward direction (5' to 3') and three in the reverse direction (3' to 5'). I have chosen the longest amino acid sequence which is in 5'3' Frame 2. Below are the equivalent fasta formats.

Fasta format

```
> VIRT-2302:5'3' Frame 2, start_pos=44
MLARALLLCAVLALSHTANPCCSHPCQNRGVCMSVGFDQYKCDCTRTGFY
GENCSTPEFLTRIKLFLKPTPNTVHYILTHFKGFWNVVNNIPFLRNAIMS
YVLTSRSHLIDSPPTYNADYGYKSWEAFSNLSYYTRALPPVPDDCPTPLG
VKGKKQLPDSNEIVEKLLLRRKFIPDPQGSNMMFAFFAQHFTHQFFKTDH
KRGPAFTNGLGHGVDLNHIYGETLARQRKLRLFKDGKMKYQIIDGEMYPP
TVKDTQAEMIYPPQVPEHLRFAVGQEVFGLVPGLMMYATIWLREHNRVCD
VLKQEHPEWGDEQLFQTSRLILIGETIKIVIEDYVQHLSGYHFKLKFDPE
LLFNKQFQYQNRIAAEFNTLYHWHPLLPDTFQIHDQKYNYQQFIYNNSIL
LEHGITQFVESFTRQIAGRVAGGRNVPPAVQKVSQASIDQSRQMKYQSFN
EYRKRFMLKPYESFEELTGEKEMSAELEALYGDIDAVELYPALLVEKPRP
DAIFGETMVEVGAPFSLKGLMGNVICSPAYWKPSTFGGEVGFQIINTASI
QSLICNNVKGCPFTSFSVPDPELIKTVTINASSSRSGLDDINPTVLLKER
STEL
```

Readings:

http://www.aspree.org/AUS/aspree-content/aspirin/how-aspirin-works.aspx

Q2. Python Exercises

1. Below shows some files with embedded sample names:

```
lane1_NewCode_L001_R1.fastq.gz
lane1_NoIndex_L001_R1.fastq.gz
lane1_NoIndex_L001_R2.fastq.gz
pipeline_processing_output.log
lane7027_ACTGAT_JH25_L001_R1.fastq.gz
lane7027_ACTTGA_E30_1_2_Hap4_24h_L001_R1.fastq.gz
lane7027_AGTTCC_JH14_L001_R1.fastq.gz
lane7027_CGGAAT_JH37_L001_R1.fastq.gz
lane7027_GCCAAT_E30_1_2l_Hap4_log_L001_R1.fastq.gz
lane7127_GGCTAC_E30_1_4_Hap4_48h_L001_R1.fastq.gz
```

Write a Python code to extract the sample name from these files ignoring any files which do not match the format given below.

The format is:

1. Written lane number
2. Barcode
3. Sample name
4. Numeric lane number (starting with L)
5. Read number (R1/2/3/4)
6. File extension

Eg. `Lane8127_GCCAAT_S30_1_2l_Hap4_log_L001_R1.fastq.gz` the sample name would be, `S30_1_2l_Hap4_log`

```python
import re


def extract_sample_name(file_name):
    pattern = r'^lane\d+_([A-Z\d]+)_([A-Za-z\d_]+)_L\d+_([R]\d+)\.fastq\.gz$'
    match = re.match(pattern, file_name)

    if match:
        return match.group(2)


file_names = [
    "lane1_NewCode_L001_R1.fastq.gz",
    "lane1_NoIndex_L001_R1.fastq.gz",
    "lane1_NoIndex_L001_R2.fastq.gz",
    "pipeline_processing_output.log",
    "lane7027_ACTGAT_JH25_L001_R1.fastq.gz",
    "lane7027_ACTTGA_E30_1_2_Hap4_24h_L001_R1.fastq.gz",
    "lane7027_AGTTCC_JH14_L001_R1.fastq.gz",
    "lane7027_CGGAAT_JH37_L001_R1.fastq.gz",
    "lane7027_GCCAAT_E30_1_2l_Hap4_log_L001_R1.fastq.gz",
    "lane7127_GGCTAC_E30_1_4_Hap4_48h_L001_R1.fastq.gz",
    "lane8127_GCCAAT_S30_1_2l_Hap4_log_L001_R1.fastq.gz"
]

for file_name in file_names:
    sample_name = extract_sample_name(file_name)
    if sample_name is not None:
        print(f"File: {file_name}, Sample Name: {sample_name}")
```

```
Run: Q2
C:\Users\maheshl\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\maheshl\PycharmProjects\pythonProject1\Uni\CB\Q2.py
File: lane7027_ACTGAT_JH25_L001_R1.fastq.gz, Sample Name: JH25
File: lane7027_ACTTGA_E30_1_2_Hap4_24h_L001_R1.fastq.gz, Sample Name: E30_1_2_Hap4_24h
File: lane7027_AGTTCC_JH14_L001_R1.fastq.gz, Sample Name: JH14
File: lane7027_CGGAAT_JH37_L001_R1.fastq.gz, Sample Name: JH37
File: lane7027_GCCAAT_E30_1_2l_Hap4_log_L001_R1.fastq.gz, Sample Name: E30_1_2l_Hap4_log
File: lane7127_GGCTAC_E30_1_4_Hap4_48h_L001_R1.fastq.gz, Sample Name: E30_1_4_Hap4_48h
File: lane8127_GCCAAT_S30_1_2l_Hap4_log_L001_R1.fastq.gz, Sample Name: S30_1_2l_Hap4_log

Process finished with exit code 0
```

2. Create a FASTA file by obtaining 10 Dengue 1- Envelop gene DNA sequences from NCBI. Write a Python-program that reads the FASTA file, cleans up the header line to have only Accession number & gene-name and print headers and sequences to standard output as multi-FASTA-file again.

```python
import re

from Bio import Entrez, SeqIO


def fetch_dengue_sequences():
    Entrez.email = "maheshlakshan7óó@gmail.com"
    handle = Entrez.esearch(db="nucleotide", term="Dengue 1 Envelope", retmax=10)
    record = Entrez.read(handle)
    ids = record["IdList"]

    dengue_sequences = []
    for record_id in ids:
        handle = Entrez.efetch(db="nucleotide", id=record_id, rettype="gb", retmode="text")
        seq_record = SeqIO.read(handle, "genbank")
        dengue_sequences.append(seq_record)

    return dengue_sequences


def write_fasta_file(sequences, output_file="dengue_sequences.fasta"):
    with open(output_file, "w") as output_handle:
        SeqIO.write(sequences, output_handle, "fasta")


def clean_up_header(header):
    pattern = r'(\S+).*?\((\w+)\) gene'

    match = re.search(pattern, header)

    if match:
        accession_number = match.group(1)
        gene_name = match.group(2)
        cleaned_header = f">{accession_number}_{gene_name}"
        return cleaned_header
    else:
        return ">Unknown"


def print_multi_fasta(file_path):
    sequences = list(SeqIO.parse(file_path, "fasta"))

    for seq_record in sequences:
        cleaned_header = clean_up_header(seq_record.description)
        print(cleaned_header)
        print(seq_record.seq)


if __name__ == "__main__":
    dengue_sequences = fetch_dengue_sequences()
    write_fasta_file(dengue_sequences)
    print_multi_fasta("dengue_sequences.fasta")
```

3. Write a Python program to search the DNA Sequence for the presence of one of the following Transcription Factor Binding Sites(TFBS) with ambiguity codes. Search for all the positions in the sequence where TFBS is located.

Code	Represents
A	Adenine
G	Guanine
C	Cytosine
T	Thymine
Y	Pyrimidine (C or T)
R	Purine (A or G)
W	weak (A or T)
S	strong (G or C)
K	keto (T or G)
M	amino (C or A)
D	A, G, T (not C)
V	A, C, G (not T)
H	A, C, T (not G)
B	C, G, T (not A)

Transcription Factor	**Consensus Sequence**
RUNX1	BHTGTGGTYW
TGIF1	WGACAGB
IKZF1	BTGGGARD

The sequence is shown below.

```
>search_seq
GACACCTCAGTACTAGGATGNNNNNNTATCAGCCTGAACTAGCAGGCCTGGTTCCAAATT
TTTTTATCAACACTCGTAGGGGGATTATCCTAGAGGGGGTCTGGGATTTCTTTGACATCA
GAGTATTTTTGCCTTGCTCCTTCACAATTTGGGAACAAATAATTTAGTGGTTATTAACCC
TGGCTACGCACTGGAAACTTTAAAAATAATGCTGGTATGAAATTTACACAGAGTATCGTG
AAAATTTTCACTGAGTACCATGTGGTTATACATTGGATAAGGCTCCAGGAAGCAGCTACT
GGAAGACAGCCATGCCAAGAGTGGTTAGTGGTTGGAATTTTGGCAAGTCAGTTTTAGTCT
GCCTTATCAAATACATGGGCATACAGATAAATCCTTAGATGGCTCTCCTACTTACTGAAA
CATTTTCTATCTATCTATCTATCTATCTATCTATTTGGGAAGCTATCTATCTATCTATCA
TTTATTTAAGGTAGTCTCTATCTGCCTCTGTCTCTGTCTGTCTCTGTGTCTCTGTGTCTG
TCTGCTCTCTCTCTCTCTCTGTGGGAATCTCTCTCTGTGTGTGTGTGTGTGTATGTGTGTGT
GTGTGTGTGTGGTGTGCATGAACATGAGTAAAATCCATAAGGAAACTTTCAGAGTTGGTC
CTCTCCTTATATCAAATGGATCCAGGAATTAAACTCAGGTTCAATTCTTGGTGCCTTTAC
TAGTTGAGCCATCTCACTGGCTCTTCATCATCTTTAGAATAAACTCACTTTATTACACAC
```

ACACACACACACACAACCTGGGAGTACACACACACACACAACCAAAGCCCCAACGGAAAA

CTACAATATTATAATGAATACACAGGTTCTCAACATAGTCTCTGCCACGCTTGCAGACAA

AGATGAGTAGAAGTAGAAAGAACCAGGGAAACGTGGAGCAAGTCAGAAGGAATAACAGTC

AGAAGGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAGTAACAGTCAGAAGGAATAGC

AGTCAGAAGGAATAACAGTCAGAAGACAGCACAGTCAGAAGGAATAACAGTCAGAAGGAA

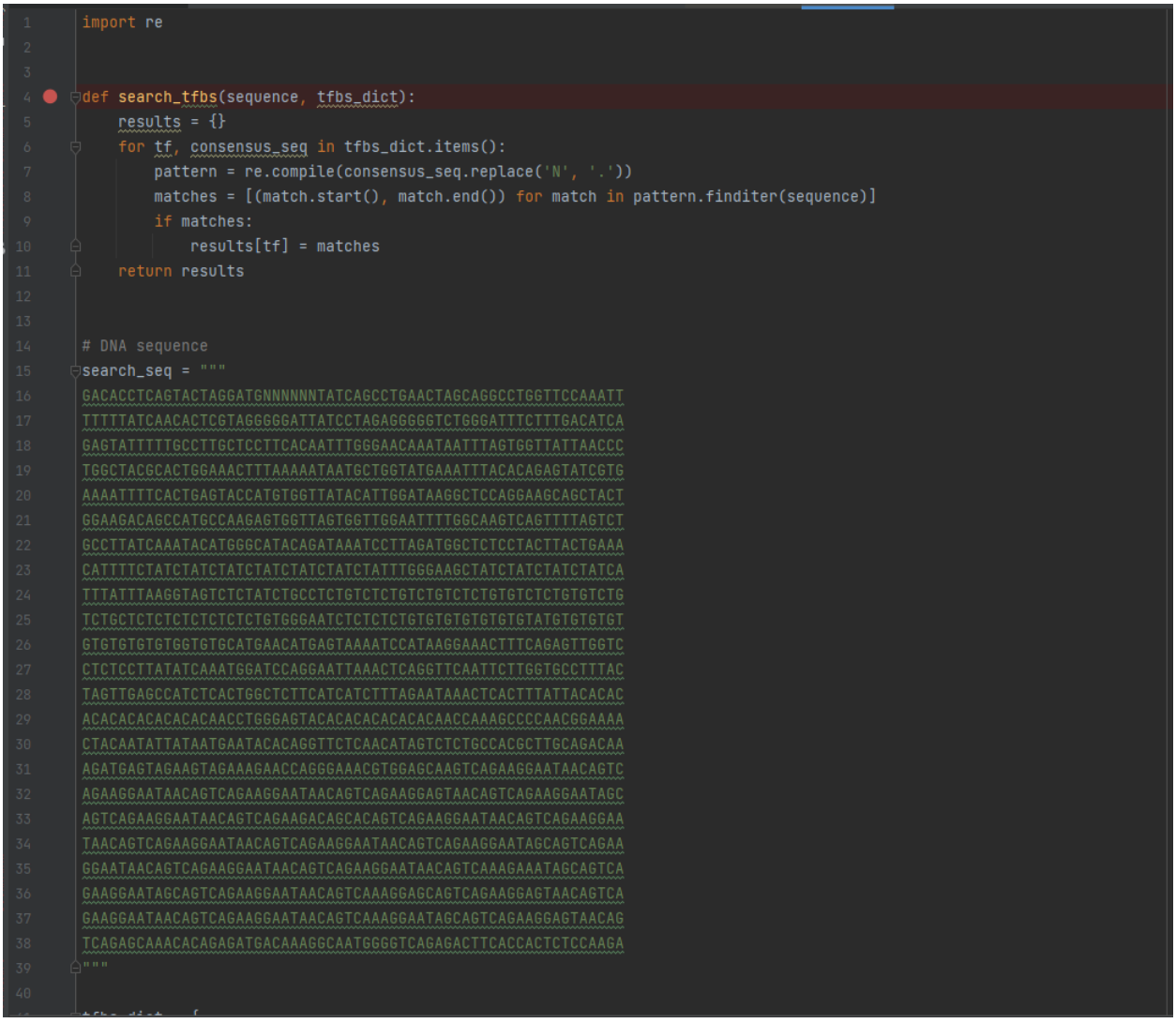
TAACAGTCAGAAGGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAATAGCAGTCAGAA

GGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAATAACAGTCAAAGAAATAGCAGTCA

GAAGGAATAGCAGTCAGAAGGAATAACAGTCAAAGGAGCAGTCAGAAGGAGTAACAGTCA

GAAGGAATAACAGTCAGAAGGAATAACAGTCAAAGGAATAGCAGTCAGAAGGAGTAACAG

TCAGAGCAAACACAGAGATGACAAAGGCAATGGGGTCAGAGACTTCACCACTCTCCAAGA



```python
import re


def search_tfbs(sequence, tfbs_dict):
    results = {}
    for tf, consensus_seq in tfbs_dict.items():
        pattern = re.compile(consensus_seq.replace('N', '.'))
        matches = [(match.start(), match.end()) for match in pattern.finditer(sequence)]
        if matches:
            results[tf] = matches
    return results


# DNA sequence
search_seq = """
GACACCTCAGTACTAGGATGNNNNNNTATCAGCCTGAACTAGCAGGCCTGGTTCCAAATT
TTTTTATCAACACTCGTAGGGGGATTATCCTAGAGGGGGTCTGGGATTTCTTTGACATCA
GAGTATTTTTGCCTTGCTCCTTCACAATTTGGGAACAAATAATTTAGTGGTTATTAACCC
TGGCTACGCACTGGAAACTTTAAAAATAATGCTGGTATGAAATTTACACAGAGTATCGTG
AAAATTTTCACTGAGTACCATGTGGTTATACATTGGATAAGGCTCCAGGAAGCAGCTACT
GGAAGACAGCCATGCCAAGAGTGGTTAGTGGTTGGAATTTTGGCAAGTCAGTTTTAGTCT
GCCTTATCAAATACATGGGCATACAGATAAATCCTTAGATGGCTCTCCTACTTACTGAAA
CATTTTCTATCTATCTATCTATCTATCTATCTATTTGGGAAGCTATCTATCTATCTATCA
TTTATTTAAGGTAGTCTCTATCTGCCTCTGTCTCTGTCTGTCTCTGTGTCTCTGTGTCTG
TCTGCTCTCTCTCTCTCTCTGTGGGAATCTCTCTCTGTGTGTGTGTGTGTATGTGTGTGT
GTGTGTGTGTGGTGTGCATGAACATGAGTAAAATCCATAAGGAAACTTTCAGAGTTGGTC
CTCTCCTTATATCAAATGGATCCAGGAATTAAACTCAGGTTCAATTCTTGGTGCCTTTAC
TAGTTGAGCCATCTCACTGGCTCTTCATCATCTTTAGAATAAACTCACTTTATTACACAC
ACACACACACACACAACCTGGGAGTACACACACACACACACAACCAAAGCCCCAACGGAAAA
CTACAATATTATAATGAATACACAGGTTCTCAACATAGTCTCTGCCACGCTTGCAGACAA
AGATGAGTAGAAGTAGAAAGAACCAGGGAAACGTGGAGCAAGTCAGAAGGAATAACAGTC
AGAAGGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAGTAACAGTCAGAAGGAATAGC
AGTCAGAAGGAATAACAGTCAGAAGACAGCACAGTCAGAAGGAATAACAGTCAGAAGGAA
TAACAGTCAGAAGGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAATAGCAGTCAGAA
GGAATAACAGTCAGAAGGAATAACAGTCAGAAGGAATAACAGTCAAAGAAATAGCAGTCA
GAAGGAATAGCAGTCAGAAGGAATAACAGTCAAAGGAGCAGTCAGAAGGAGTAACAGTCA
GAAGGAATAACAGTCAGAAGGAATAACAGTCAAAGGAATAGCAGTCAGAAGGAGTAACAG
TCAGAGCAAACACAGAGATGACAAAGGCAATGGGGTCAGAGACTTCACCACTCTCCAAGA
"""

tfbs_dict = {
```

```python
tfbs_dict = {
    'RUNX1': 'BHTGTGGTYW',
    'TGIF1': 'WGACAGB',
    'IKZF1': 'BTGGGARD'
}


def replace_ambiguous_bases(consensus_sequence):
    modified_sequence = consensus_sequence\
        .replace('Y', '[CT]')\
        .replace('R', '[AG]')\
        .replace('W', '[AT]')\
        .replace('S', '[CG]')\
        .replace('K', '[TG]')\
        .replace('M', '[AC]')\
        .replace('D', '[AGT]')\
        .replace('V', '[ACG]')\
        .replace('H', '[ACT]')\
        .replace('B', '[CGT]')
    return modified_sequence


# Replace ambiguity codes with '[ACGT]' in each TFBS consensus sequence
for tf, consensus_seq in tfbs_dict.items():
    modified_sequence = replace_ambiguous_bases(consensus_seq)
    tfbs_dict[tf] = modified_sequence

results = search_tfbs(search_seq, tfbs_dict)

for tf, positions in results.items():
    print(f"{tf} found at positions: {positions}")
```

```
C:\Users\maheshl\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\maheshl\PycharmProjects\pythonProject1\Uni\CB\Q4.py
TF: RUNX1 with Concensus Seq: '[CGT][ACT]TGTGGT[CT][AT]' found at positions: [(263, 273)]
Sequence: CATGTGGTTA
TF: TGIF1 with Concensus Seq: '[AT]GACAG[CGT]' found at positions: [(309, 316), (1061, 1068)]
Sequence: AGACAGC
Sequence: AGACAGC
TF: IKZF1 with Concensus Seq: '[CGT]TGGGA[AG][AGT]' found at positions: [(462, 470), (570, 578), (811, 819)]
Sequence: TTGGGAAG
Sequence: GTGGGAAT
Sequence: CTGGGAGT

Process finished with exit code 0
```

Q3 – Biopython

Biopython Tutorial and Cookbook https://biopython.org/DIST/docs/tutorial/Tutorial.html#sec2

1. Write a Biopython program that asks the user to input a DNA-sequence and then translates the sequence to protein sequence.

```python
from Bio.Seq import Seq

# Get user input for DNA sequence
dna_sequence = input("Enter the DNA sequence: ")

dna_seq = Seq(dna_sequence)

protein_seq = dna_seq.translate()

print("Translated Protein Sequence:", protein_seq)
```

```
C:\Users\maheshl\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\maheshl\PycharmProjects\pythonProject1\Uni\CB\T1.py
Enter the DNA sequence: GTGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
Translated Protein Sequence: VAIVMGR*KGAR*

Process finished with exit code 0
```

2. Write a Biopython program that will find all articles related to Alzheimer's in PubMed. Print the total number of articles available and the authors.

```python
from Bio import Entrez

Entrez.email = "maheshlakshan766@gmail.com"

handle = Entrez.esearch(db="pubmed", term="Alzheimer's")
record = Entrez.read(handle)
pubmed_ids = record["IdList"]

if pubmed_ids:
    handle = Entrez.efetch(db="pubmed", id=pubmed_ids, rettype="xml")
    records = Entrez.read(handle)

    print("Total number of articles:", record['Count'])

    for record in records["PubmedArticle"]:
        authors = ", ".join(author["LastName"] + " " + author["Initials"] for author in record["MedlineCitation"]["Article"]["AuthorList"])
        print(f"Authors: {authors}")
else:
    print("No articles found.")

```

```
C:\Users\maheshl\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\maheshl\PycharmProjects\pythonProject1\Uni\CB\T2.py
Total number of articles: 223385
Authors: Burgdorf JG, Ornstein KA, Liu B, Leff B, Brody AA, McDonough C, Ritchie CS
Authors: Matsuoka T, Oya N, Narumoto J, Kitani-Morii F, Niwa F, Mizuno T, Akazawa K, Yamada K, Abe M, Takano H, Wakasugi N, Shima A, Sawamoto N, Ito H, Toda W, Hanakawa T
Authors: Karimi Darabi M, Nazeri Z, Rafeeinia A, Pezeshki SP, Kheirollah A, Farbood Y, Adelipour M, Azizidoost S, Cheraghzadeh M
Authors: Zhang H, Xu C, Yuan C, Shi B, Zhu W, Wang H, Fu F, Tang D, Wang Y
Authors: Prado P, Medel V, Gonzalez-Gomez R, Sainz-Ballesteros A, Vidal V, Santamaria-Garcia H, Moguilner S, Mejia J, Slachevsky A, Beherens MI, Aguillon D, Lopera F, Parra MA, Matallana D, Maito MA, Garcia AM, Custodio N, Fun
Authors: Ssonko M, Hardy A, Naganathan V, Kalula S, Combrinck M
Authors: Lee S, Kim E, Moon CE, Park C, Lim JW, Baek M, Shin MK, Ki J, Cho H, Ji YW, Haam S
Authors: Fagon A, Poindessous JL
Authors: LÜ W, Zhang M, Yu W, Kuang W, Chen L, Zhang W, Yu J, LÜ Y
Authors: Dyer AH, Dolphin H, O'Connor A, Morrison L, Sedgwick G, McFeely A, Killeen E, Gallagher C, Davey N, Connolly E, Lyons S, Young C, Gaffney C, Ennis R, McHale C, Joseph J, Knight G, Kelly E, O'Farrelly C, Bourke NM, Fal
Authors: Almeida MF, Farizatto KLG, Almeida RS, Bahr BA
Authors: AbdEl-Raouf K, Farrag HSH, Rashed R, Ismail MA, El-Ganzuri MA, El-Sayed WM
Authors: Ye J, Wu C, Chen J, Wang H, Pan Y, Huang X, Wu J, Zhong X, Zhou H, Wang W, Wu S, Zhou T, Wang L, Lu P, Ruan C, Guo J, Ning Y, Xiao A
Authors: Dinh QN, Arumugam T
Authors: Hippman RS, Snead AM, Petros ZA, Korkmaz-Vaisys MA, Patel S, Sotelo D, Dobria A, Salkovski M, Nguyen TTA, Linares R, Cologna SM, Gowrishankar S, Aldrich LN
Authors: Sun X, Eastman G, Shi Y, Saibaba S, Oliveira AK, Lukens JR, Norambuena A, Thompson JA, Purdy MD, Dryden K, Pardo E, Mandell JW, Bloom GS
Authors: Jutkowitz E, Shewmaker P, Ford CB, Smith VA, O'Brien E, Shepherd-Banigan M, Belanger E, Plassman BL, Burke JR, Van Houtven CH, Wetle T
Authors: Lachner C, Craver EC, Babulal GM, Lucas JA, Ferman TJ, White RO, Graff-Radford NR, Day GS
Authors: Ghorat F, Sepidarkish M, Saadattalab F, Rezghi M, Shahrestani S, Gholamalizadeh M, Doaei S
Authors: Ramirez S, Koerich S, Astudillo N, De Gregorio N, Al-Lahham R, Allison T, Rocha NP, Wang F, Soto C

Process finished with exit code 0
```

Once It fetches only 20 records that's why there is only 20 authors entry in the Terminal.

3. Write a Biopython-program that finds CpG-islands from a given DNA-sequence.

```python
from Bio.Seq import Seq
from Bio.SeqUtils import nt_search

dna_seq = Seq(input("Enter the DNA sequence: "))

dna_seq_str = str(dna_seq)


def find_cpg_islands(sequence):
    cpg_positions = nt_search(sequence, "CG")[1:]
    islands = []
    current_island = []

    for pos in cpg_positions:
        if not current_island or pos == current_island[-1] + 1:
            current_island.append(pos)
        else:
            islands.append(current_island)
            current_island = [pos]

    islands.append(current_island)
    return islands


cpg_islands = find_cpg_islands(dna_seq_str)

```

```
C:\Users\maheshl\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\maheshl\PycharmProjects\pythonProject1\Uni\CB\T3.py
Enter the DNA sequence: GTGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
CpG Islands: [[18], [33]]

Process finished with exit code 0
```