Lab Sheet 1 *: *Basic Sequence Analysis Programs in Python

- Q1: Initialise a variable with a signal peptide sequence, change the case and print it. Also print the length of the sequence. (Hint- use string manipulation functions)
- Q2: Check the existence of a particular amino acid in a peptide sequence
- Q3: Count the number of Nucleotides in a DNA Sequence (Human Insulin)

  - Using a for loop
  - Using a while loop
  - Using python library function

- Q4: Calculate the 'GC Content' of a DNA Sequence
- Q5: Check the existence of a 'TATA Box' in a DNA sequence
- Q6: Find the Reverse Complement of a DNA String (Human Insulin) - Double Helix Form
- Q7:Convert a DNA sequence into an mRNA Sequence - (Transcription)
- Q8: Check the existence of a start codon in the mRNA sequence
- Q9: Check the existence of stop codon in the mRNA sequence
- Q10: Find positions of the start and stop codon present in an mRNA sequence. Extract the coding region using the start and stop codons.
- Q11: Use a python dictionary to convert an peptide chain from its one letter codon representation to three letter codon representation
- Q12 : Calculates the molecular weight of a protein based on weight of individual amino acids (Given as a dictionary).

```
from google.colab import drive
drive.mount('/content/drive')
```

```
#Q1: Initialise a variable with a signal peptide sequence, change the case and print it. Also print the length of the sequence.
#A signal peptide is a short peptide (usually 16-30 amino acids long) that control protein secretion and translocation in all living organis
signalpeptide = "maskatlllaftllfatcia"
signalpeptide = signalpeptide.upper()
count = len(signalpeptide);
print ("There are %d amino acids present in the given signal peptide :%s " %(count,signalpeptide));
```

```
There are 20 amino acids present in the given signal peptide :MASKATLLLAFTLLFATCIA
```

```
#Q2: Check the existence of a particular amino acid in a peptide sequence
#Check whether Lysine (R) is present in a peptide sequence

peptide ="MRVLLVALALLALAASATS"
if('K' in peptide):
  print(" The amino acid Lysine is present")
else:
  print(" The amino acid Lysine is absent")
```

```
 The amino acid Lysine is absent
```

```
#Q3: Count the Number of Nucleotides (Bases) in a DNA Sequence
#dna sequence of insulin gene
import re;
dna = """AGCCCTCCAGGACAGGCTGCATCAGAAGAGGCCATCAAGCAGGTCTGTTCCAAGGGCCTTTGCGTCAGGT
GGGCTCAGGATTCCAGGGTGGCTGGACCCCAGGCCCCAGCTCTGCAGCAGGGAGGACGTGGCTGGGCTCG
TGAAGCATGTGGGGGTGAGCCCAGGGGCCCCAAGGCAGGGCACCTGGCCTTCAGCCTGCCTCAGCCCTGC
CTGTCTCCCAGATCACTGTCCTTCTGCCATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTG
GCCCTCTGGGGACCTGACCCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAG
CTCTCTACCTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGACCT
GCAGGGTGAGCCAACTGCCCATTGCTGCCCCTGGCCGCCCCCAGCCACCCCCTGCTCCTGGCGCTCCCAC
CCAGCATGGGCAGAAGGGGGCAGGAGGCTGCCACCCAGCAGGGGGTCAGGTGCACTTTTTTAAAAAGAAG
TTCTCTTGGTCACGTCCTAAAAGTGACCAGCTCCCTGTGGCCCAGTCAGAATCTCAGCCTGAGGACGGTG
TTGGCTTCGGCAGCCCCGAGATACATCAGAGGGTGGGCACGCTCCTCCCTCCACTCGCCCCTCAAACAAA
TGCCCCGCAGCCCATTTCTCCACCCTCATTTGATGACCGCAGATTCAAGTGTTTTGTTAAGTAAAGTCCT
GGGTGACCTGGGGTCACAGGGTGCCCCACGCTGCCTGCCTCTGGGCGAACACCCCATCACGCCCGGAGGA
GGGCGTGGCTGCCTGCCTGAGTGGGCCAGACCCCTGTCGCCAGGCCTCACGGCAGCTCCATAGTCAGGAG
ATGGGGAAGATGCTGGGGACAGGCCCTGGGGAGAAGTACTGGGATCACCTGTTCAGGCTCCCACTGTGAC
GCTGCCCCGGGGCGGGGGAAGGAGGTGGGACATGTGGGCGTTGGGGCCTGTAGGTCCACACCCAGTGTGG
GTGACCCTCCCTCTAACCTGGGTCCAGCCCGGCTGGAGATGGGTGGGAGTGCGACCTAGGGCTGGCGGGC
AGGCGGGCACTGTGTCTCCCTGACTGTGTCCTCCTGTGTCCCTCTGCCTCGCCGCTGTTCCGGAACCTGC
TCTGCGCGGCACGTCCTGGCAGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGC
CCTTGGCCCTGGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCT
CTACCAGCTGGAGAACTACTGCAACTAGACGCAGCCCGCAGGCAGCCCCACACCCGCCGCCTCCTGCACC
GAGAGAGATGGAATAAAGCCCTTGAACCAGC"""
dna = re.sub('\s','',dna);



# Count the nucleotide Using for loop
dna = dna.upper()
C1=C2=C3=C4=0;
for i in range(len(dna)):
  if dna[i] == 'A':
    C1+=1
  if dna[i] == 'T':
    C2+=1
  if dna[i] == 'G':
    C3+=1
  if dna[i] == 'C':
    C4+=1
print("Number of bases: A : ", C1)
print("Number of bases: T : ", C2)
print("Number of bases: G : ", C3)
print("Number of bases: C : ", C4)


     Number of bases: A :  247
     Number of bases: T :  259
     Number of bases: G :  456
     Number of bases: C :  469


# Count the nucleotide Using while loop
dna = dna.upper()
C1=C2=C3=C4=0;
i = 0;
while i < len(dna):
  if dna[i] == 'A':
    C1+=1
  if dna[i] == 'T':
    C2+=1
  if dna[i] == 'G':
    C3+=1
  if dna[i] == 'C':
    C4+=1
  i = i+1
print("Number of bases: A : ", C1)
print("Number of bases: T : ", C2)
print("Number of bases: G : ", C3)
print("Number of bases: C : ", C4)

     Number of bases: A :  247
     Number of bases: T :  259
     Number of bases: G :  456
     Number of bases: C :  469
```

```
#Count nucleotides using python library functions
dna = dna.upper()
c1 = dna.count('A')
c2 = dna.count('T')
c3 = dna.count('G')
c4 = dna.count('C')
Total = len(dna)
print ("Total nucleotides are : %d" % (Total) )
print ("The frequency of nucleotides are : A->" + str(c1) + " T->" + str(c2) + " G->" + str(c3) + " C->" +str(c4))
```

```
Total nucleotides are : 1451
The frequency of nucleotides are : A->247 T->259 G->456 C->469
```

```
# Q4: Calculate the 'GC Content' of a DNA Sequence
# The GC pair is bound by three hydrogen bonds, while AT pairs are bound by two hydrogen bonds. And so,
# The GC content affects the stability of DNA.
# The GC content affects the secondary structure of mRNA.
# The GC content affects the annealing temperature for template DNA in PCR experiments.
# Formula : Count(G + C)/Count(A + T + G + C) * 100%.

c1 = dna.count('G')
c2 = dna.count('C')
total = len(dna)
print ("Total nucleotides present are %d " %(total))
gcpercentage = ( (c1+c2)/total ) *100
print (" The GC Percentage is : " +str(gcpercentage))
```

```
Total nucleotides present are 1451
 The GC Percentage is : 63.74913852515507
```

```
# Q5: Check the existence of a 'TATA Box' in a DNA sequence
# TATA box (also called the Goldberg–Hogness box) is a sequence of DNA found in the core promoter region of genes in archaea and eukaryotes
# TATAWAW, where W is either A or T.
import re;
exp = "TATA[AT]A[AT]";
dna2 = "TAGACGTTATAAAATGCCCTCAGATAGCCG"
matchobj = re.search(exp, dna2)
print("The TATA Box is present at postion: " , matchobj.start())
```

```
The TATA Box is present at postion:  7
```

** Q6: Problem : Find the Reverse Complement of a DNA String**

Description: In DNA strings, symbols 'A' and 'T' are complements of each other, as are 'C' and 'G'. Given a nucleotide p, we denote its complementary nucleotide as p. The reverse complement of a DNA string Pattern = p1…pn is the string Pattern = pn … p1 formed by taking the complement of each nucleotide in Pattern, then reversing the resulting string.

For example, the reverse complement of Pattern = "GTCA" is Pattern = "TGAC". Reverse Complement Problem

Find the reverse complement of a DNA string.

Given: A DNA string Pattern. Return: Pattern, the reverse complement of Pattern. Algorithm:

1. Find the complementary sequence
2. Reverse it.

Sample Dataset

AAAACCCGGT

Sample Output

ACCGGGTTTT

Visit http://rosalind.info/problems/ba1c/ . Solve the problem. Use the sample dataset given in the site.

```
#dna = "GTAGGGCACTACTTTTAACATGGAAGCAGGTGGATTGCACCAGGGGACCGACAGTCTGCAATCCTCCACCGACGACCGGATGGGACACCGCGATTGGACACACGGGAATGCTATGATGGCTTACGCTAACGA
dna="GTAGGGCACTACTTTT"
cdna=""
for i in range(len(dna)):
  if dna[i] == 'A':
    cdna += 'T'
  if dna[i] == 'T':
    cdna += 'A'
  if dna[i] == 'G':
    cdna += 'C'
  if dna[i] == 'C':
    cdna += 'G'
rcdna = cdna[::-1]
print("The double stranded DNA is : ")
#print (dna);
print (rcdna)
```

```
The double stranded DNA is :
AAAAGTAGTGCCCTAC
```

```
#Q7: Convert a DNA sequence into an mRNA Sequence
dna = """AGCCCTCCAGGACAGGCTGCATCAGAAGAGGCCATCAAGCAGGTCTGTTCCAAGGGCCTTTGCGTCAGGT
GGGCTCAGGATTCCAGGGTGGCTGGACCCCAGGCCCCAGCTCTGCAGCAGGGAGGACGTGGCTGGGCTCG
TGAAGCATGTGGGGGTGAGCCCAGGGGCCCCAAGGCAGGGCACCTGGCCTTCAGCCTGCCTCAGCCCTGC
CTGTCTCCCAGATCACTGTCCTTCTGCCATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTG
GCCCTCTGGGGACCTGACCCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAG
CTCTCTACCTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGACCT
GCAGGGTGAGCCAACTGCCCATTGCTGCCCCTGGCCGCCCCCAGCCACCCCCTGCTCCTGGCGCTCCCAC
CCAGCATGGGCAGAAGGGGGCAGGAGGCTGCCACCCAGCAGGGGGTCAGGTGCACTTTTTTAAAAAGAAG
TTCTCTTGGTCACGTCCTAAAAGTGACCAGCTCCCTGTGGCCCAGTCAGAATCTCAGCCTGAGGACGGTG
TTGGCTTCGGCAGCCCCGAGATACATCAGAGGGTGGGCACGCTCCTCCCTCCACTCGCCCCTCAAACAAA
TGCCCCGCAGCCCATTTCTCCACCCTCATTTGATGACCGCAGATTCAAGTGTTTTGTTAAGTAAAGTCCT
GGGTGACCTGGGGTCACAGGGTGCCCCACGCTGCCTGCCTCTGGGCGAACACCCCATCACGCCCGGAGGA
GGGCGTGGCTGCCTGCCTGAGTGGGCCAGACCCCTGTCGCCAGGCCTCACGGCAGCTCCATAGTCAGGAG
ATGGGGAAGATGCTGGGGACAGGCCCTGGGGAGAAGTACTGGGATCACCTGTTCAGGCTCCCACTGTGAC
GCTGCCCCGGGGCGGGGGAAGGAGGTGGGACATGTGGGCGTTGGGGCCTGTAGGTCCACACCCAGTGTGG
GTGACCCTCCCTCTAACCTGGGTCCAGCCCGGCTGGAGATGGGTGGGAGTGCGACCTAGGGCTGGCGGGC
AGGCGGGCACTGTGTCTCCCTGACTGTGTCCTCCTGTGTCCCTCTGCCTCGCCGCTGTTCCGGAACCTGC
TCTGCGCGGCACGTCCTGGCAGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGC
CCTTGGCCCTGGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCT
CTACCAGCTGGAGAACTACTGCAACTAGACGCAGCCCGCAGGCAGCCCCACACCCGCCGCCTCCTGCACC
GAGAGAGATGGAATAAAGCCCTTGAACCAGC"""
#dna= "TACGTGTC"
mrna=""
for base in dna:
  if(base == 'A'):
    mrna += 'U'
  if(base == 'T'):
    mrna += 'A'
  if(base == 'G'):
    mrna += 'C'
  if(base == 'C'):
    mrna += 'G'
```

```
print ("The mRNA sequence is: \n",mrna);
```

```
The mRNA sequence is:
 UCGGGAGGUCCUGUCCGACGUAGUCUUCUCCGGUAGUUCGUCCAGACAAGGUUCCCGGAAACGCAGUCCACCCGAGUCCUAAGGUCCCACCGACCUGGGGUCCGGGGUCGAGACGUCGUCCCUCCUGCACCGACC
```

```
#Q8: Check the existence of a start codon in the mRNA sequence
import re;
#mrna = "UUCUACAAUGCCUACCUAACA"
if(re.search("AUG",mrna)):
  print("The start codon AUG is present in the mRNA sequence");
  pos = re.search("AUG",mrna)
  print("The occurrence is at : ",pos.start());
else:
  print("The start codon AUG is  not present in the mRNA sequence");
```

```
The start codon AUG is present in the mRNA sequence
The occurrence is at :  355
```

```
#Q9: Check the existence of stop codon in the mRNA sequence
#Stop Codons are UAA,UAG,UGA
import re;
if(re.search("UAA|UAG|UGA",mrna)):
  print("The stop codon is present in the mRNA sequence");
  pos = re.search("UAA|UAG|UGA",mrna)
  print("The occurrence of ",pos.group()," is at : ",pos.start());
else:
  print("The stop codon not present in the mRNA sequence");
```

```
The stop codon is present in the mRNA sequence
The occurrence of  UAG  is at :  20
```

```
#Q10: Find positions of the start and stop codon present in an mRNA sequence. Extract the coding region using the start and stop codons.
exp = "AUG\w+UAA|UAG|UGA";
x= re.search(exp,mrna);
str = x.group()[:-3]
print("The coding region is (with stop codon) : ",x.group())
print("The coding region is : ",str)
```

```
The coding region is (with stop codon) :  AUGCCUACCUAA
The coding region is :  AUGCCUACC
```

```
#Q11: Use a python dictionary to convert an peptide chain from its one letter codon representation to three letter codon representation
three_letter_code = {'A':'ALA','C':'CYS','D':'ASP','E':'GLU','F':'PHE','G':'GLY',
                     'H':'HIS','I':'ILE','K':'LYS','L':'LEU','M':'MET','N':'ASN',
                     'P':'PRO','Q':'QLN','R':'ARG','S':'SER','T':'THR','V':'VAL',
                     'W':'TRP','Y':'TYR'}
peptide1 = "MASKATLLLAFTLLFATCIA"
peptide2 =""
for aa in peptide1:
     peptide2 +=three_letter_code[aa]
print("Polypeptide chain")
print("One letter code : ",peptide1)
print("Three letter code: ",peptide2)
```

```
Polypeptide chain
One letter code :  MASKATLLLAFTLLFATCIA
Three letter code:  METALASERLYSALATHRLEULEULEUALAPHETHRLEULEUPHEALATHRCYSILEALA
```

```
# Q12 : calculates the molecular weight of a protein based on weight  of individual amino acids (Given as a dictionary)
# There is a correction due to the fact that each bond involves the loss of a water molecule (with molecular weight of 18)
peptide = "MASKATLLLAFTLLFATCIA"
prot_weight = {'A':89, 'V':117, 'L':131, 'I':131, 'P':115,
 'F':165, 'W':204, 'M':149, 'G':75, 'S':105,
 'C':121, 'T':119, 'Y':181, 'N':132, 'O':146,
```