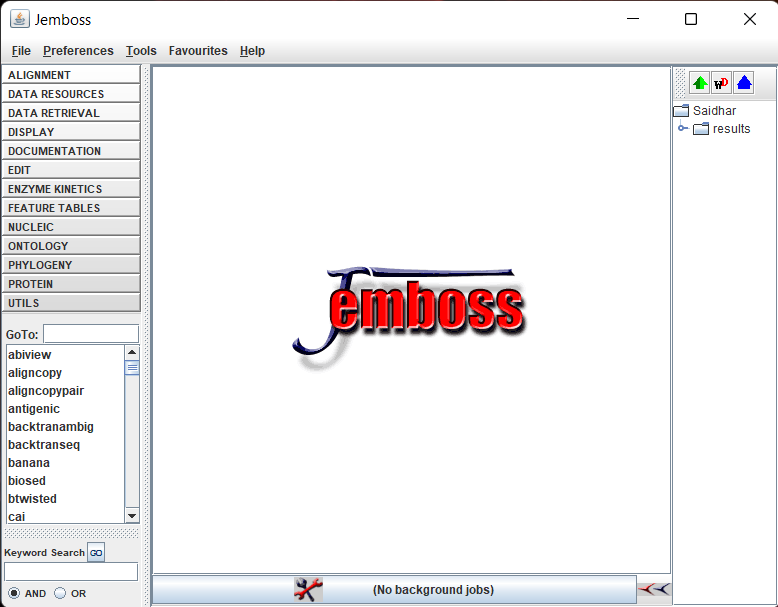
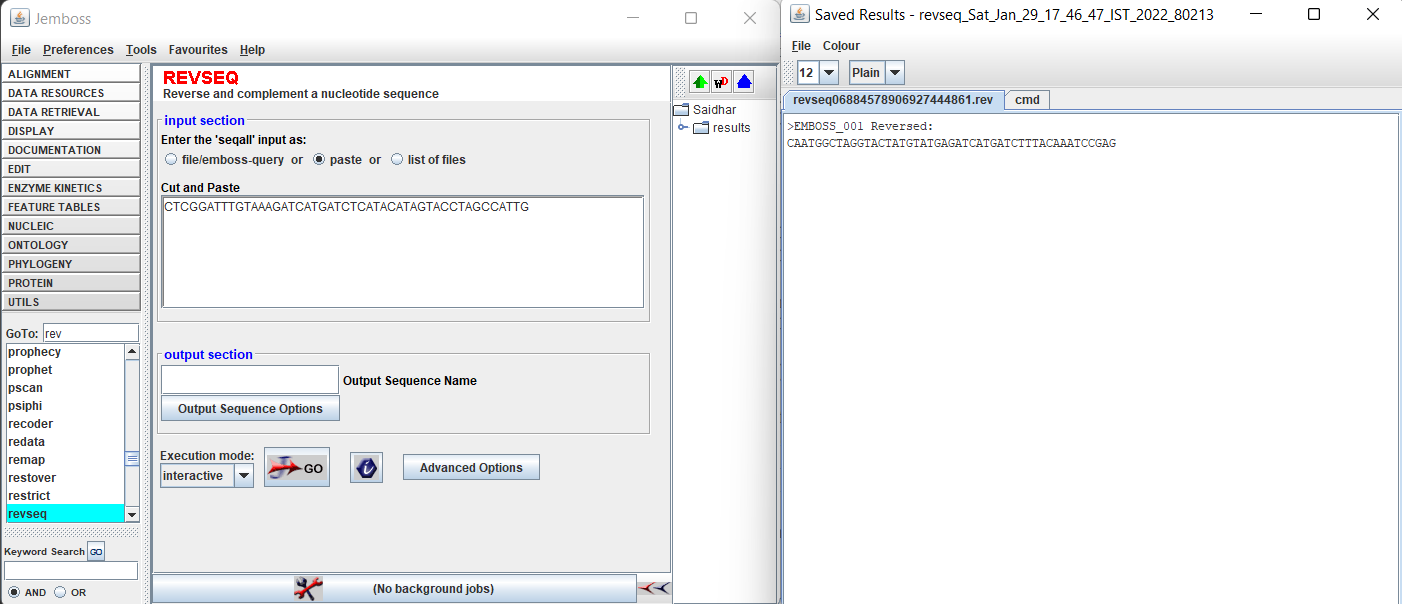
Assignment 1 BT3040

* Note: Though the codes are typed in this document, separate python files of code are submitted in the submission.

1. Steps taken to download the EMBOSS package:

* Visit the given website and scroll down to “Platform-Dependent Notes” section where we can find the ftp address to the package.
* Go to “My Computer/My PC” and right click and select “Add a Network Location” to add the FTP address in it.
* Proceed with the given steps to get the FTP address and download the package after the steps are completed.
* Install the EMBOSS package along with Java 32-bit version to get it running.

1. To find complementary sequence using EMBOSS:

* Open EMBOSS and search for the REVSEQ function
* Check the paste radio button and paste the given sequence.
* Click Go to get the results.

1. Code for Finding complementary sequence for the given DNA sequence:

string="CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG"

string=string.replace("C", "P")

string=string.replace("G", "C")

string=string.replace("A", "Q")

string=string.replace("T", "A")

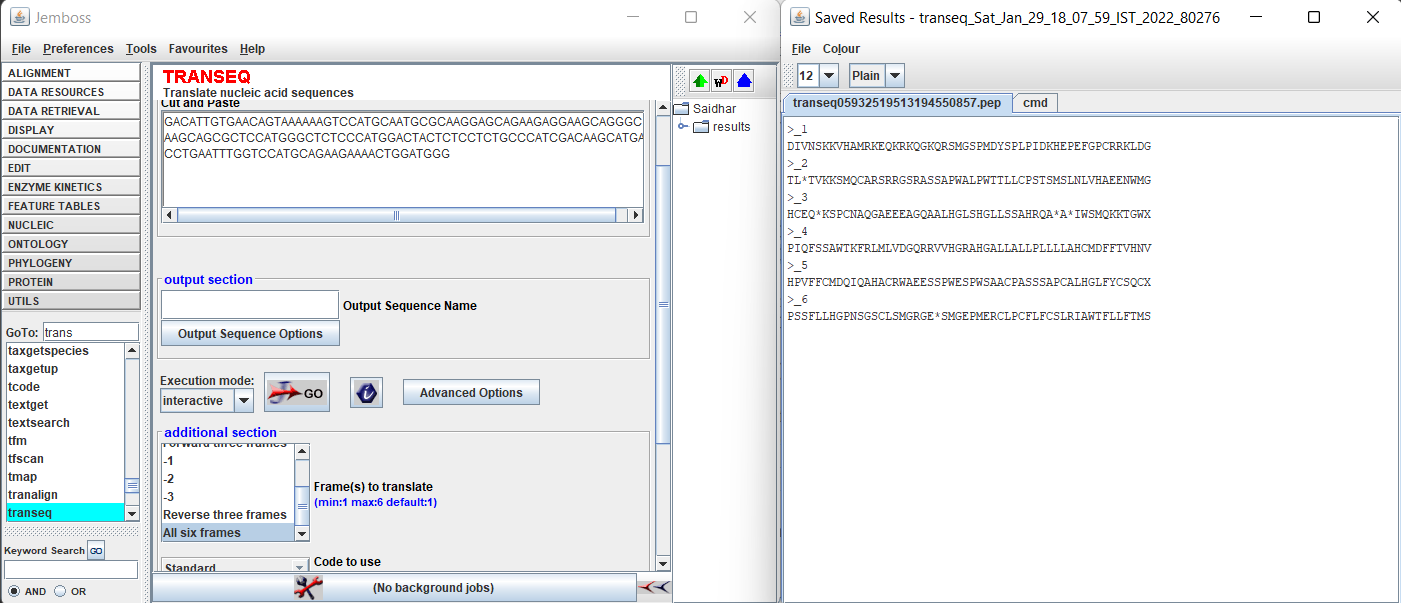
string=string.replace("P", "G")

string=string.replace("Q", "T")

string=string[::-1]

print(string)

1. Using the TRANSEQ function, we can find protein sequence for the given DNA sequence. (Make sure to select “All six reading Frames”)



(ii) From the above figure, we can say that the frame equivalent to the given sequence is the 4th protein sequence. That is the first reverse reading frame.

1. Program to translate DNA sequence to Protein sequence:

def match(a, b, c):

if (a=='A'):

if (b=='A'):

if (c=='T' or c=='C'):

return 'N'

elif (c=='A' or c=='G'):

return 'K'

elif (b=='C'):

return 'T'

elif (b=='G'):

if (c=='T'or c=='C'):

return 'S'

elif (c=='A'or c=='G'):

return 'R'

elif (b=='T'):

if (c=='T'or c=='C'or c=='A'):

return 'I'

elif (c=='G'):

return 'M'

elif (a=='C'):

if (b=='A'):

if (c=='T'or c=='C'):

return 'H'

elif (c=='A'or c=='G'):

return 'Q'

elif (b=='C'):

return 'P'

elif (b=='G'):

return 'R'

elif (b=='T'):

return 'L'

elif (a=='G'):

if (b=='A'):

if (c=='T'or c=='C'):

return 'D'

elif (c=='A'or c=='G'):

return 'E'

elif (b=='C'):

return 'A'

elif (b=='G'):

return 'G'

elif (b=='T'):

return 'V'

elif (a=='T'):

if (b=='A'):

if (c=='T'or c=='C'):

return 'Y'

elif (b=='C'):

return 'S'

elif (b=='G'):

if (c=='C'or c=='T'):

return 'C'

elif (c=='G'):

return 'W'

elif (b=='T'):

if (c=='C'or c=='T'):

return 'F'

elif (c=='A'or c=='G'):

return 'L'

string="GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG"

compstr=string[::-1]

compstr=compstr.replace("A", "P")

compstr=compstr.replace("C", "Q")

compstr=compstr.replace("G", "C")

compstr=compstr.replace("T", "A")

compstr=compstr.replace("P", "T")

compstr=compstr.replace("Q", "G")

s=[]

string=list(string)

compstr=list(compstr)

strstr=string.copy()

revstr=compstr.copy()

for j in range(0,6):

print("Reading Frame no.: ", j+1)

if (j<3):

s=[]

for i in range(int(len(string)/3)):

s.append(match(strstr[3\*i], strstr[3\*i+1], strstr[3\*i+2]))

s=['\*' if i is None else i for i in s]

if (j==2):

s.pop(len(s)-1)

s.append('X')

strstr.pop(0)

strstr.append(string[j])

for i in range(len(s)):

print(s[i], end='')

print("")

else:

s=[]

for i in range(int(len(string)/3)):

s.append(match(revstr[3\*i], revstr[3\*i+1], revstr[3\*i+2]))

s=['\*' if i is None else i for i in s]

if (j==5):

s.pop(len(s)-1)

s.append('X')

revstr.pop(0)

revstr.append(compstr[len(compstr)-1-j+3])

for i in range(len(s)):

print(s[i], end='')

print("")

1. Program to find the positions and total number of matches for a user defined string in a given DNA sequence:

def find\_motif(ms, ss):

arr=[]

for i in range(len(ms)-len(ss)):

if (ms[i:i+len(ss)]==ss):

arr.append(i+1)

return arr

mainstring="GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG"

substring=input()

arr=find\_motif(mainstring, substring)

print("total number of matches: ", len(arr))

print("Position of matches:")

for i in range(len(arr)):

print(arr[i], end=' ')

1. I’ve explored EMBOSS software and discovered some cool features. Some of them are:

|  |  |
| --- | --- |
| Description | Program Name |
| Calculating GC Content  Protein to Nucleotide Seq.  Plot Bending & Curvature of B-DNA  Twisting of B-DNA | GEECEE  BACKTRANSEQ  BANANA  BTWISTED |

1. Program to Calculate Average Base Stacking Energy for the given sequence:

def database(string):

if (string[0]=="A"):

if (string[1]=="A"):

return -4

elif (string[1]=="T"):

return -7

elif (string[1]=="C"):

return -5

elif (string[1]=="G"):

return -11

elif (string[0]=="T"):

if (string[1]=="A"):

return -7

elif (string[1]=="T"):

return -2

elif (string[1]=="C"):

return -3

elif (string[1]=="G"):

return -4

elif (string[0]=="C"):

if (string[1]=="A"):

return -9

elif (string[1]=="T"):

return -5

elif (string[1]=="C"):

return -6

elif (string[1]=="G"):

return -7

elif (string[0]=="G"):

if (string[1]=="A"):

return -9

elif (string[1]=="T"):

return -6

elif (string[1]=="C"):

return -4

elif (string[1]=="G"):

return 11

def avg\_base\_stack\_egy(string):

arr=[]

summ=0

for i in range(len(string)-1):

arr.append(string[i:i+2])

summ+=database(arr[i])

avg=summ/(len(arr))

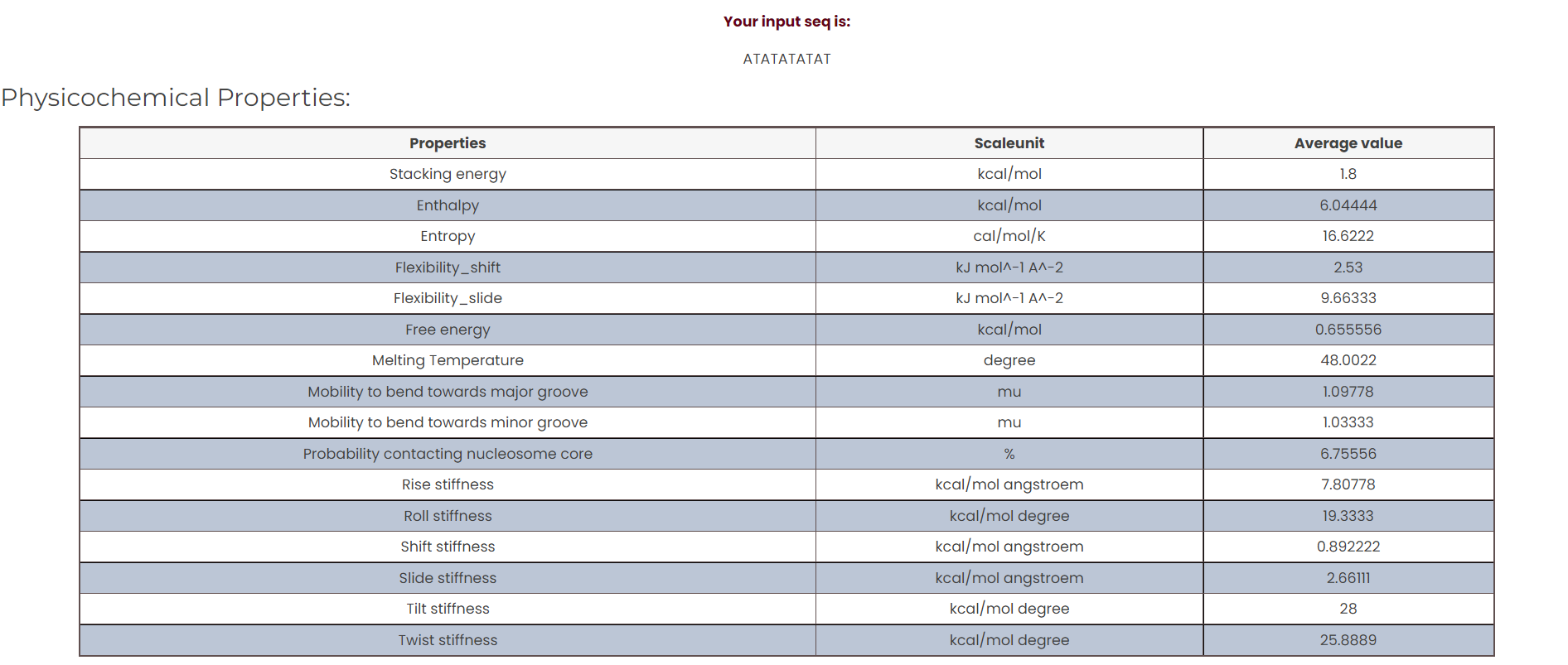
return avg

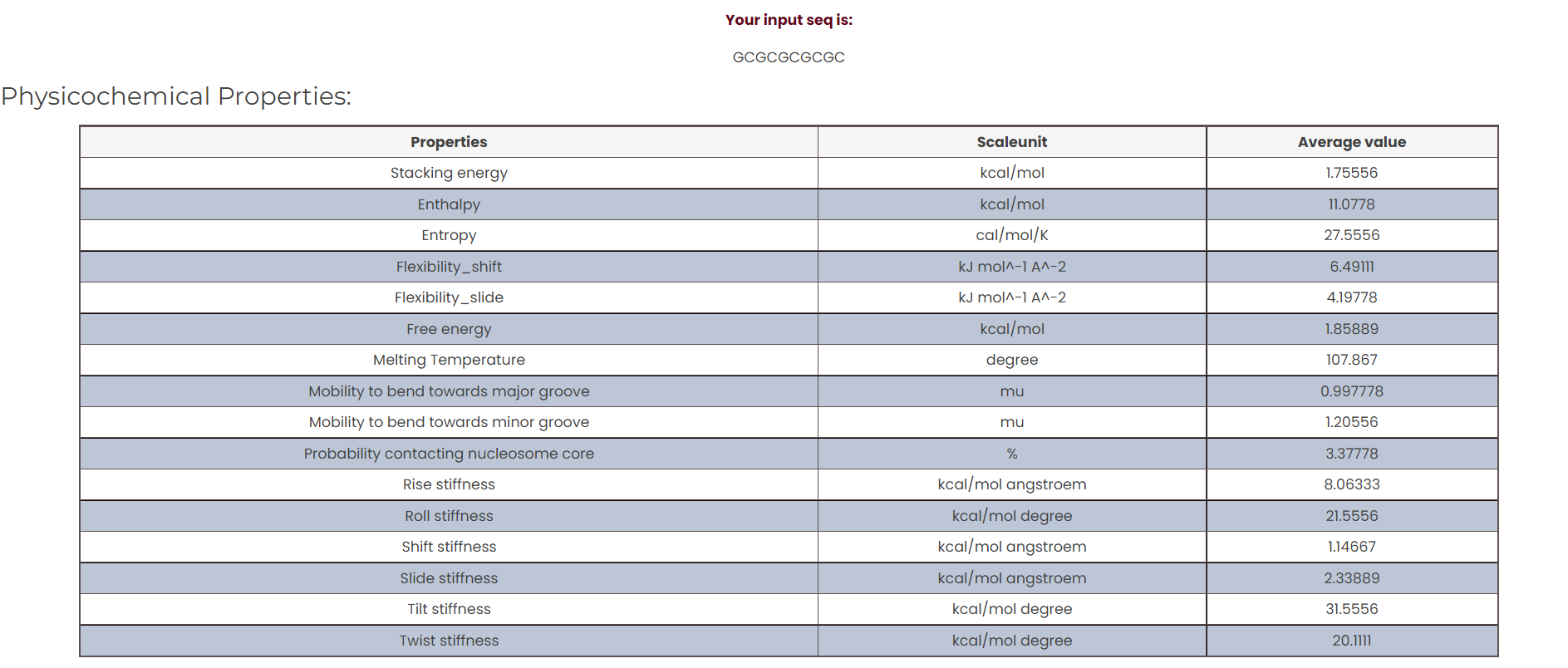
string="CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG"

print(avg\_base\_stack\_egy(string))

1. Using the Seq2Feature tool, to compute the average melting temperature,
2. Visit the site, click compute, click Average Property and enter the sequence in FASTA format. Then check the Physiochemical Properties check box and submit it. The results are as follows:
3. Melting Temperature: 48.0022
4. Melting Temperature: 107.867

And the reason why the Melting temperature of the 2nd sequence is higher than the first is the fact that the bond strength between the main and complementary strand. For the first sequence, i.e., for ATATATATAT, the complementary strand would be TATATATATA and the bonds between A and T are only two hydrogen bonds when compared to three hydrogen bonds of G and C which were present in 2nd sequence.





1. By checking the Nucleotide Content Checkbox in the Seq2Feature tool, we can find the AT and GC content of the given sequence.

AT\_Content: 58.333333

GC\_Content: 41.666667

