Assignment 7

1. To find the amino acid composition of the given sequences:
   - Make an empty dictionary and update it as per the number of same type of letter that comes.
   - If wanted in alphabetical form, make an array which contains the single letters of all amino acids in alphabetical form and print it as per those values.

   The code looks as follows:

```
import pandas as pd

strings=['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',
         '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

'''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQ
NLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']
AA_all=['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y']
pandadict={'A':[],'C':[],'D':[],'E':[],'F':[],'G':[],'H':[],'I':[],'K':[],'L':[],'M':[],'N':[],
           'P':[],'Q':[],'R':[],'S':[],'T':[],'V':[],'W':[],'Y':[]}

for i in strings:
    aacompdict={}
    for j in range(len(i)):
        if i[j] in aacompdict:
            aacompdict.update({i[j]:aacompdict[i[j]]+1})
        else:
            aacompdict[i[j]]=1
    for j in aacompdict:
        aacompdict[j]=aacompdict[j]*100/len(i)
    #print(aacompdict, '\n')
```

```
    '''for j in AA_all:
        if j in i:
            print(j, ':', aacompdict[j], end=" ")
        else:
            print(j, ':', '0', end=" ")'''
    for i in AA_all:
        if i in aacompdict:
            pandadict[i].append(aacompdict[i])
        else:
            pandadict[i].append(0)
    aacompdict.clear()
print(pd.DataFrame.from_dict(pandadict, orient='index', columns=['Seq 1', 'Seq 2', 'Seq 3']))
```

The Output is as Follows:

```
       Seq 1      Seq 2      Seq 3
A  17.910448  19.117647  10.596026
C   1.492537   0.000000   0.000000
D   2.985075   2.941176   5.960265
E   2.985075   1.470588   3.973510
F   1.492537   5.882353   5.298013
G   5.970149  14.705882  15.231788
H   4.477612   0.000000   1.324503
I   1.492537  11.764706   5.298013
K   7.462687   1.470588   3.973510
L   8.955224  13.235294   5.960265
M   0.000000   7.352941   0.662252
N   1.492537   0.000000   3.311258
P   7.462687   4.411765   2.649007
Q   1.492537   2.941176   3.311258
R   5.970149   2.941176   3.311258
S   5.970149   0.000000   9.933775
T  10.447761   1.470588   4.635762
```

```
V   1.492537   8.823529   5.298013
W   5.970149   0.000000   3.973510
Y   4.477612   1.470588   5.298013
```

2. The molecular weight of the given sequences can be calculated as summing the weights of individual amino acids and subtracting the water molecule which will be ejected when two amino acids bond. The Code for this Problems is as follows:

```
mwdict={'A': 85, 'C': 115, 'D': 130, 'E': 145, 'F': 160, 'G': 70, 'W': 200,
'H': 150, 'I': 125, 'K': 145, 'L': 125, 'M': 143, 'N': 130, 'Y': 175,
'P': 110, 'Q': 140, 'R': 170, 'S': 100, 'T': 115, 'V': 110}

strings=['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',
         '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

'''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQ
NLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']

for i in strings:
    mw=0
    aacompdict={}
    for j in range(len(i)):
        if i[j] in aacompdict:
            aacompdict.update({i[j]:aacompdict[i[j]]+1})
        else:
            aacompdict[i[j]]=1
    for k in aacompdict:
        mw+=aacompdict[k]*mwdict[k]
    mw-=(len(strings[strings.index(i)])-1)*18
    #print("Molecular Weight of seq", i, "=", mw, "\n")
    print("Molecular Weight of seq", strings.index(i)+1, "=", mw, "\n")
```

and the Output is:

```
Molecular Weight of seq 1 = 7127

Molecular Weight of seq 2 = 6529

Molecular Weight of seq 3 = 15453
```

3. From the result obtained form question 1, we take the standard deviation of that with the individual protein molecules of group 2 and add them. The smaller result is the one which group the given sequence belongs to.
   The code is as follows:

```
group_A = {'A': 8.47, 'D': 5.97, 'C': 1.39, 'E': 6.32, 'T': 5.79,
'F': 3.91, 'G': 7.82, 'H': 2.26, 'I': 5.71, 'V': 7.02,
'K': 5.76, 'L': 8.48, 'M': 2.21, 'N': 4.54, 'W': 1.44,
'P': 4.63, 'Q': 3.82, 'R': 4.93, 'S': 5.94, 'Y': 3.58}
group_B = {'A': 8.95, 'D': 5.91, 'C': 0.47, 'E': 4.78, 'T': 6.54,
'F': 3.68, 'G': 8.54, 'H': 1.25, 'I': 4.77, 'V': 6.76,
'K': 4.93, 'L': 8.78, 'M': 1.56, 'N': 5.74, 'W': 1.24,
'P': 3.74, 'Q': 4.75, 'R': 5.24, 'S': 8.05, 'Y': 4.13}
aacompdict={}
strings=['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',
         '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

'''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQ
NLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']
for i in strings:
    arra=[]
    arrb=[]
    print("sequence serial number", strings.index(i)+1)
    for j in range(len(i)):
        if i[j] in aacompdict:
            aacompdict.update({i[j]:aacompdict[i[j]]+1})
        else:
```

```
            aacompdict[i[j]]=1
    for j in aacompdict:
        aacompdict[j]=aacompdict[j]*100/len(i)
    for k in aacompdict:
        arra.append(aacompdict[k]-group_A[k])
        arrb.append(aacompdict[k]-group_B[k])
    for k in range(len(arra)):
        arra[k]*=arra[k]
        arrb[k]*=arrb[k]
    print("std dev from A=", end=" ")
    print(sum(arra))
    print("std dev from B=", end=" ")
    print(sum(arrb))
    if sum(arra)>sum(arrb):
        print("The given sequence is close to B")
    else:
        print("the given sequence is close to A")
```

4. Computing the residue pair preferences for the given three sequences with the given conditions:

```
import pandas as pd
def ResPairPref(seq):
    AA_all=['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y']

    Pref_1 = [[0 for i in range(len(AA_all))] for i in range(len(AA_all))]
    Pref_2 = [[0 for i in range(len(AA_all))] for i in range(len(AA_all))]
    Pref_3 = [[0 for i in range(len(AA_all))] for i in range(len(AA_all))]
    composition = [0]*len(AA_all)
    for i in range(len(seq)):
        aa = seq[i]
        ind = AA_all.index(aa)
        composition[ind]+=1
```

```
    for i in AA_all:
        for j in AA_all:
            pair = i+j
            for k in range(len(seq)-1):
                if pair==seq[k:k+2]:
                    Pref_1[AA_all.index(i)][AA_all.index(j)]+=1
                    Pref_2[AA_all.index(i)][AA_all.index(j)]+=1
                    Pref_3[AA_all.index(i)][AA_all.index(j)]+=1
            if composition[AA_all.index(i)]==0 or composition[AA_all.index(j)]==0:
                Pref_1[AA_all.index(i)][AA_all.index(j)]=0
                Pref_2[AA_all.index(i)][AA_all.index(j)]=0
                Pref_3[AA_all.index(i)][AA_all.index(j)]=0
            else:

Pref_1[AA_all.index(i)][AA_all.index(j)]=(Pref_1[AA_all.index(i)][AA_all.index(j)]*100)/(composition
[AA_all.index(i)]+composition[AA_all.index(j)])

Pref_2[AA_all.index(i)][AA_all.index(j)]=(Pref_2[AA_all.index(i)][AA_all.index(j)]*100)/(len(seq)-1)

Pref_3[AA_all.index(i)][AA_all.index(j)]=(Pref_3[AA_all.index(i)][AA_all.index(j)]*100)/(composition
[AA_all.index(i)]*composition[AA_all.index(j)])
    print('\nPair-wise preference score for the given sequnce:')
    print('\nFormula 1 : ')
    print(pd.DataFrame(Pref_1, columns=AA_all, dtype=float).to_string())

    print('\nPair-wise preference score for the given sequnce:')
    print('\nFormula 2 : ')
    print(pd.DataFrame(Pref_2, columns=AA_all, dtype=float).to_string())

    print('\nPair-wise preference score for the given sequnce:')
    print('\nFormula 3 : ')
    print(pd.DataFrame(Pref_3, columns=AA_all, dtype=float).to_string())
```

```
    '''for i in range(AA_all):
        for j in range(AA_all):
            print(Pref_1[i][j], end=' ')
        print('\n')

    print('\nFormula 2 : ')
    for i in range(AA_all):
        for j in range(AA_all):
            print(Pref_2[i][j], end=' ')
    print('\n')

    print('\nFormula 3 : ')
    for i in range(AA_all):
        for j in range(AA_all):
            print(Pref_3[i][j], end=' ')
    print('\n')
    return Pref_1,Pref_2,Pref_3'''

strings=['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',
         '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

'''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQ
NLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']
for i in strings:
    print("For Sequence", strings.index(i))
    ResPairPref(i)
```

*The Output is attached to Excel File.

5. The Hydrophobicity, Helical contact area and total non-bonded energy for the given sequences can be calculated using the following way:

```
hgmdict={'A': 13.85, 'C': 15.37, 'D': 11.61, 'E': 11.38, 'F': 13.93, 'G': 11.38, 'W': 15.48,
```

```
'H': 13.82, 'I': 15.28, 'K': 11.58, 'L': 14.13, 'M': 13.86, 'N': 11.61, 'Y': 13.88,
'P': 12.35, 'Q': 12.61, 'R': 13.10, 'S': 13.39, 'T': 12.70, 'V': 14.56}
cadict={'A': 20, 'C': 25, 'D': 26, 'E': 33, 'F': 46, 'G': 13, 'W': 61,
'H': 37, 'I': 39, 'K': 46, 'L': 35, 'M': 43, 'N': 28, 'Y': 46,
'P': 22, 'Q': 36, 'R': 55, 'S': 20, 'T': 28, 'V': 33}
etdict={'A': 1.9, 'C': 2.04, 'D': 1.52, 'E': 1.54, 'F': 1.86, 'G': 1.9, 'W': 1.87,
'H': 1.76, 'I': 1.95, 'K': 1.37, 'L': 1.97, 'M': 1.96, 'N': 1.56, 'Y': 1.69,
'P': 1.7, 'Q': 1.52, 'R': 1.48, 'S': 1.75, 'T': 1.77, 'V': 1.98}

strings=['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',
         '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

'''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQ
NLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']
for i in strings:
    hgm=0
    ca=0
    et=0
    for j in i:
        hgm+=hgmdict[j]
        ca+=cadict[j]
        et+=etdict[j]
    hgm=hgm/len(i)
    print("For Sequence", strings.index(i)+1)
    print("Hydrophobicity:", hgm)
    print("Helical Contact Area:", ca)
    print("Total Non-Bonded Energy:", et)
```

And the output is:

```
For Sequence 0
Hydrophobicity: 13.214477611940309
Helical Contact Area: 2156
```

```
Total Non-Bonded Energy: 117.74000000000005
For Sequence 1
Hydrophobicity: 13.483382352941172
Helical Contact Area: 2067
Total Non-Bonded Energy: 126.66000000000003
For Sequence 2
Hydrophobicity: 13.073443708609275
Helical Contact Area: 4616
Total Non-Bonded Energy: 267.7500000000001
```