Flow Control

- Due: January 30, 2023 at 11:55 pm, submit in Canvas
- program filename: proteinParams.py
- total points possible: 44
- extra-credit possible: 6
- submit two files:
  - Lab03.ipynb (the notebook), and
  - proteinParams.py (a python file that contains your code)

Building and Testing

For this lab, you will submit two files: the notebook with your inspection info and results, and file named proteinParams.py. You can do all of your work from within jupyter, then copy/paste your program to a new text file using your favorite text editor or python IDE. Test that program file !!! On Mac systems you will use terminal, and on windows systems you will likely use cmd. You will then navigate to the directory where your proteinParam.py lives, then execute:
python3 proteinParam.py
or
python proteinParam.py

Protein parameters

In this exercise, you'll create a Python program to calculate the physical-chemical properties of a protein sequence similar to what ProtParam outputs. Your task is to develop the ProteinParam class included below in template form. The program includes the methods that you will need, along with the completed "main" that does all of the input and output.

When testing, you will type the protein sequence, hit return, the program will respond with the required output. You can then enter a new protein string and a new analysis will be presented. When you are finished, type ctrl-D to signal the end (this is done by holding the control key down as you type the letter D).

Your program will read in the protein sequence and print out the:

- number of amino acids and total molecular weight,
- Molar extinction coefficient and Mass extinction coefficient,
- theoretical isoelectric point (pI), and
- amino acid composition

For example, if I enter the protein sequence: VLSPADKTNVKAAW

then the program should output:

```
Number of Amino Acids: 14
Molecular Weight: 1499.7
molar Extinction coefficient: 5500.00
mass Extinction coefficient: 3.67
Theoretical pI: 9.88
Amino acid composition:
A = 21.43%
C = 0.00%
D = 7.14%
E = 0.00%
F = 0.00%
G = 0.00%
H = 0.00%
I = 0.00%
K = 14.29%
L = 7.14%
M = 0.00%
N = 7.14%
P = 7.14%
Q = 0.00%
R = 0.00%
S = 7.14%
T = 7.14%
V = 14.29%
W = 7.14%
Y = 0.00%
```

Hints:

The input sequence is not guaranteed to be uppercase and might contain unexpected characters.
Only count the following (A, C, D, E, F, G, H, I, L, K, M, N, P, Q, R, S, T, V, Y, W) or the lower-case equivalents, and ignore anything else. Math details are included below.

To get full credit on this assignment, your code needs to:

- Run properly (execute and produce the correct output)
- Include a docstring overview about what your program is designed to do with expected inputs and outputs
- include docstrings for every class, method within that class
- Include any assumptions or design decisions you made in writing your code as # comments
- Contain in-line comments using #-style where appropriate. Make sure to fix the template (below) to conform.

- Submit your proteinParams.py file and the notebook using Canvas.

Congratulations, you finished your third lab assignment!

Design specification

__init__

There are a number of ways to design this. Your __init__ method could save an attribute which is just the input string. A more effective solution would compute and save the aaComposition dictionary. All of the protein parameter methods can operate very efficiently using a dictionary. Use of an aaComposition here will save you quite a bit of work.

aaCount()

This method will return a single integer count of valid amino acid characters found. Do not assume that this is the length of the input string, since you might have spaces or invalid characters that are required to be ignored.

aaComposition() - 4 points

This method is to return a dictionary keyed by single letter Amino acid code, and having associated values that are the counts of those amino acids in the sequence. Make sure to include all 20 amino acids. Proper amino acids that are not represented in the sequence should have a value of zero. Note: if you have already calculated a composition dictionary in **init**, then just return that dictionary here.

molecularWeight() - 8 points

This method calculates the molecular weight (MW) of the protein sequence. If we have the composition of our protein, this is done by summing the weights of the individual Amino acids and excluding the waters that are released with peptide bond formation.

$$MW_{H_2O} + \sum_{aa}^{Nres} N_{aa}(MW_{aa} - MW_{H_2O})$$

charge(pH) -- 10 points

This method calculates the net charge on the protein at a specific pH (specified as a parameter of this method). The method is used by the pI method. I have marked it with the single _ notation to advise future users of the class that this is not part of the defined interface and it just might change.

If we have the composition of our protein, we can then calculate the net charge at a particular pH, using the pKa of each charged Amino acid and the Nterminus and Cterminus.

$$netCharge = \left[\sum_{aa=\{Arg,Lys,His,Nterm}^{} N_{aa} \frac{10^{pKa(aa)}}{10^{pKa(aa)} + 10^{pH}}\right] - \left[\sum_{aa=\{Asp,Glu,Cys,Tyr,Cterm}^{} N_{aa} \frac{10^{pH}}{10^{pKa(aa)} + 10^{pH}}\right]$$

I have provided pKa tables for each AA, and the pKa for the N-terminus and C-terminus.

pI() - 10 points

The theoretical isoelectric point can be estimated by finding the particular pH that yields a neutral net Charge (close to 0). There are a few ways of doing this, but the simplest might be to iterate over all pH values to find the one that is closest to 0. Doing this by hand is painful, but its not that bad to do computationally. Remember that we want to find the best pH, accurate to 2 decimal places.

extra credit (3 points) for pI() method

Another way of doing the pI calculation would use a binary search over the pH range. This works because we expect a single zero crossing to exist in the range, and the function will be well behaved across the range of charge() as a function of pH (0-14 range). You then make the algorithm operate to any specified precision using an optional parameter (set the default parameter: precision = 2).

molarExtinction() - 8 points

The extinction coefficient indicates how much light a protein absorbs at a certain wavelength. It is useful to have an estimation of this coefficient for measuring a protein with a spectrophotometer at a wavelength of 280nm. It has been shown by Gill and von Hippel that it is possible to estimate the molar extinction coefficient of a protein from knowledge of its amino acid composition alone. From the molar extinction coefficient of tyrosine, tryptophan and cystine at a given wavelength, the extinction coefficient of the native protein in water can be computed using the following equation.

$$E = N_Y E_Y + N_W E_W + N_C E_C$$

where:

- N_Y is the number of tyrosines, N_W is the number of tryptophans, N_C is the number of cysteines,
- E_Y, E_W, E_C are the extinction coefficients for tyrosine, tryptophan, and cysteine respectively.

I have supplied the molar extinction coefficients at 280nm for each of these residues in a dictionary(aa2abs280) in the program template.

Note that we will assume for this exercise that all Cysteine residues are represented as Cystine. Under reducing conditions, Cystine does not form however and Cysteine residues do not contribute to absorbance at 280nm.

massExtinction()

We can calculate the Mass extinction coefficient from the Molar Extinction coefficient by dividing by the molecularWeight of the corresponding protein.

extra credit for molarExtinction() and massExtinction() - 3 points

As mentioned above, we are assuming that all Cysteine residues are present as Cystine. Provide an optional parameter to both molarExtinction() and massExtinction() to calculate these values assuming reducing conditions. Use an optional parameter with a default of True (Cystine=True) to allow evaluation of both molar and mass extinction under both oxidizing (default) and reducing conditions.

Inspection info

Describe all of the information that your inspection team needs to know to understand your design and implementation. Examples:

- How did you save the essential data attribute in objects of ProteinParam ?
- How did you make use of that save attribute for each of your methods ?
- How did you implement the charge method? how is pH given to charge by the pI method ?
- How did you iterate across the range of pH in order to get 2 decimal points of precision (7.16, for example)
- How did you calculate massExtinction coefficient without having to redo your work from molarExtinction
- How did you make use of the many dictionaries that are given in order to avoid having to build them from scratch ?

Protein Param

In [ ]:
```
#!/usr/bin/env python3
# Name: Your full name (cruzID)
# Group Members: List full names (and cruzIDs) or "None"

class ProteinParam :
# These tables are for calculating:
#     molecular weight (aa2mw), along with the mol. weight of H2O (mw2O)
#     absorbance at 280 nm (aa2abs280)
#     pKa of positively charged Amino Acids (aa2chargePos)
#     pKa of negatively charged Amino acids (aa2chargeNeg)
#     and the constants aaNterm and aaCterm for pKa of the respective termini
#  Feel free to move these to appropriate methods as you like

# As written, these are accessed as class attributes, for example:
# ProteinParam.aa2mw['A'] or ProteinParam.mw2O

    aa2mw = {
        'A': 89.093,  'G': 75.067,  'M': 149.211, 'S': 105.093, 'C': 121.158,
        'H': 155.155, 'N': 132.118, 'T': 119.119, 'D': 133.103, 'I': 131.173,
        'P': 115.131, 'V': 117.146, 'E': 147.129, 'K': 146.188, 'Q': 146.145,
        'W': 204.225,  'F': 165.189, 'L': 131.173, 'R': 174.201, 'Y': 181.189
        }

    mw2O = 18.015
    aa2abs280= {'Y':1490,  'W': 5500,  'C': 125}

    aa2chargePos = {'K': 10.5,  'R':12.4,  'H':6}
    aa2chargeNeg = {'D': 3.86,  'E': 4.25,  'C': 8.33,  'Y': 10}
    aaNterm = 9.69
    aaCterm = 2.34

    def __init__ (self, protein):
        pass

    def aaCount (self):
        pass

    def pI (self):
        pass

    def aaComposition (self) :
        pass

    def _charge_ (self):
        pass

    def molarExtinction (self):
        pass

    def massExtinction (self):
        myMW =  self.molecularWeight()
        return self.molarExtinction() / myMW if myMW else 0.0

    def molecularWeight (self):
        pass

# Please do not modify any of the following.  This will produce a standard output that can be parsed

import sys
def main():
    inString = input('protein sequence?')
    while inString :
        myParamMaker = ProteinParam(inString)
        myAnumber = myParamMaker.aaCount()
        print ("Number of Amino Acids: {aaNum}".format(aaNum = myAnumber))
        print ("Molecular Weight: {:.1f}".format(myParamMaker.molecularWeight()))
        print ("molar Extinction coefficient: {:.2f}".format(myParamMaker.molarExtinction()))
        print ("mass Extinction coefficient: {:.2f}".format(myParamMaker.massExtinction()))
        print ("Theoretical pI: {:.2f}".format(myParamMaker.pI()))
        print ("Amino acid composition:")

        if myAnumber == 0 : myAnumber = 1  # handles the case where no AA are present

        for aa,n in sorted(myParamMaker.aaComposition().items(),
                           key= lambda item:item[0]):
            print ("\t{} = {:.2%}".format(aa, n/myAnumber))

        inString = input('protein sequence?')

if __name__ == "__main__":
    main()
```

Inspection results

Who participated in your code inspection? What did they suggest? How was the inspection valuable to you or to your team ?