Exercise 1

Due: Wednesday, 10/5/22

Objective: To learn how to use Python to read files, process string data, manipulate lists and arrays, output to display, and perform simple numerical calculations.

Proteins are sequence-defined biopolymers made of the natural twenty amino acids. Unlike synthetic polymers, most proteins spontaneously organize or "fold" to a unique three dimensional structure in aqueous solution, the dominant environment in the cell. This process is driven by the protein's desire to tuck hydrophobic amino acids in the core of the structure away from water (which is ~70% of cells).

The determination of the 3D structure of a particular protein (i.e., the x, y, z coordinates of all of its constituent atoms) is a major experimental enterprise as it provides important clues to protein function. The Protein Data Bank (www.pdb.org) is an open repository for experimentally determined structures of proteins, currently with nearly 200k. Structures and experimental details are contained in .pdb files that can be freely downloaded.

In this assignment, you will write a Python program to compute the radius of gyration R_g of a protein structure read from a .pdb file. You will compare the R_g for all amino acid residues (monomers) in a given protein with that for only the hydrophobic ones, and compute a typical ratio $R_{g,\text{phobic}}/R_{g,\text{all}}$ to show that hydrophobic residues are closer to the protein core on average.

In order to simplify the exercise, you will not consider every atom in a protein structure, but only one atom per amino acid residue (monomer): the so-called alpha carbon, notated with the code CA, is the central atom of every amino acid along the protein backbone. In addition, you will need to keep track of the type of each amino acid. There are twenty natural amino acids that comprise protein structures, and these are represented by three-letter codes (e.g., ALA for alanine and LYS for lysine).

Part a

Download from the course site the collection of protein structures (.pdb files) that will be used in this example. The files are compressed in an archive; unzip this archive to a new path on your computer where you will store your code and perform your computations.

Part b

Begin a new Python module named "**`exercise1.py`**". Add to this module a function **`ReadPdb`**, with the following characteristics:

```
function:
       ReadPdb
arguments:
       PdbFile, the string filename of a pdb file
returns:
       Pos, a (N,3) dimensional NumPy array containing the position of the alpha
          carbon in each of the N amino acid residues
       ResNames, a length-N list containing the three-letter code for the type of
          each of the corresponding amino acid residues
```

Protein .pdb files are in text format and follow simple data-storing rules. These files can contain lots of extra data, like experimental conditions, but you will only consider the small subset of data corresponding to the actual atomic coordinates. All of the lines in this data subsection start with the keyword "ATOM". Moreover, when you arrive at a line that starts with the keyword "TER", you have reached the end of the data and should stop reading the file. Thus, you can process the data by reading a file line-by-line and filtering for those that start with these two keywords. A sample excerpt from a pdb file is shown below:

```
ATOM      1  N   MET A   1     -14.136   1.321   3.616  1.00  0.93           N
ATOM      2  CA  MET A   1     -13.451   0.063   4.032  1.00  0.36           C
ATOM      3  C   MET A   1     -11.981   0.360   4.336  1.00  0.36           C
ATOM      4  O   MET A   1     -11.557   1.499   4.315  1.00  0.64           O
ATOM      5  CB  MET A   1     -13.532  -0.987   2.924  1.00  1.26           C
ATOM      6  CG  MET A   1     -14.934  -0.967   2.313  1.00  1.15           C
…
ATOM    853  HB3 GLU A  56      12.401  -1.598  -3.876  1.00  0.11           H
ATOM    854  HG2 GLU A  56      11.996   0.639  -4.981  1.00  0.95           H
ATOM    855  HG3 GLU A  56      10.814   0.986  -3.722  1.00  0.97           H
TER     856      GLU A  56
```

The format of the ATOM lines is as follows (from www.pdb.org):

COLUMNS	DATA TYPE	FIELD	DEFINITION
1 - 6	Record name	"ATOM "	
7 - 11	Integer	serial	Atom serial number.
13 - 16	**Atom**	**name**	**Atom name.**
17	Character	altLoc	Alternate location indicator.
18 - 20	**Residue name**	**resName**	**Residue name.**
22	Character	chainID	Chain identifier.
23 - 26	Integer	resSeq	Residue sequence number.
27	AChar	iCode	Code for insertion of residues.
31 - 38	**Real(8.3)**	**x**	**Orthogonal coordinates for X in Angstroms.**
39 - 46	**Real(8.3)**	**y**	**Orthogonal coordinates for Y in Angstroms.**
47 - 54	**Real(8.3)**	**z**	**Orthogonal coordinates for Z in Angstroms.**
55 - 60	Real(6.2)	occupancy	Occupancy.
61 - 66	Real(6.2)	tempFactor	Temperature factor.
77 - 78	LString(2)	element	Element symbol, right-justified.
79 - 80	LString(2)	charge	Charge on the atom.

You will only focus on five subsections of each line: the atom name, the residue name, and the x, y, z coordinate fields. In addition, you will only use lines for atoms named CA, indicating the alpha carbon.

Part c

Add to your module a function **`ResHydrophobic`** with the following characteristics:

```
function:
      ResHydrophobic
arguments:
      ResNames, length-N list of three letter residue codes
returns:
      IsPhobic, a length-N Boolean array with True or False values indicating
         whether the residue codes in ResNames correspond to hydrophobic or
         hydrophilic residues
```

You may consider the following amino acid residues to be hydrophobic:

hydrophobic amino acid residue name	**three letter code**
alanine	ALA
cysteine	CYS
phenylalanine	PHE
isoleucine	ILE
leucine	LEU
methionine	MET
proline	PRO
valine	VAL
tryptophan	TRP

The other amino acids are:

other amino acid residue name	**three letter code**
aspartate (aspartic acid)	ASP
glutamate (glutamic acid)	GLU
glycine	GLY
histidine	HIS
lysine	LYS
asparagine	ASN
glutamine	GLN
arginine	ARG
serine	SER
threonine	THR
tyrosine	TYR

You might initially construct the variable `IsPhobic` as a list (of True and False values). Then, turn it into an array using the statement `IsPhobic = np.array(IsPhobic, bool).`

Part d
Add to your module a function **`RadiusOfGyration`** with the following characteristics:

```
function:
       RadiusOfGyration
arguments:
       Pos, dimension (N,3) array of atomic positions
returns:
       Rg, a float with the corresponding radius of gyration
```

The radius of gyration for a collection of N coordinates is defined by the following formula:

$$R_g^2 = \frac{1}{N}\sum_{i=1}^{N}|\mathbf{r}_i - \bar{\mathbf{r}}|^2 \qquad \text{where} \qquad \bar{\mathbf{r}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{r}_i$$

Your code should *not* involve any explicit loops (e.g., "`for i in range…`"). Instead, make use of the numpy functions `mean` and `sum`, which are much faster. Generally, you want to avoid explicit loops for your code to be more *Pythonic* and take advantage of internal speed optimizations.

Part e
Using the functions above, add module-level code that accomplishes the following tasks:

- Makes a list of all pdb files (in the working path) using the glob module.
- For each pdb file,
    1. Reads the file
    2. Computes R_g for all amino acid residues
    3. Computes $R_{g,\text{phobic}}$ for only the hydrophobic amino acid residues
    4. Computes the ratio $R_{g,\text{phobic}}/R_g$
    5. Prints out the filename, number of residues, and items 2-4, in a single line.

Hint: you can return a (M,3) array of positions of just the hydrophobic residues using array selection per **`Pos[IsPhobic]`**, where M is the number of hydrophobic residues and the number of True values in **`IsPhobic`**. Notice that **`IsPhobic`** must be an array (not a list) for this to work.

Part f
Using your favorite graphing program, make two plots using the data for all of the proteins:
- $R_{g,\text{phobic}}$ and R_g as a function of chain length (number of residues)
- the ratio $R_{g,\text{phobic}}/R_g$ as a function of chain length (number of residues)

Part g – ADVANCED TRACK

A contact between two amino acid residues can be declared when the distance between their alpha carbons is less than a certain cutoff, say 9 Å. Statistics of contacts in experimentally-determined structures have been used to extract statistical "bioinformatics" interaction potentials.

Add to your module a function **`GetContacts`** with the following characteristics:

```
function:
       GetContacts
arguments:
       Pos, dimension (N,3) array of atomic positions
returns:
       Contacts, a list of (i,j) tuples giving all residue-residue contacts in
            Pos
```

Use a CA-CA cutoff distance of 9 Å to identify a contact. You will have to consider all pairs of positions in Pos. Then, as you are parsing through structures in your code, tabulate statistics of contacts in order to calculate the following quantities:

- $f_k =$ the fraction of all amino acids in the dataset that are of type k (e.g., are an alanine)
- $c_{kl} =$ the fraction of all contacts in the dataset that involve amino acids of types k and l

With these quantities, compute "statistical" interaction potentials for contacts between amino acid types:

$$u_{kl} = -k_B T \ln\left(\frac{c_{kl}}{f_k f_l}\right)$$

Since there are 20 amino acids, there are a total of 210 possible interactions. Note that the energies as calculated above are not binding energies between amino acid contacts (i.e., the energy change for bringing two separated amino acids into contact), but rather relative ones between different contact pair types (i.e., an alanine-leucine versus a glycine-cysteine contact energy). Therefore, it is possible to globally shift the energies that you calculate since they have no absolute scale. To keep things standardized, shift all energies by the same amount such that the average u_{kl} is zero.

Make gridded contact map graph using a color scale to show the relative energies between each amino acid type, where each has its own column and row. Then underneath, list of the top 5 most favorable amino acid contact pairs and the top 5 least favorable, along with their statistical potentials. Use $T = 300\ K$ such that $k_B T \approx 0.6\ kcal/mol$. Finally, compare your results to the seminal work by Miyazawa and Jernigan [Macromolecules 18, 534 (1985)] – how good is the agreement?

Part h – ADVANCED TRACK

A more accurate estimation of effective inter-contact energies might take into account the fact that amino acids closer together on the protein chain have a higher probability of being in contact (e.g., residues separated by 10 residues are more local and likely to be in contact than those separated by 20). Consider two amino acid residues a distance n apart in sequence, where n is the number of amino acids between them in the chain plus two (i.e., the inclusive distance). How does the probability that they will be in contact vary with n?

Using the protein structures provided, calculate this probability and prepare a graph as a function of n. First do this for all amino acid types lumped together, then repeat it for hydrophobic-hydrophobic, hydrophobic-hydrophilic, and hydrophilic-hydrophilic contacts – such that your graph contains four curves. Are there any distinctions between the different contact pair types?

What to turn in

For the assignment, submit a short summary of your results, clearly indicating the problem components. In addition, attach a copy of your Python code for reference.