Tracks 1 & 3: Assignment 1 CHEM 2310

Objective: Practice and apply basic Python concepts to an example problem -- particularly reading files, processing string data, printing text, writing simple functions, performing simple numerical calculations, and making plots.

Due date: Friday, Feb. 17, 2012

Overview[1]

In this assignment, you will put together a Python program to compute the radius of gyration for all amino acids, and separately for hydrophobic amino acids, in a set of proteins available through the course website. You will also make several plots relating to the radius of gyration.

This exercise will provide an opportunity to use Python to do something nontrivial and interesting, while hopefully providing step-by-step guidance through what you need to do so that you can start learning in a hands-on way. Feel free to ask questions if you get stuck. The assignment is relatively involved, so it would be good to get started on it soon.

Recap of proteins

Proteins are life's molecular machines. They are biopolymers typically composed of chains of 20 building blocks called amino acids. Proteins are critical to biological function. One of their characteristic properties is that they typically spontaneously organize or "fold" into shapes related to their function. This folding process is typically driven by the hydrophobic effect, with hydrophobic (water-avoiding) amino acids forming a "hydrophobic core" at the center of the protein.

Many 3D structures of proteins have already been determined experimentally and are available through the Protein Data Bank (www.pdb.org). These structures typically are obtained via X-ray crystallography or NMR and contain coordinates of all of the atoms in the protein, or at least all of the non-hydrogen atoms, as well as other information such as experimental details. Structures can be freely downloaded from the PDB.

Additional details

Here, you will write a program that computes the radius of gyration, R_g, for a set of protein structures that are provided; this roughly describes the characteristic size of the object. You will compare the R_g for all amino acids with that for just the hydrophobic amino acids, and compute a typical ratio $R_{g,phobic}/R_{g,all}$. This is expected to show that hydrophobic residues are typically closer to the core of a protein.

[1] This assignment adapted from a related assignment by M. Scott Shell

To simplify the exercise, we will not consider every atom in every residue of every protein; instead, we will consider just one atom per residue, the alpha-carbon atom, which, in the files, has the name “CA”. This is the central atom of every amino acid along the protein backbone.

Here, you will need to extract the coordinates of every CA atom, and also track the corresponding residue names. Ultimately, we will determine which residues are hydrophobic versus not from looking at these residue names, and use this information and the coordinates in computing the different R_g values.

Steps in the assignment

Step 1:

Work in the HW1 directory within your Assignments directory on Lee.

Obtain protein files to work on: Copy proteins.tar.gz from /panfs/lee-nas/home/dmobley/2310/T1_3_A1 to your HW1 directory. Extract these using ‘tar -xvf proteins.tar.gz’.

Also obtain a copy of template.py from the same place and rename it Assignment1.py. This contains a skeleton of the program you need to write, including some descriptions of functions and indications of what you need to fill in.

Step 2:

Your Assignment1.py module (basically, a file) will ultimately contain several functions relating to this project. Skeletons are already present, but you need to actually write the code, as most of what’s there so far is just documentation and some hints. Start by writing the function ReadPdb, which has the following properties:

```
ARGUMENTS:

     PdbFile: The filename (string) of a pdb file to be opened

RETURNS:

     Pos: an Nx3 dimensional NumPy array containing the position of
     the alpha carbon in each of the N amino acid residues in the
     protein

     ResNames, a length-N list containing the three letter codes
     corresponding to the amino acid “residue” names
```

A set of pdb files to work on were provided in Step 1. The PDB files which are provided have already been cleaned of a variety of background information so that the main information they contain is the 3D structure -- in particular, the ATOM and TER sections which contain the information we need.

PDB files are text formatted, which simplifies extracting the data. Your function will simply need to open each PDB file, read through the data, and save coordinates and residue name for each CA atom you encounter until you get to the TER entry, which is the last entry in the file. Here is a sample excerpt of a PDB file:

```
ATOM      1  N   LYS     1       7.408 -27.931  10.066  1.00100.80
ATOM      2  CA  LYS     1       8.494 -28.577   9.334  1.00103.71
ATOM      3  C   LYS     1       9.823 -27.845   9.548  1.00 95.34
ATOM      4  O   LYS     1      10.877 -28.481   9.697  1.00 95.58
ATOM      5  CB  LYS     1       8.165 -28.653   7.838  1.00110.41
ATOM      6  CG  LYS     1       7.403 -27.446   7.310  1.00114.25
...
ATOM   5188  HE3 MET   329     -16.466   1.624 -26.351  1.00 49.50
ATOM   5189  HZ1 MET   329     -16.174  -0.708 -26.147  1.00 45.05
ATOM   5190  HZ2 MET   329     -14.806  -0.446 -25.444  1.00 45.05
ATOM   5191  HZ3 MET   329     -16.129   0.013 -24.763  1.00 45.05
TER
```

To extract the data you need, you need to know the file format here.

```
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.
```

You will only need the entries in boldface -- atom name, residue name, and x, y, and z coordinates. You will only need to save information when the atom name is CA.

As a hint, note that if you have read a line in the pdb file into the variable 'line', you could extract the atom number (columns 7-11 in the pdb format) using the following code snippet:

```
>>> atomnum = line[6:11]

>>> atomnum = int( atomnum.strip() )
```

Recall that Python numbering starts at 0, and list slices do not include the final element, so this is equivalent to extracting columns 7-11. The “strip” function removes extra characters (such as spaces) which may also be in this field.

As a starting point, you might want to begin by outlining the logical steps you need to do to get the coordinates of just the CA atoms from a pdb file and store them in the indicated formats, in English. This might begin something like: (1) Open the pdb file. (2) Read all the lines from the pdb file. (3) For every line of the file, check and see if it is a ‘CA’ atom. (4) For every ‘CA” atom, ...

Once you have the English outline written, for example on scratch paper, work on translating it to Python. Try out your thoughts in the Python interpreter before storing them into Assignment1.py. And feel free to e-mail your instructor your code with questions when you get stuck.

You should test your function on a specific PDB file once you think you have it working, and verify that it operates as expected.

Step 3:

Write the code for the function ResHydrophobic with the following characteristics:

```
ARGUMENTS:

     ResNames: a (length N) list containing three letter residue
name codes.

RETURNS:

     IsPhobic: a (length N) list containing True for every
hydrophobic residue code and False for every other residue
```

The following amino acids (and corresponding three letter codes) should be considered hydrophobic. Alanine (ALA), cysteine (CYS), phenylalanine (PHE), isoleucine (ILE), leucine (LEU), methionine (MET), proline (PRO), valine (VAL), tryptophan (TRP). Test your function on a list of residue names that you provide and verify that it operates as expected.

Step 4:

Implement a RadiusOfGyration function with the following properties:

```
ARGUMENTS:

      Pos: a Nx3 dimensional array of atomic positions

RETURNS:

      Rg: A float with the corresponding radius of gyration
```

The radius of gyration can be calculated with the following expression:

$$R_g^2 = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i-\bar{x})^2+(y_i-\bar{y})^2+(z_i-\bar{z})^2)}$$

where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are the average x, y, and z coordinates, respectively (that is, the coordinates of the center of the protein). If we use vector notation, these could also be written in more compact form (using boldface to denote vectors):

$$R_g^2 = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{r_i}-\bar{\mathbf{r}}\right|^2$$

with

$$\bar{\mathbf{r}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{r_i}$$

While in general it's a good idea to test each function independently of the others, it may be hard to test this one without your other functions.

Step 5:

Use the functions you have already written to perform the following tasks:

- List all PDB files in the working path (assuming you've put them in the working path) using the glob module ('import glob'). (An example of this is provided in the template).
    - As in standard command-line interfaces, "*" is a wildcard character, so "*.pdb" means all pdb files

- For each PDB file:
  - Read the file
  - Compute R_g over all the amino acids
  - Compute $R_{g,phobic}$ for just hydrophobic amino acids (*hint:* if you have a list of hydrophobic residues stored in IsPhobic, “indices(isPhobic)” will get you the indices of hydrophobic residues; you can then use these to access the positions of just the hydrophobic residues)
  - Store the number of residues and two Rg values to plot after you finish these calculations
- Use matplotlib (aka pylab) to make the following plots:
  - R_g and $R_{g,phobic}$ versus chain length (number of residues)
  - The ratio $R_g/R_{g,phobic}$ versus chain length

Turn in:

Turn in your plots, with axes labeled, either by e-mail or printouts in class. I will also inspect your Python code which should be stored as described on Lee.