The University of Ottawa

CHEMISTRY 8309 – 2021

Advanced Scientific Programming for Chemists

Instructor: Professor Tom Woo
Office: D'Iorio Hall 303
Lab Office: Working from home this term.
email: twoo@uottawa.ca

Credits: 1.5 credits (half semester course)

Lectures: This is mostly a reading course with lecture notes provided. A few lectures will be given by Zoom and will be arranged to fit everyone's schedules. Lectures will be announced as needed.

Website: http://titan.chem.uottawa.ca/CHMprogramming

Email Woo for password.

Office Hours: email for an appointment

Textbook: Lecture notes will be on-line.

Grading Scheme: 100% assignments.

Start Date: Week of November 1.

Aims and Goals:

The aim of the course is to make you a better scientific programmer and help you develop transferable skills that can be applied in more general contexts. Whilst most of the concepts can be applied to whatever language or tools you use, the classes will be taught using the Python programming language. Python has an expressive syntax that is very easy to pick up, but is also advanced enough that complex projects can also be handled. It is hoped that you will gain the following from this course:

- Become a better scientific programmer
- Write effective code that others can easily understand.
- Be able to use the tools available to you to make your life easier.
- Work on code effectively as part of a team.

This course serves as a guide for a number of different topics and you will have to do self-guided learning to become fully proficient. You will be expected to read and practice as necessary to complete the exercises. There is an endless quantity of great documentation out there, so you will need to learn to read around the subjects with the class notes as a guide.

Course Topics

The following topics will be covered.

- Best Practices
- Pure Functional Programming and Stateless Programs
- Object Oriented Programming
- Testing
- Debugging
- Profiling and Code Optimization
- Parallel Programming Methods
- Using a Source Revision Control System
- Project Design and Management
- Data Science Libraries and Methods

Prerequisites:

You are required to have previous programming experience in any language and have some basic knowledge about Python. This is not an introductory course to computer programming and help will not be given for basic programming issues.

If you have never coded in Python before there are many resources to help you learn. We recommend the website 'Learn Python the Hardway' at:

http://learnpythonthehardway.org

which provides a set of exercises that covers the basics and some more advanced concepts (The on-line version is free – one just needs to look a bit on the website). To prepare for this course, one should be able to complete up to exercise 39. 'Learn Python the Hardway' is an introduction to computer programming, so an experienced programmer can go through it quickly and focus on the particulars of the Python language. The exercises are best done if you type them out yourself to help you learn, and you should definitely do that rather than be tempted to copy-paste everything. There are also many resources on the web including the official documentation, which is great http://docs.python.org/index.html. This includes a tutorial and full documentation for all the standard modules. Topics which students should pay particular attention to:

- scoping of variables in Python
  - a good introduction to this can be found here.
- Passing variables in Python (i.e. by reference, by value etc.)

Precourse exercises: We have developed some Python exercises (see appendix to this course outline) that students should complete to demonstrate one's programming experience. Ideally, student's should finish the exercises before the course starts. Nevertheless, these exercises will make up part of the first assignment and will need to be completed within the first 2 weeks of the course.

Precourse Exercises

1. Write a Python code that will read a Gaussian output file from a geometry optimization run and print out (to the 'screen' rather than to a new file) the last set of Cartesian coordinates of the molecule in the following format (standard xyz format):

```
22
Number of geometries found:  15
C     0.00000000  0.00000000  0.0000000
C    -1.40389800  0.51763100 -0.28577600
C    -1.57838100  2.01469200 -0.22545400
H    -2.62851400  2.25755700 -0.29817700
H    -1.16760100  2.40342000  0.69878600
H    -1.03173400  2.48283200 -1.03289400
O    -2.31388200 -0.25095800 -0.51016300
.
.
.
```

number of atoms

this is a title line, but for this exercise, print out the number of geometries found in the output. i.e geometry optimization steps.

These are the Cartesian coordinates of each atom in Angstroms

If you are unfamiliar with the Gaussian output, below is an exerpt from the output where the Cartesian coordinates are given:

```
                         Input orientation:
 ---------------------------------------------------------------------
Center     Atomic      Atomic             Coordinates (Angstroms)
Number     Number       Type              X           Y           Z
---------------------------------------------------------------------
      1          6           0       -0.207205   -0.026856    0.039222
      2          6           0       -0.060604   -0.173270    1.548397
      3          6           0        1.283189    0.206042    2.119026
      4          1           0        1.322362   -0.072025    3.162192
      5          1           0        2.076666   -0.287255    1.570281
      6          1           0        1.442591    1.270477    2.011758
      7          8           0       -0.972592   -0.604102    2.220567
      8          7           0        0.523581    1.137387   -0.462727
      9          1           0        0.054744    2.012250   -0.317744
     10          1           0        0.857552    1.047053   -1.402604
     11          1           0        0.235871   -0.914142   -0.402929
     12          1           0       -1.267662   -0.038704   -0.191885
---------------------------------------------------------------------
```

Please note, that the coordinates as shown in the excerpt above will be printed multiple times as the program optimizes the geometry of the molecule. You are to extract only the last set of coordinates printed in the output file. Additionally, one will need to convert the atomic number into the atomic symbol. Your code should be able to do this for at least the first 55 elements of the periodic table. Sample Gaussian output files are available upon request.

Only standard Python modules should be used. e.g. '*math*', '*numpy*', or '*sys*'.

2. Modify your Python code from Exercise 1 above, to print the Cartesian coodinates in the same format except that the atoms of each element type are grouped together. So if your output from Exercise 1 was the following:

```
C     0.00000000  0.00000000  0.00000000
C    -1.40389800  0.51763100 -0.28577600
O    -1.57838100  2.01469200 -0.22545400
H    -2.62851400  2.25755700 -0.29817700
O    -1.16760100  2.40342000  0.69878600
H    -1.03173400  2.48283200 -1.03289400
C    -1.53423400  3.26573700 -0.85747700
```

the output for this exercise would be in the following where the coordinates for all carbon atoms are grouped together, followed by the coodinates for all oxygen atoms, etc..:

```
C     0.00000000  0.00000000  0.00000000
C    -1.40389800  0.51763100 -0.28577600
C    -1.53423400  3.26573700 -0.85747700
O    -1.57838100  2.01469200 -0.22545400
O    -1.16760100  2.40342000  0.69878600
H    -2.62851400  2.25755700 -0.29817700
H    -1.03173400  2.48283200 -1.03289400
```

3. Write a Python code to read the output of a Gaussian frequency calculation, to read all of the normal mode vibrational frequencies (given in cm^{-1}). Using the vibrational frequencies you code will calculate the vibrational entropy at 300 K using the following equation:

$$S_{vib} = R\sum_{i}^{vibs}\left(\frac{\gamma_i}{e^{\gamma_i}-1} - \ln\left(1-e^{-\gamma_i}\right)\right) \qquad \gamma_i = \frac{h\nu_i}{k_B T}$$

where the sum is over all normal mode vibrations. h is Plank's constant, ν_i is the frequency in s^{-1}, k_B is the Boltzmann constant, R is ideal gas-constant (use units of cal $mol^{-1}K^{-1}$. Your code should ignore imaginary frequencies (these are printed as negative vibrational frequencies in Gaussian) and warn the user how many imaginary frequencies were detected if any. Sample Gaussian output files are available upon request.

The standard Python math module will need to be used for the natural log and exponential functions.