CH40208: TOPICS IN COMPUTATIONAL CHEMISTRY

INTRODUCTION TO PYTHON

INTRODUCTION

- Aim is to give experience with computer programming in Python for computational chemistry applications
- Will build on the first and second year Python labs
- Dr Benjamin Morgan
  - Office: 1S 0.12; Email: b.j.morgan@bath.ac.uk
- Dr Andrew McCluskey
  - Office: 1S 0.15 (Wednesdays Only); Email: andrew.mccluskey@diamond.ac.uk

ASSESSMENT

- x Dec: xx:xx Multiple Choice Questions and Error Spotting exercise
  - MCQs cover all of the material up to that date
  - Error spotting should be familiar from earlier work
  - Do not spend more than 30 minutes on either
- x Dec: xx:xx Programming test
  - Up to 3 hours
- Both parts are “open book” assessments; you may consult lecture notes, etc.

ASSESSMENT

- x Dec: xx:xx M
  - MCQs cove
  - Error spotti
  - Do not spe
- x Dec: xx:xx P
  - Up to 3 hours
- Both parts are "open book" assessments; you may consult lecture notes, etc.

FIRST AND SECOND YEAR PYTHON

- Much of the first few weeks will feel like revision from first and second year
- More details and more opportunity for programming
  - Rather than filling in blanks
- If you would like to revise first or second year material, this should be available on moodle

JUPYTER NOTEBOOK

- As with the first and second year labs, we will be using Jupyter Notebooks to interact with the Python programming language
- Create a folder on your H: drive named "CH40208" then visit the JupyterHub and navigate to this folder

https://chsv-jupyter.bath.ac.uk/

VARIABLE TYPES

- *Variables* are containers used to store data
- Different types of variables exist, and define the operations that can be performed
  - Integers: whole numbers (`int`)
  - Floats: numbers with decimal points (`float`)
  - Complex: complex number (`complex`)
  - String: some text (`str`)
  - Boolean: logical information, True or False (`bool`)

VARIABLE ASSIGNMENT

- The *assignment* of the variable define the value that the container holds
- This links the variable name with some location in computer memory, and places the value there.
- This means we can then use that variable in other parts of the code

VARIABLES

DEMO

ARITHMETIC

- Python *natively* can do basic mathematical operations
  - Addition: `(a + b)`
  - Subraction: `(a - b)`
  - Multiplication: `(a * b)`
  - Division: `(a / b)`
  - Exponent: `(a ** b)`

ARITHMETIC

- Python will follow the *order of operations* that should be familiar from mathematics
  - BODMAS/BIDMAS/PIMDAS/POMDAS
  - **B**rackets
  - **O**rder
  - **D**ivide/**M**ultiply
  - **A**ddition/**S**ubtraction

ARITHMETIC

DEMO

MIXED MODE OPERATIONS

- As mentioned previously, not all variables are the same
- What happens when a mathematical operation is performed on variables of different types
  - `int` and `float`
  - `float` and `complex`
  - `float` and `str`?

MIXED MODE OPERATIONS

DEMO

OUTPUT

- Currently we are using the intrinsic functionality of the Jupyter Notebook to print the output from the last line in a given cell
- For printing not at the end of a cell, or from within a script the `print` function is necessary
- Print formatting is a useful tool in Python to make the print statements that you create easier to understand

INPUT

- In addition to the output of information, it is also of interest to read information from the user
- Python has multiple ways to receive information in (some of which will be introduced in the following weeks
- The first is the `input` function

INPUT/OUTPUT

DEMO

LOGICAL OPERATORS

- Python and Jupiter Notebook can be used as a simple calculator
- Let's make or code more intelligent!
- To do this we can use *Boolean logic*; `True` or `False` questions
- Python is able to assess the truth of a particular operation

LOGICAL OPERATORS

Some logical operators

Name	Equals	Less than	Less than or equal	Greater than	Greater than or equal	Not equal
Operator	==	<	<=	>	>=	!=

LOGICAL OPERATORS

DEMO

FLOW CONTROL

- We are then able to use this Boolean logic to *control* the path that the code will follow
- To do this we use `if` statements; these ask `if x is True`?
  - Note the the `is True` part is often implicit
- The `if` statement is often accompanied by an `else`; which is the path taken when `x is False`
- The third modifier in an `if` statement is the `elif` (short for else if); this offers an alternate path to follow

FLOW CONTROL

DEMO

MORE LOGICAL OPERATORS

- Logical operators can be extended to include those which link two statements
- These are the AND and OR operators; which are foundational to computational logic

The results of an AND operation

Input A	Input B	Logic	Output
True	False	AND	False
True	True	AND	True
False	False	AND	False

The results of an OR operation

Input A	Input B	Logic	Output
True	False	OR	True
True	True	OR	True
False	False	OR	False

MORE LOGICAL OPERATORS

DEMO

HOW TO WRITE GOOD CODE

- A lot of computer programming is about approaching the problem in the most constructive way
- In all of the exercises in this course, you will be given a *spec*; this is a description in plan English of what the code should perform
- To produce the best code, you should try and translate this into an *algorithm*; a step by step route (although not computer code) to complete the goals outlined in the spec
- The final step is then to take the algorithm and translate each individual step into the appropriate Python

PROBLEM

- In a *single* Jupyter Notebook cell, write a tool to convert from temperature in Fahrenheit to temperature in Celsius
- Consider the *algorithm* that you should employ to create useful code, **before** you start to code

$$T(^\circ\mathrm{C}) = \frac{5(T(^\circ\mathrm{F}) - 32)}{9}$$

PROBLEM

- The second problem this week involves calculating the equilibrium constant
- You need to use the logical expressions that have been introduced to control the flow of the program such that is can deal with multiple different units

$$K = \exp\left(\frac{-\Delta G}{RT}\right) = \exp\left(\frac{-\Delta g}{k_B T}\right)$$