Lecture 2: Pythonic logic and loops

Dr Benjamin J. Morgan[1] and Dr Andrew R. McCluskey[1,2]

[1] *Department of Chemistry, University of Bath, email: b.j.morgan@bath.ac.uk*
[2] *Diamond Light Source, email: andrew.mccluskey@diamond.ac.uk*

September 3, 2019

Aim

In this lecture, you will learn about logical operations, conditional statements, and for and while loops.

1 Loops

One of the best uses of programming (and computers) is to perform repetitive task over and over. For this we use *loops*, within Python there are two common types of loop:

- `for` loops iterate over a given sequence.
- `while` loops repeat as long as a certain logical operation is `True`.

An example of each of a `for` and `while` loop is shown below, both perform the same function,

```
# For loop

for i in range(5):
    print(i)

i = 0
while i < 5:
    print(i)
    i = i + 1
```

Both of these code blocks will print the numbers 0 to 4, however the `for` loop is clearly more concise. Additionally, the `while` loop is more prone to accidently running an *infinite*. If you were to forget to manually iterate the variable `i` (this is the line `i = i + 1`), then the `while` condition would always be `True` and therefore the code would run forever within this loop. For this reason it is suggested that, where possible, you use a `for` loop over a `while` loop.

The `for` loop will iterate the variable (in the example above this variable is named `i`) through whatever sequence is given (this is `range(5)` above, which is equivalent to the *list* `[0, 1, 2, 3, 4]`). The sequence does not necessarily have to be a `range` command, it may be any `list` or `numpy.ndarray` (we will discuss these types later in the course). For example, in the code below we iterate though the first ten chemical element symbols,

```
# Printing the periodic table

elements = ["H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne"]

for symbol in elements:
   print(symbol)

for i, symbol in enumerate(elements):
   print("The index of the list for {} is {}.".format(symbol, i)).
```

It is possible to use the `enumerate` command to count through the list during the loop, as shown in the second example above.

Exercise

- Recall from first and second year, that Python counts indices in a list from 0. How could the above code be adapted such that the correct atomic number will be printed?

1.1 Escaping loops

Sometimes it is computationally efficient to leave a `for` loop, to skip a particular value, under a certain condition. For this, the commands `break` and `continue` are available. The `break` command will exit the *inner-most* loop that is being carried out, while the `continue` command will skip the current value and jump immediately to the next. Examples of how these may be used are shown below, where the `len` function will return the *length* of the list,

```
# Finding the zero in a list

numbers = [1, 5, 7, 0, 2, 6, 2]
for i in range(len(numbers)):
   if numbers[i] == 0:
      break

print("The zero is at index {}.".format(i))

# Making all the negative values positive

numbers = [-2, 4, 1, -5, 2, 6, -3, -4]
for i in range(len(numbers)):
   if numbers[i] >= 0:
      continue
   else:
      numbers[i] = numbers[i] * -1
```

Note that the above examples are toy problems and there are more efficient way to carry-out these specific operations in Python.

2 Problem

2.1 Equilibrium constants

Write code that will calculate values of the equilibrium constant, K, for a given free-energy change over a range of temperatures. The program should ask the user for a free-energy value, ΔG or Δg, and to specify the units for this (either $\mathrm{kJ\,mol^{-1}}$, eV, or J). The initial temperature, T_{init}, final temperature, T_{final}, and temperature step size, T_{step} should also be entered by the user (in K). In order to learn more about how to do this with the `range` function, check the documentation online (`https://www.w3schools.com/python/ref_func_range.asp`). The equilibrium constant equation is,

$$K = \exp\left(\frac{-\Delta G}{RT}\right) = \exp\left(\frac{-\Delta g}{k_B T}\right) \tag{1}$$

where, $R = 8.314\,\mathrm{J\,K^{-1}\,mol^{-1}}$, $k_B = 1.3806 \times 10^{-23}\,\mathrm{J}$, and $1\,\mathrm{eV} = 96.485\,\mathrm{kJ\,mol^{-1}}$.

When you check for what is typed, don't forget to check for upper-case as well as lower-case letters, as these characters have different ascii codes. You should also anticipate the possibility of the user entering a completely different letter (by mistake): what action would be appropriate in this event? Additionally, make sure that the user cannot make the temperature *unphysical* (e.g. less than or equal to zero). Again, remember to plan before you code.

Test the code using a temperature range from 100 K to 2000 K with a step size of 100 K, and with free energies of:

1. $-12.177\,\mathrm{kJ\,mol^{-1}}$
2. $-0.1452\,\mathrm{eV}$
3. $-2.6308 \times 10^{-20}\,\mathrm{J}$

Comment on the values at 300 K.