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*

July 1, 2019

Aim

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

1 Logical operators

Previously, we observed how to use Python or a Jupyter Notebook as a simple calculator. However, programs become more useful when we are able to make the program more "intelligent" allowing it to perform different calculations under different circumstances. This ability relies heavily on the use of logical operators, as we shall see. A logical operator is code that returns a Boolean, either `True` or `False`. The logical operator `==` is one of the most common, and translates to *is equal to*, this operator will return `True` if the values on either side are the same, for example,

```
# The truth

print(14.0/2.0 == 7.0)
print(13.0/2.0 == 7.0)
```

This code will return `True` then `False`.

The equals operator is only one of many logical operators that is available in Python, some are given in Table 1. Each of these may be used in the same syntax as the equals operator.

1.0.1 Exercise

- Write code that will return the result of the following logical operations:
  1. $1 = 4$
  2. $10 < 15$
  3. $3.1415 \neq 3$

Table 1: Some logical operators available in Python.

Name	Mathematical Symbol	Operator
Equals	$=$	`==`
Less than	$<$	`<`
Less than or equal	$\leq$	`<=`
Greater than	$>$	`>`
Greater than or equal to	$\geq$	`>=`
Not equal	$\neq$	`!=`

2 if, elif, and else flow control

The `if` statement is one of the simplest, and most powerful, opeartions that Python can perform. This allows the code to apply different operations *if* certain criteria are `True`. An example of a Pythonic *if statement* is shown below,

```
# The if operator

if prior_meetings == 0:
    print("Hello World!")
```

Note that in Python the indentation is incredibly important (it is how the interpreter determines what is and is not part of the *if statement*. The above code asks the question, does the variable `prior_meetings` has the value `0`, and if it does print the string `Hello World!`

The `if` statement may be used in a more extended context, such as if the *logical argument* in the if statement is not True, an `else` can be included (or even an `elif`), as shown below,

```
# Five greetings to an old friend

if prior_meetings == 0:
    print("Hello World!")
elif prior_meetings == 1:
    print("Oh! You again")
elif prior_meetings == 2:
    print("Oh! You again")
elif prior_meetings == 3:
    print("Oh! You again")
elif prior_meetings == 4:
    print("Oh! You again")
elif prior_meetings == 5:
    print("Oh! You again")
elif prior_meetings > 5:
   print("Hello old friend!")
else:
   print("Meetings should be a positive number or zero")
```

3 AND and OR operators

In addition to the logical operators introduced above there are some others that it is important, and useful to be aware of. These are the `and` and `or` operators, which have important *but different* actions:

- The `and` operation returns `True` if both operations are true.
- The `or` operation returns `True` if either operations are true.

The code below gives and example of the syntax for these operations,

```
# Using and and or

if 4 > 3 and 3 > 2:
   print("This is true")
elif 4 > 3 or 3 > 3:
   print("This is also true")
else:
   print("Nothing is true")
```

3.0.1 Exercise

- Use the `or` operator to reduce the `prior_meetings` code above to just 8 lines long. **Hint**: think about other operators from Table 1 to reduce the length of the code.

4 for loops