Comparison Operators in Python!

These operators will allow us to compare variables and output a Boolean value (True or False).

If you have any sort of background in Math, these operators should be very straight forward.

Table of Comparison Operators

In the table below, let a=3 and b=4.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Chained Comparison Operators

An interesting feature of Python is the ability to *chain* multiple comparisons to perform a more complex test. You can use these chained comparisons as shorthand for larger Boolean Expressions.

Few examples are given below…

In [1]:
```
1 < 2 < 3
```

Out[1]: True

The above statement checks if 1 was less than 2 **and** if 2 was less than 3. We could have written this using an **and** statement in Python:

In [2]:
```
1<2 and 2<3
```

Out[2]: True

The **and** is used to make sure two checks have to be true in order for the total check to be true. Let's see another example:

In [3]:
```
1 < 3 > 2
```

Out[3]: True

The above checks if 3 is larger than both of the other numbers, so you could use **and** to rewrite it as:

In [4]:
```
1<3 and 3>2
```

Out[4]: True

It's important to note that Python is checking both instances of the comparisons. We can also use **or** to write comparisons in Python. For example:

In [5]:
```
1==2 or 2<3
```

Out[5]:
```
True
```

Note how it was true; this is because with the **or** operator, we only need one *or* the other to be true. Let's see one more example to drive this home:

In [6]:
```
1==1 or 100==1
```

Out[6]:
```
True
```