8 Operators

- There are a variety of **operators** that can be used to manipulate numeric and String data. You've already seen the concatenation (+) operator for putting two strings together, as well as the multiplication (*) operator for multiplying two numbers. We'll explore each category of operators in turn.

1. The **mathematical operators** are applied to numbers. The most common ones are:

+	addition	*	multiplication	/	division
−	subtraction	**	power	%	modulo

→ Write a new program that asks the user to enter a number, and then apply each of the mathematical operators to that number with a constant (e.g., 10). Output all the results in a neat and orderly list, something like this:

```
please enter a number: 5
you entered 5
the mathematical operators applied to your number and 10 are:
5 + 10 = 15
5 - 10 = -5
5 * 10 = 50
5 ^ 10 = 9765625
5 / 10 = 0.5
5 % 10 = 5
```

2. The **relational operators** include **equality** operators and **comparison** operators. The equality operators can be used to evaluate whether any two values are the same as each other (or not). The comparison operators can be used with numeric values, as well as with Strings. Both categories of relational operators are listed below:

==	equality	>	greater than	<	less than
!=	inequality	>=	greater than or equal to	<=	Less than or equal to

The result of applying a relational operator to two values is either **True** or **False**, i.e., Boolean values. For example, if you run the following code:

```
x = 5
y = 7
print ( x < y )
```

the output would say **True**.

→ Write a new program that asks the user to enter two numbers. Then compare the two numbers using all 6 relational operators (listed above) and output all the results in a neat and orderly list, something like this:

```
please enter a number: 2
please enter another number: 8
2 == 8 : False
2 != 8 : True
2 > 8  : False
2 >= 8 : False
2 < 8  : True
2 <= 8 : True
```

→ **Challenge question:** Create a new version of the programme and ask the user to enter two words (i.e., Strings) instead of two numbers.

How much code do you have to change in order to make it work with a different data type?

3. The **Boolean operators** are applied to Boolean values (**True, False**), and there are three of them: **and**, **or** and **not**. Their values are de ned below, assuming that A and B are Booleans:

A	B	not A	A and B	A or B
True	True	False	True	True
True	False		False	True
False	True	True	False	True
False	False		False	False

→ Write a new program that defines two Boolean variables, A and B. Initialise A to True and B to False. Output the following:

(1) the values of each variable;
(2) the results of applying the not operator to each variable; and
(3) the results of applying the **and** and **or** operators to all pairwise combinations of the variables, including themselves.

Make sure your output is neat and orderly, something like this:

```
| A = True    | not A = False  |
| B = False   | not B = True   |
| A and B = False  | A and A = True  | B and B = False  |
| A or B = True    | A or A = True   | B or B = False   |
```

9 Random numbers

It is often useful to be able to ask the computer to generate a random number. In other words, instead of initialising a variable to a particular value, e.g.:

y = 10

we might want to initialise a variable to a number within a range of values, e.g.:

y = value between 0 and 10

For example, if you are simulating rolling a die, then you would want to be able generate a random integer between 1 and 6.

The simplest way to generate a random number in Python is to invoke the random() function, which is defined in the random module. In order to use this function, you need to import the module and then you can call the function.

```python
import random
n = random.random()
print 'n = ' + str( n )
```

→ Write a new program like the one above and run it several times. Run your program 10 times. What do you notice about the values being returned?

If you look at the documentation for this function, you'll see that it returns: "the next random floating-point number in the range [0.0, 1.0)"
(from https://docs.python.org/2/library/random.html#module-random), in other words, a number within the half-closed interval between 0:0 and 1:0, meaning that the value returned is
$0.0 \leq n \leq 1.0$.

→ Modify your program so that it prints a value between 2 and 12, e.g., simulating rolling two dice. You need to make sure of three things:

(1) your output is an *integer*;
(2) your output is greater than or equal to 2; and
(3) your output is less than or equal to 12.

→ Modify your program so that it prints a value between 0 and 51, e.g., simulating drawing a card from a deck of playing cards.

Some hints:

- the int() function will truncate its argument (down)
- the round() function will round off its argument (up)
- the random.randint(a,b) function will return a random integer in the closed interval [a, b]