Comparisons

In school you learn that the equals sign, `=`, means that the things to the left and right of the symbol have the same value. Then you learn that there are greater than and less than signs (`>` and `<` respectively). Programming languages also have these comparisons. However we do not use the `=` for comparison because we use it for assigning values. Instead we use `==` to see if two items/objects have the same value.

```
print 1 == 1.0
print -2 == 1
print "cat" == "dog"
```

```
True
```

```
False
```

`True` and `False` are the only two values in the datatype Boolean. Most people call it "bool". They can be saved to a variable like any other datatype. Less than and greater than are pretty easy:

```
print 1 < 4
print 2 > 7
print "a" < "b"
print "b" < "a"
```

`True False True False` are the respective results for each comparison.

Finally, we have `!=` which means "does not equal"

```
print 1 != 2
print 1 != 1.0
```

```
True False
```

Less than or equal to and greater than or equal to

As you might have noticed, the standard keyboard does not have the "less than or equal to" or the "greater than or equal to" symbol. Instead, python uses `<=` and `>=` respectively.

```
print 1 < 2
print "apple" <= "banana"
print 30 >= 31
print 12345 <= 12345
```

`True True False True` ## Comparing different datatypes

I'm sure `1 < 3.2` makes sense, because you are comparing a number to another number. But what about `"apple" < "cat"` or `3 < "3"` or `2+2 == 'fish'`. All

of that should be errors, right? There is no way that any of this should work. Actually it does!

Comparing Strings

When we have a list of name, we like to have an order to it. Reverse alphabetical by last name was a favorite in my geometry class. In alphabetical ordering, A comes first following by B followed by C and so on. One could say A is less than B if they have numerical values. Actually that is true. For the comparison `"apple" < "cat"`, Python returns `True`.

In fact, all the comparisons with strings above work and will not throw an error. This goes back to datatypes and how everything is stored as binary in memory. When ASCII was made, there was logic to the binary assigned allowing them to have the fact that `"a" < "b"` evaluate to `True` work.

!

The not character, `!`, allows you to do the opposite of your result.

```
print !(1 <2)
```

```
False
```

This is useful when you are trying to say when a certain thing does not happen.