BOOLEAN ALGEBRA AND COMPARISON OPERATORS

One would be in pretty deep trouble if one couldn't tell the difference between what's small and big, what's true and false. As any other language, Erlang has ways to let you use boolean operations and to compare items.

Boolean algebra is dirt simple:

```
1> true and false.
false
2> false or true.
true
3> true xor false.
true
4> not false.
true
5> not (true and true).
false
```

Note: the boolean operators `and` and `or` will always evaluate arguments on both sides of the operator. If you want to have the short-circuit operators (which will only evaluate the right-side argument if it needs to), use `andalso` and `orelse`.

Testing for equality or inequality is also dirt simple, but has slightly different symbols from those you see in many other languages:

```
6> 5 =:= 5.
true
7> 1 =:= 0.
false
8> 1 =/= 0.
true
9> 5 =:= 5.0.
false
10> 5 == 5.0.
true
11> 5 /= 5.0.
false
```

First of all, if your usual language uses `==` and `!=` to test for and against equality, Erlang uses `=:=` and `=/=`. The three last expressions (lines 9 to 11) also introduce us to a pitfall: Erlang won't care about floats and integers in arithmetic, but will do so when comparing them. No worry though, because the `==` and `/=` operators are there to help you in these cases. This is important to remember whether you want exact equality or not.

Other operators for comparisons are `<` (less than), `>` (greater than), `>=` (greater than or equal to) and `=<` (less than or equal to). That last one is backwards (in my opinion) and is the source of many syntax errors in my code. Keep an eye on that `=<`.

```
12> 1 < 2.
true
13> 1 < 1.
false
14> 1 >= 1.
true
15> 1 =< 1.
true
```

What happens when doing `5 + llama` or `5 == true`? There's no better way to know than trying it and subsequently getting scared by error messages!

```
12> 5 + llama.
** exception error: bad argument in an arithmetic
expression
in operator  +/2
called as 5 + llama
```

Welp! Erlang doesn't really like you misusing some of its fundamental types! The emulator returns a nice error message here. It tells us it doesn't like one of the two arguments used around the `+` operator!

Erlang getting mad at you for wrong types is not always true though:

```
13> 5 =:= true.
false
```

Why does it refuse different types in some operations but not others? While Erlang doesn't let you *add* anything with everything, it will let you *compare* them. This is because the creators of Erlang thought pragmaticism beats theory and decided it would be great to be able to simply write things like general sorting algorithms that could order any term. It's there to make your life simpler and can do so the vast majority of the time.

There is one last thing to keep in mind when doing boolean algebra and comparisons:

```
14> 0 == false.
false
15> 1 < false.
true
```

Chances are you're pulling your hair if you come from procedural languages or most object-oriented languages. Line 14 should evaluate to `true` and line 15 to `false`! After all, false means 0 and true is anything else!

Except in Erlang. Because I lied to you. Yes, I did that. Shame on me.

Erlang has no such things as boolean `true` and `false`. The terms true and false are atoms, but they are integrated well enough into the language you shouldn't have a problem with that as long as you don't expect false and true to mean anything but false and true.

Note: The correct ordering of each element in a comparison is the following:
`number < atom < reference < fun < port < pid < tuple < list < bit string`
You don't know all these types of things yet, but you will get to know them through the book. Just remember that this is why you can compare anything with anything! To quote Joe Armstrong, one of the creators of Erlang: "The actual order is not important - but that a total ordering is well defined is important."

Source : http://learnyousomeerlang.com/starting-out-for-real#numbers