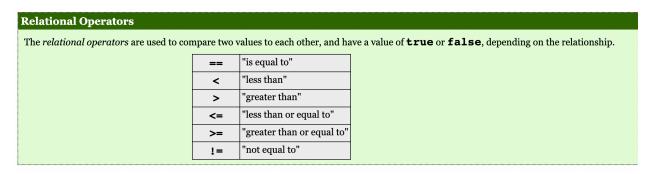
Lesson 2.2 Comparing Numbers

We'll often want to perform operations conditionally based on comparing two numerical values. With integer values, this is easily done using one of the six *relational operators*.

Relational Operators



Example:

```
Scanner in = new Scanner(System.in);
System.out.print("Enter your age: ");
double age = in.nextDouble();
if (age >= 18)
{
    System.out.println("You can vote!");
}
else
{
    System.out.println("You can't vote (yet).");
}
```

Logical Operators

In addition to the *relational operators* that compare two values there are *logical operators* that allow you to form more complex logical relationships.

The logical operators produce a value of true or false based on evaluating boolean values. | Logical operators produce a value of true or false based on evaluating boolean values. | "and": true if both expressions are true | "or": true if either expression is true | "not": true if the expression is false, and false if the expression is true

```
AND Example:

if (age >= 13 && age < 20)
    System.out.println("You're a teenager.");</pre>
```

OR Example:

```
if (age < 0 || age > 110)
    System.out.println("I don't think I believe you.");
```

NOT Example:

```
System.out.print("Enter a positive number");
double value = in.nextDouble();
if (!(value > 0))
    System.out.println("ERROR: Positive number expected");
```

You might notice that in the NOT example above we could have re-written the if statement so that it doesn't require a negation:

```
if (value <= 0)...
```

While this is correct, and you may even choose to write it without the ! in there, there will be occasions where using the *not* operator will allow your code to make more sense, and be more readable to you and others.

Common Mistakes

```
Common mistakes: you can't say if (0 < age < 21)...

and you can't say if (roll == 7 | 11)...

Each individual Boolean expression has be complete:
```

```
if (0 < age && age < 21)....
if (roll == 7 || roll == 11)...</pre>
```