How to approach Booleans and logic expressions with Diagrams!

To predict to which Boolean value an expression evaluates to (**either True or False**), you can break down the process by evaluating the smaller expressions that make the larger one. Let's solve **the example in the diagrams below.**

First: we will use ***Truth tables***, so let's learn a little bit about them...

The Boolean value (**True of False**) of an expression that uses logic operators (**and, or, not**) is determined by Truth tables. They determine which Boolean value corresponds to logic operations that use **and, or, not**.

Below are the truth tables for the operators **and** (**Logical conjunction**), **or** (**Logical Disjunction**) and **not (Logical negation)** . These tables determine which Boolean will result from more complex expressions that use logic operators.

For example, two expressions (p and q, which are generic names for expressions in logic) joined by "and" will evaluate to True only if they are both True. Otherwise, they will evaluate to False (we will see an example of this in the diagrams below)

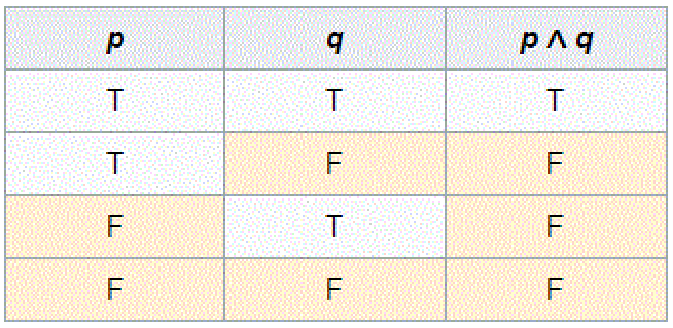
These are the Truth Tables we will need:

=> With the **and** operator, if both expressions are True, joining them with the and operator will evaluate to True. If one or both of them are False, it will evaluate to False.

=> With the **or** operator, if either one of the expressions is True, it will evaluate to True, otherwise, if they are both False, it will evaluate to False.

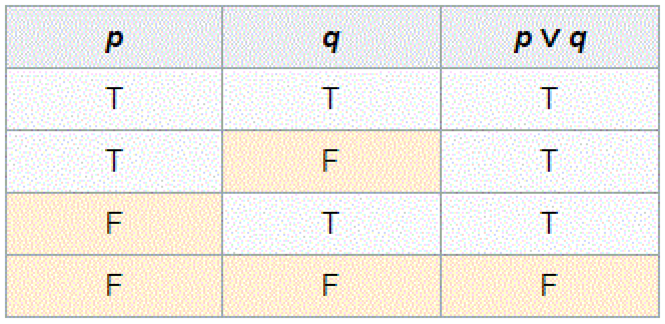
=> With the **not** operator, the expression will evaluate to the opposite Boolean that's next to the operator not. (**not True** evaluate to **False**) and (**not False** evaluates to **True**)

Logical Conjunction



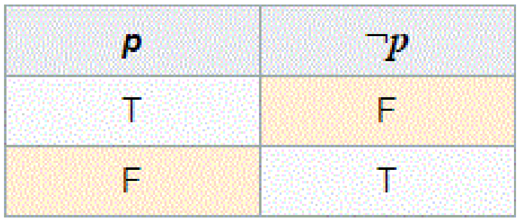
p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Logical Disjunction



p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Logical Negation



p	$\neg p$
T	F
F	T

Source: Wikipedia Truth table

EXAMPLE:

- ***FIRST STEP:*** In this example, we first start analyzing the expressions inside the innermost parentheses. This expression can be broken down into three smaller evaluations. First we analyze 5 > 6, 100 < 3 which are both **False**. Then, we analyze their conjunction (which Boolean is returned when two False statements are joined by "and"). In this case, it's False. So the expression in the innermost parentheses will evaluate to **False**, and in the next steps, we will stop thinking of the elements inside these parentheses as an expression and start thinking of it as the Boolean **False** for subsequent evaluations.
- ***SECOND STEP:***: Now we analyze the expression inside the next parentheses. This is comprised of two evaluations. First, 7<2 which is **False**. Second, **False or False** . According to our truth table, two **False** statements joined by the "or" operator will evaluate to **False.**
- ***THIRD STEP:*** Now we analyze the third parentheses, and there is only one evaluation to make, logical negation. The operator "not" followed by a Boolean evaluates to the opposite Boolean. If an expression is *True*, **not** followed by that expression will evaluate to *False*, and if it's *False*, it will evaluate to *True*.

So, in this case, we have a **not** followed by an expression that we determined evaluates to **False.** This turns into **True**.

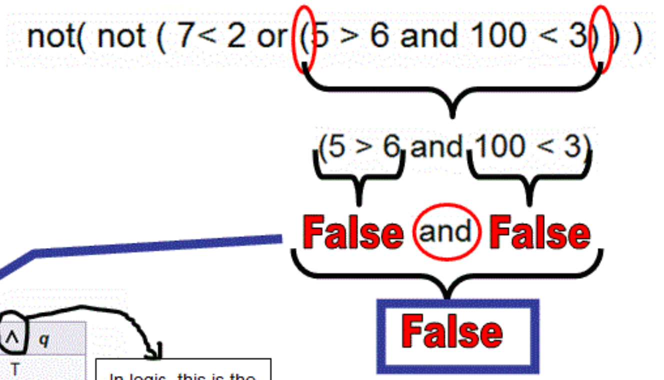
And finally, we have the same case again, **not** next to an expression that evaluates to **True** turns it into **False**, and this is the final result of evaluating this expression.

By: kiara-elizabeth

not(not (7< 2 or (5 > 6 and 100 < 3)))

1st Step

Start with the innermost parentheses and work your way out



The first parentheses evaluates to **False**

Logical Conjunction

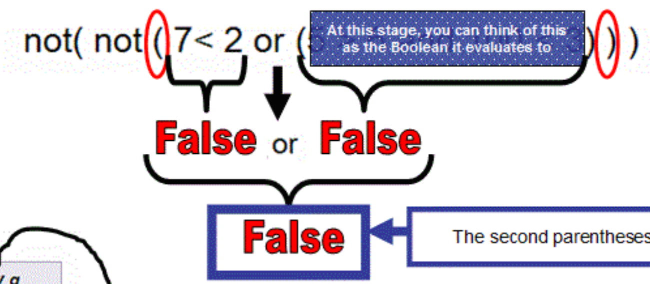
p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

In logic, this is the symbol for **and** (^)

If two expressions are **False**, their logical Conjunction **(and)** is False

By: kiara-elizabeth

2nd Step



not(not (7< 2 or (At this stage, you can think of this as the Boolean it evaluates to)))
False or False
False

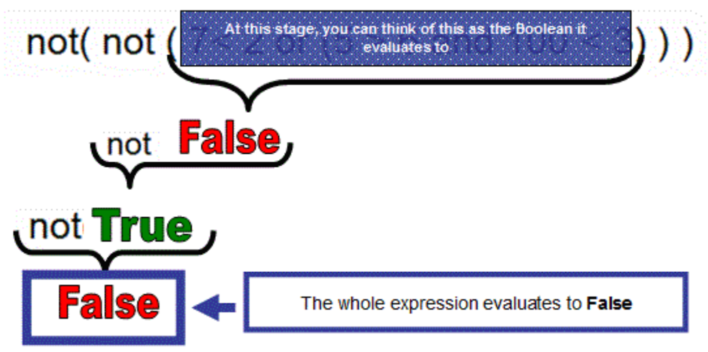
The second parentheses evaluates to **False**

Logical Disjunction

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

In logic, this is the symbol for **or** (v)

3rd Step



not(not (At this stage, you can think of this as the Boolean it evaluates to)))
not False
not True
False

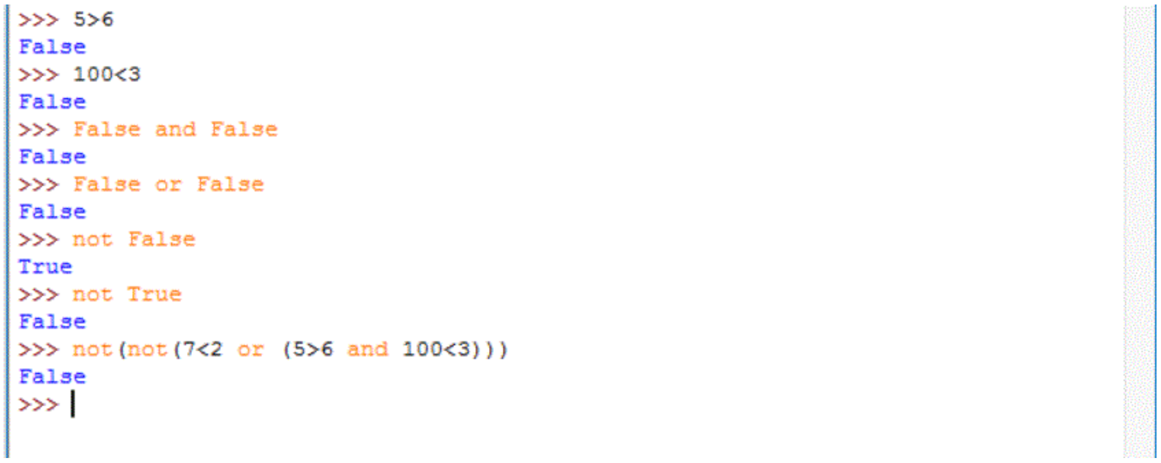
The whole expression evaluates to **False**

Logical Negation

p	¬p
T	F
F	T

In logic, this is the symbol for **not** (¬)

Here you can see this broken down by steps executed in python's shell and finally the whole expression to confirm it evaluates to False:



```
>>> 5>6
False
>>> 100<3
False
>>> False and False
False
>>> False or False
False
>>> not False
True
>>> not True
False
>>> not(not(7<2 or (5>6 and 100<3)))
False
>>> 
```

Hope it helps! If you have any questions, please ask them on the forums!

Estefania (kiara-elizabeth).