Introduction

Using relational operators and logic operators, we can build boolean expressions, which allow us to ask questions within our program – is the balance greater than 0? is the age less than 21? is the user already registered? These expressions can be used in if statements and while loops in order to execute portions of code if the statement is true.

Information

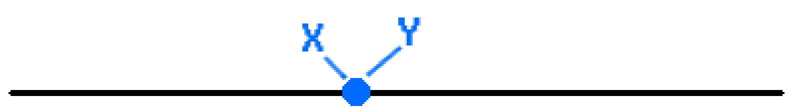
Boolean Expressions

A boolean expression is an expression that results in either true or false. For example:

- Is x greater than y?
- Is grade less than 70?
- Are width and length equal, and is length and height equal?

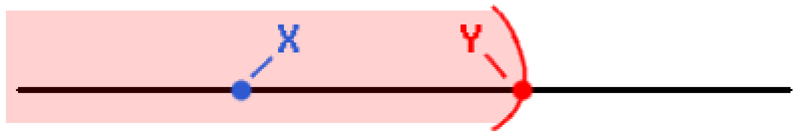
To write these statements in code, we use the **relational operators.** To ask more than one question at a time, we combine boolean expressions with **logic operators.**

Visual Representations

Boolean Expression	Representation
`x == y` is x and y equal?	
`x != y` is x and y not equal?	
`x <= y` is x less than or equal to y?	

`x < y`

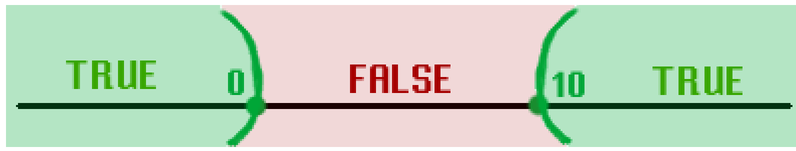
is x less than to y?



`X < 0 || x > 10`

True: x is less than 0, or greater than 10, exclusive.

False: x is between 0 and 10, inclusive.

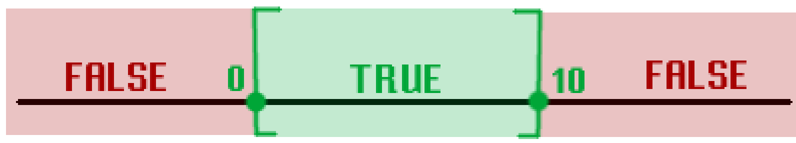


An OR is the appropriate way to represent this. AND is incorrect, as x cannot be both less than 0 AND greater than 10.

`X >= 0 && x <= 10`

True: x is between 0 and 10, inclusive.

False: x is less than 0, or greater than 10, exclusive



Truth Tables

We can use **truth tables** to diagram out all possible results of a conditional statement. For example:

X >= 0
True
False

In this case, we're asking the question, “is X greater than or equal to 0?” and there can only be two outcomes: true or false. If X happens to be 5, then the result will be true. If X is -2, then it will be false. But the specific values of X we don't care about, we only care about the states: True or False will be returned from the conditional statement.

From there, we can analyze more complex statements by viewing all possible states. If we have two conditions: **X >= 0** AND **X <= 0**, then there can be four possible states:

X >= 0	**X <= 0**
True	True
True	False
False	True
False	False

They can both be true, they can both be false, or we could have some combination of one true and one false. In the table above, we're analyzing these two statements independently of each other: X >= 0 isn't going to affect the result of X <= 0, but if we're looking at both of them together, we can at least diagram out all possible “states”.

From this, we can fill out a full truth table for a more complex statement, like asking if:

X >= 0 AND X <= 0

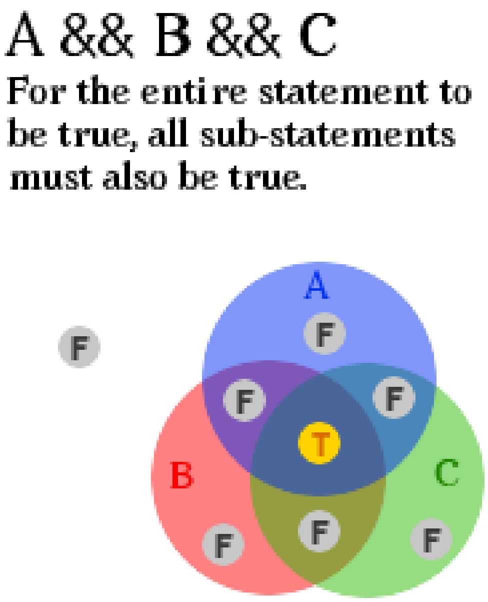
X >= 0	**X <= 0**	**X >= 0 AND X <= 0**
True	True	True
True	False	False
False	True	False
False	False	False

The only way we get the result of TRUE is if both sub-statements are true.

We can represent simple AND and OR statements in a more generic sense, using “A” and “B” as questions:

A	**B**	**A and B**
True	True	True
True	False	False
False	True	False
False	False	False

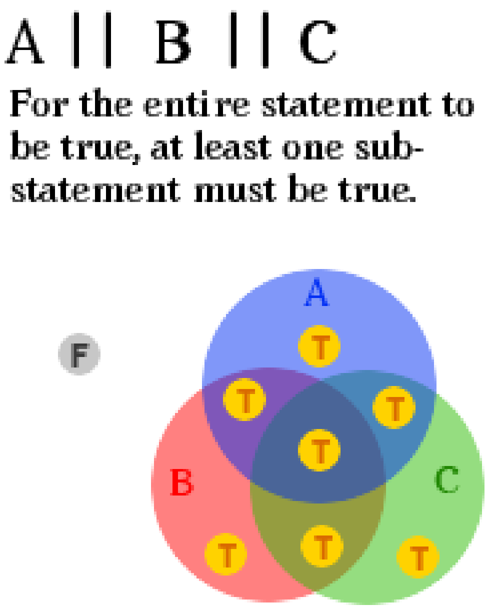
A	**B**	**A or B**
True	True	True
True	False	True
False	True	True
False	False	False



AND statements

A	**B**	**A and B**
True	True	True
True	False	False
False	True	False
False	False	False

- If <u>all</u> sub-questions are true, then the entire statement is true.
- If <u>any</u> sub-questions are false, then the entire statement is false.



OR statements

A	**B**	**A or B**
True	True	True
True	False	True
False	True	True
False	False	False

- If <u>any</u> sub-questions are true, then the entire statement is true.
- If <u>all</u> sub-questions are false, then the entire statement is false.

Lab Questions

Fill in the following truth tables

1.

We don't care about the **values** of A, B, or C, we are just asking the question "Is A less than B, AND is B less than C?".

A < B	B < C	A < B && B < C
T		
T		
F		
F		

2.

Here, think of "StateIsKansas" and "StateIsMissouri" as boolean variables, which can be assigned to either "true" or "false".

StateIsKansas	StateIsMissouri	StateIsKansas OR StateIsMissouri
T		
T		
F		
F		

3.

A	B	C	A && B	(A && B) || C
T	*T*	*T*		
T	*T*	*F*	***T***	***T***
T	*F*	*T*		
T	*F*	*F*		
F	*T*	*T*		
F	*T*	*F*		
F	*F*	*T*	***F***	***T***
F	*F*	*F*		