1. What is the Boolean data type's two values? How do you go about writing them?

**Ans:** Boolean data type is a data type that has one of two possible values (usually denoted true and false) which is intended to represent the two truth values of logic and Boolean algebra.

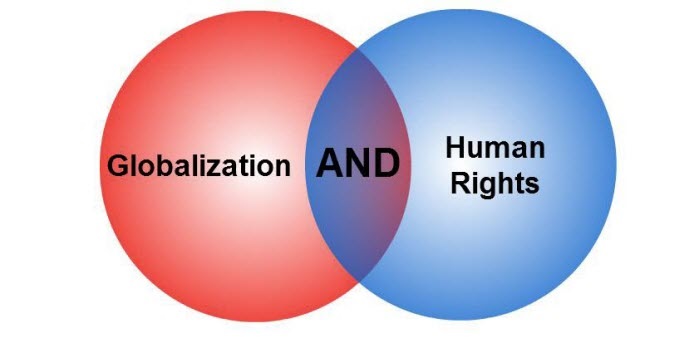
A Boolean data type is declared with the bool keyword and can only take the values true or false. When the value is returned, true = 1 and false = 0.

True and False, using capital T and F, with the rest of the word in lowercase.

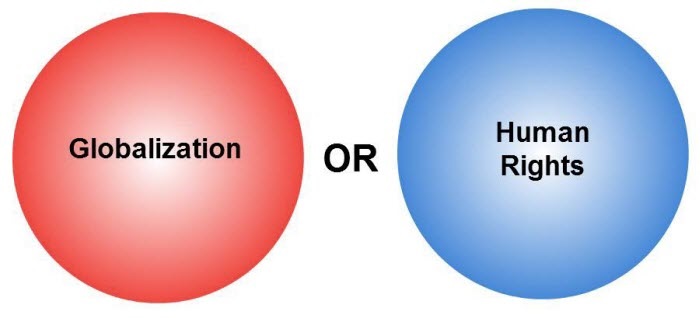
1. What are the three different types of Boolean operators?

**Ans:**  AND, OR and NOT are the three types of Boolean operators.

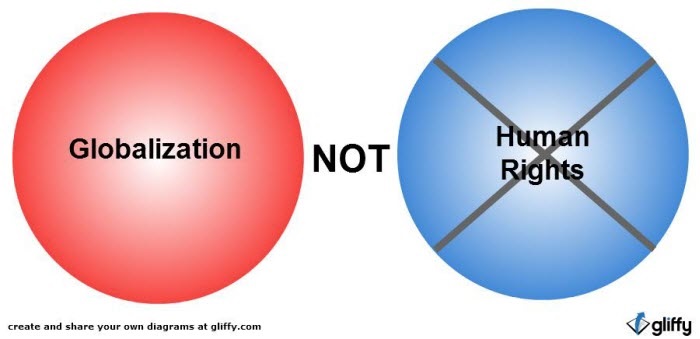
The Boolean Operators use the words AND, OR, NOT to combine keywords and thus broaden or narrow your search results. Here are some examples of these operators:



Using the Boolean Operator AND will narrow your search results. In this case, using AND retrieve search results containing both keywords globalization and human rights.



Using the Boolean Operator OR will broaden your search results. In this case, using OR will retrieve search results containing either the keywords globalization or human rights.



Using the Boolean Operator NOT will narrow your search results. In this case, using NOT will retrieve search results containing the keyword globalization but will not retrieve search results containing the keyword human rights.

1. Make a list of each Boolean operator's truth tables (i.e., every possible combination of Boolean values for the operator and what it evaluates).

**Ans: 1. and** – returns **True** only if both operands are true. In any other case,**False** will be returned. For example, the following expression will evaluate to True: **5 < 7 and 5 > 3**, because 5 is indeed less than 7 and greater than 3. Here is the **and operator’s truth table** (a table that lists all the possible inputs and the results for the logical operators):

|  |  |  |
| --- | --- | --- |
| **and operator’s truth table** | | |
| **First operand** | **Second operand** | **Result** |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

 Here are a couple of examples:

>>> 5 < 7 and 5 > 3

True

>>> 3 > 3 and 55 > 30

False

>>> 15 / 3 >= 200 and 3 == 3

False

>>> 55 == 55 and 3 <= 3

True

**2. or** – returns **True** when one or both operands are true. For example, the expression **5 < 3 or 3 == 3** will return **True** because the second operand (**3 == 3**) evaluates to **True**. Only if both operands are false will **False** be returned. The truth table for this operator looks like this:

|  |  |  |
| --- | --- | --- |
| **or operator’s truth table** | | |
| **First operand** | **Second operand** | **Result** |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

Examples:

>>> 3 == 3 or 5 < 3

True

>>> 12 <= 1 or 5 < 1

False

>>> 20 + 3 >= 23 or 5 != 5

True

**3. not** – negates the truth value of a single operand. In other words, **True** becomes**False** and vice versa. The truth table here is smaller because only a single operand is used:

|  |  |  |
| --- | --- | --- |
| **not operator’s truth table** | | |
| **Operand** | **Result** |  |
| True | False |  |
| False | True |  |

 Examples:

>>> not True

False

>>> not False

True

>>> not (5 > 3 and 5 > 2)

False

>>> not (5 > 3 and 5 < 33)

False

>>> not (5 < 3 and 5 < 33)

True

1. What are the values of the following expressions?

(5 > 4) and (3 == 5)

not (5 > 4)

(5 > 4) or (3 == 5)

not ((5 > 4) or (3 == 5))

(True and True) and (True == False)

(not False) or (not True)

**Ans:** (5 > 4) and (3 == 5): false

not (5 > 4): false

(5 > 4) or (3 == 5): true

not ((5 > 4) or (3 == 5)): false

(True and True) and (True == False): false

(not False) or (not True): true

1. What are the six different types of reference operators?

**Ans**: Reference operators are used to compare values. It returns either True or False according to the condition.

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| > | Greater than - True if left operand is greater than the right | x > y |
| < | Less than - True if left operand is less than the right | x < y |
| == | Equal to - True if both operands are equal | x == y |
| != | Not equal to - True if operands are not equal | x != y |
| >= | Greater than or equal to - True if left operand is greater than or equal to the right | x >= y |
| <= | Less than or equal to - True if left operand is less than or equal to the right | x <= y |

1. How do you tell the difference between the equal to and assignment operators?

**Ans:** == is the equal to operator that compares two values and evaluates to a Boolean, while = is the assignment operator that stores a value in a variable.

1. Describe a condition and when you would use one.

**Ans:** A condition is an expression used in a flow control statement that evaluates to a Boolean value.

1. Recognize the following three blocks in this code:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

**Ans:** The three blocks are everything inside the if statement and the lines print('bacon') and print('ham').  
  
print('eggs')  
if spam > 5:  
print('bacon')  
else:  
print('ham')  
print('spam')

1. Create a programme that prints. If 1 is stored in spam, prints Hello; if 2 is stored in spam, prints Howdy; and if 3 is stored in spam, prints Salutations! if there is something else in spam.

**Ans:** The code is as below:  
  
if spam == 1:  
print('Hello')  
elif spam == 2:  
print('Howdy')  
else:  
print('Greetings!')

1. If your programme is stuck in an endless loop, what keys can you press?

**Ans:** Press CTRL-C to stop a program stuck in an infinite loop.

1. How can you tell the difference between break and continue?

**Ans:** The break statement will move the execution outside and just after a loop. The continue statement will move the execution to the start of the loop.

1. In a for loop, what is the difference between range (10), range (0, 10), and range (0, 10, 1)?

**Ans:** They all do the same thing. The range(10) call ranges from 0 up to (but not including) 10, range(0, 10) explicitly tells the loop to start at 0, and range(0, 10, 1) explicitly tells the loop to increase the variable by 1 on each iteration.

1. Using a for loop, write a short programme that prints the numbers 1 to 10 Then, using a while loop, create an identical programme that prints the numbers 1 to 10.

**Ans:** The code is as below:  
  
a) for i in range(1, 11):  
 print(i)  
  
b) i = 0  
 while i <= 9:  
 i = i + 1

print(i)

1. If you had a bacon () function within a spam module, what would you call it after importing spam?

**Ans:** This function can be called with spam.bacon().