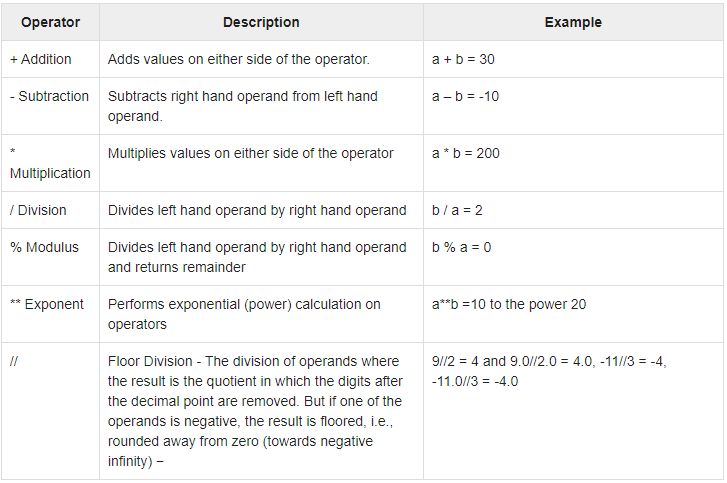
**Python Basic Operators.**

An operator, in computer programming, is a symbol that usually represents an action or process.

Python supports the following types of operators: arithmetic, comparison, assignment, logical, bitwise, membership and identity. Let's explore some of the operators below.

**Arithmetic Operators**

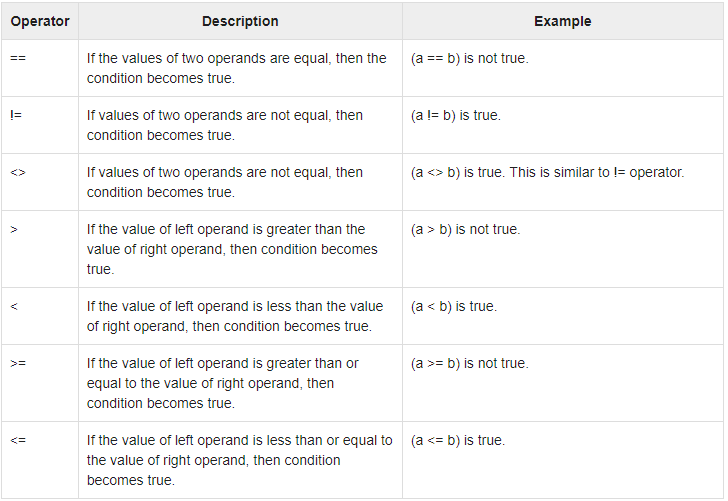
**Arithmetic operators** take numerical values (either literals or variables) as their operands and return a single numerical value.

Assume a variable **a** holds ***10*** and variable **b** holds ***20,*** *see the example below*:

**Comparison operators**

Comparison operators are used to compare values. They either return **True**  or **False** according to the condition. They are also called Relational operators.

Assume variable **a** holds **10** and variable **b** holds 20, then see the diagram below.



**Logical operators.**

Logical operators are the ***and***, ***or***, ***not*** operators.

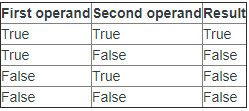
***and*** - returns **True** only if both operands are **true**. In any other case, **False** will be returned.

For example, the following expression will evaluate to True: **5 < 7 and 5 > 3**, because

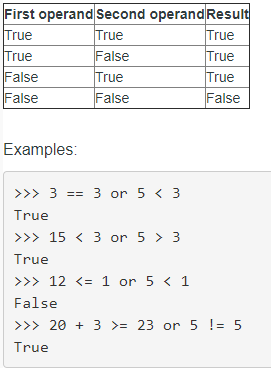
5 is indeed less than 7 and greater than 3.

Below is the ***and*** operator truth table(a table that lists all the possible inputs and

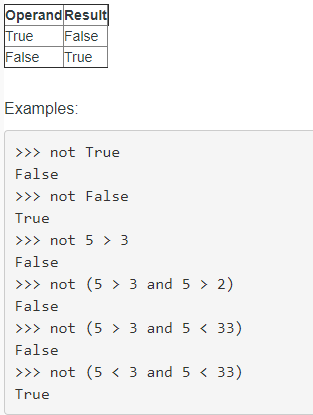
the results for the logical operators).



***or*** - returns True when one or both of the operants 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:



**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:



Let’s explain the last example – **not (5 < 3 and 5 < 33)** – and why it returns **True**. Simply evaluate the expression in the parentheses first: 5 < 3, 5 isn’t smaller than 3, so this expression is false. **5 < 33**, is smaller than 33, so this expression is **true**.

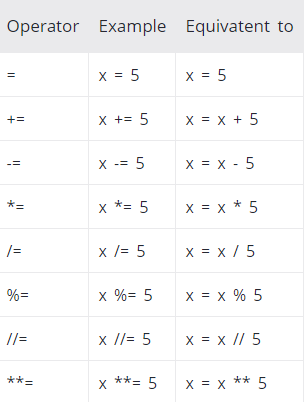
From the truth table, for the **and** operator above, we know that **False** and **True** return **False**. So the expression **5 < 3 and 5 < 33** will return **False**. Now we need to apply the **not** operator, so **False** becomes **True**.

**Assignment operators**

**Assignment operators** are used in Python to assign values to variables.

***a*** = 5 is a simple assignment operator that assigns the value of 5 on the right to the variable ***a*** on the left. There are various compound operators in Python like **a +=** 5 that adds to the variable and later assigns the same to it. It is equivalent to ***a = a + 5***.

See below, some of compound statements:



**Bitwise operators** -Bitwise operators acts on bits and performs bit by bit operation.

You're short of space and need to use as little memory as possible and you squeeze multiple bit values and groups of bits into bytes, words, double words and quad words. You choose to use bitwise operations to save space. This is the real world example of bitwise operators usage.

**Task**

**Exercise 1**

We have the variable a = 0b10111011. Use a bitmask and the value a in order to achieve a result where the third bit from the right of a is turned on. Be sure to print your answer as a bin() string.

**Exercise 2**

We have the variable a = 0b10111011..Use a bitmask and the value a in order to achieve a result where all of the bits in a are flipped. Be sure to print your answer as a bin() string.