

# Python Comparison Operators

==	Equal	x == y
----	-------	--------

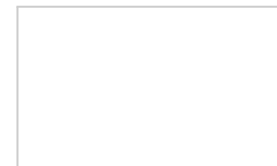
!=	Not equal	x != y
----	-----------	--------

>	Greater than	x > y
---	--------------	-------

<	Less than	x < y
---	-----------	-------

>=	Greater than or equal to	x >= y
----	--------------------------	--------

<=	Less than or equal to	x <= y
----	-----------------------	--------



# Python Logical Operators

Operator	Description	Example
and	Returns True if both	x < 5 and x < 10

statements are true

or	Returns True if one of the statements is true	$x < 5$ or $x < 4$
----	---	--------------------

not	Reverse the result, returns False if the result is true	$\text{not}(x < 5 \text{ and } x < 10)$
-----	---	---



Example :

$x = 5$

$\text{print}(x > 3 \text{ and } x < 10)$

# returns True because 5 is greater than 3 AND 5 is less than 10

## Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	$x \text{ is } y$
is not	Returns True if both	$x \text{ is not } y$

variables are not the same  
object

Example:

```
y = ["apple", "banana"]
```

```
z = x
```

```
print(x is z)      # returns True because z is the same object as x
```

```
print(x is y)      # returns False because x is not the same object as y,  
even if they have the same content
```

```
print(x == y)      # to demonstrate the difference between "is" and  
"==": this comparison returns True because x is equal to y
```

Example

```
x = ["apple", "banana"]
```

```
y = ["apple", "banana"]
```

```
z = x
```

```
print(x is not z)   # returns False because z is the same object  
as x
```

```
print(x is not y)   # returns True because x is not the same  
object as y, even if they have the same content
```

```
print(x != y)
```

## Python Bitwise Operators

perato r	Name	Description	Example
-------------	------	-------------	---------

&	AND	Sets each bit to 1 if both bits are 1	$x \& y$
	OR	Sets each bit to 1 if one of two bits is 1	$x   y$
^	XOR	Sets each bit to 1 if only one of two bits is 1	$x \wedge y$
~	NOT	Inverts all the bits	$\sim x$
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	$x \ll 2$
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	$x \gg 2$

Example :  
`print(6 & 3)`

"""

The & operator compares each bit and set it to 1 if both are 1, otherwise it is set to 0:

6 = 00000110

3 = 00000011

-----

2 = 00000010

=====

Example

print(6 | 3)

"""

The | operator compares each bit and set it to 1 if one or both is 1, otherwise it is set to 0:

6 = 00000110

3 = 00000011

-----

7 = 00000111

=====

print(6 ^ 3)

"""

The ^ operator compares each bit and set it to 1 if only one is 1, otherwise (if both are 1 or both are 0) it is set to 0:

6 = 00000110

3 = 00000011

-----

5 = 00000101