

PYTHON - BASIC OPERATORS

https://www.tutorialspoint.com/python/python_basic_operators.htm

Copyright © tutorialspoint.com

Operators are the constructs which can manipulate the value of operands.

Consider the expression $4 + 5 = 9$. Here, 4 and 5 are called operands and + is called operator.

Types of Operator

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison *Relational* Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Let us have a look on all operators one by one.

Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then –

[[Show Example](#)]

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a + b = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator	$a * b = 200$
/ Division	Divides left hand operand by right hand operand	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 0$
** Exponent	Performs exponential <i>power</i> calculation on	$a ** b = 10 \text{ to the power } 20$

	operators	
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero <i>towards negative infinity</i> –	$9//2 = 4$ and $9.0//2.0 = 4.0$, $-11//3 = -4$, $-11.0//3 = -4.0$

Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds 10 and variable b holds 20, then –

[[Show Example](#)]

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	$a == b$ is not true.
!=	If values of two operands are not equal, then condition becomes true.	$a != b$ is true.
<>	If values of two operands are not equal, then condition becomes true.	$a <> b$ is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	$a > b$ is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	$a < b$ is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	$a >= b$ is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	$a <= b$ is true.

Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then –

[[Show Example](#)]

Operator	Description	Example
=	Assigns values from right side operands to left side operand	$c = a + b$ assigns value of $a + b$ into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	$c += a$ is equivalent to $c = c + a$
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	$c -= a$ is equivalent to $c = c - a$
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	$c *= a$ is equivalent to $c = c * a$
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	$c /= a$ is equivalent to $c = c / a$ $c /= a$ is equivalent to $c = c / a$
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	$c \% = a$ is equivalent to $c = c \% a$
**= Exponent AND	Performs exponential <i>power</i> calculation on operators and assign value to the left operand	$c ** = a$ is equivalent to $c = c ** a$
//= Floor Division	It performs floor division on operators and assign value to the left operand	$c //= a$ is equivalent to $c = c // a$

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if $a = 60$; and $b = 13$; Now in binary format they will be as follows –

$a = 0011\ 1100$

$b = 0000\ 1101$

$a \& b = 0000\ 1100$

$a | b = 0011\ 1101$

$a \wedge b = 0011\ 0001$

$\sim a = 1100\ 0011$

There are following Bitwise operators supported by Python language

[[Show Example](#)]

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	$a \& b$ means 00001100
Binary OR	It copies a bit if it exists in either operand.	$a b = 61$ means 00111101
^ Binary XOR	It copies the bit if it is set in one operand but not both.	$a \wedge b = 49$ means 00110001
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	$a = -61$ (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	$a \ll 2 = 240$ means 11110000
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	$a \gg 2 = 15$ means 00001111

Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

[[Show Example](#)]

Used to reverse the logical state of its operand.

Python Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below –

[[Show Example](#)]

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below –

[[Show Example](#)]

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id_x equals id_y .
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id_x is not equal to id_y .

Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

[[Show Example](#)]

Sr.No.	Operator & Description
1	<p>**</p> <p>Exponentiation <i>raisetothepower</i></p>
2	<p>~ + -</p> <p>Complement, unary plus and minus <i>methodnamesforthelasttwoare + @and - @</i></p>

3	<code>* / % //</code> Multiply, divide, modulo and floor division
4	<code>+ -</code> Addition and subtraction
5	<code>>> <<</code> Right and left bitwise shift
6	<code>&</code> Bitwise 'AND'
7	<code>^ </code> Bitwise exclusive 'OR' and regular 'OR'
8	<code><= < > >=</code> Comparison operators
9	<code><> == !=</code> Equality operators
10	<code>= %= /= //= -= += *= **=</code> Assignment operators
11	<code>is is not</code> Identity operators

12	in not in Membership operators
13	not or and Logical operators