PYTHON BASIC OPERATORS

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 $power$ calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 = 4 and 9.0//2.0 = 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.	
<>	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 / ac /= 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 *power* 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^b = 0011 0001

~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^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 << = 240 $means$11110000
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> = 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

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 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]

Operator	Description
**	Exponentiation $raisetothepower$
~ + -	Ccomplement, unary plus and minus $methodnamesforthelasttwoare + @and - @$
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift

&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators