Operators

An operator is a symbol which operates on a value or a variable. For example: + is an operator to perform addition. Python has wide range of operators to perform various. Python language is rich in built-in operators and provides the following types of operators.

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

Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction and multiplication on numerical values Assume variable **A** holds 10 and variable **B** holds 5 then.

Operator	Description	Example
+	Adds two operands.	A + B = 15
−	Subtracts second operand from the first.	A − B = 5
*	Multiplies both operands.	A * B = 50
/	Divides numerator by de-numerator.	A / B =2
%	Modulus Operator and remainder of after an integer division.	A % B = 0

** (Exponent)	It is an exponent operator represented as it calculates the first operand power to second operand.	A**B=10**5 =100000
// (Floor division)	It gives the floor value of the quotient produced by dividing the two operands.	A//B=10//5=2

Example	**Output**
a=10	Addition= 15
b=5	Substraction= 5
print"Addition=",(a+b)	Multiplication= 50
print"Substraction=",(a-b)	Division= 2
print"Multiplication=",(a*b)	Exponent= 100000
print"Division=",(a/b)	Floor division= 2
print"Exponent=",(a**b)	
print"Floor division=",(a//b)	

Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1 or true; if the relation is false, it returns value 0 or false Assume variable **A** holds 10 and variable **B** holds 5 then .

Operator	**Description**	**Example**
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are	(A != B) is

	equal or not. If the values are not equal, then the condition becomes true.	true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is not true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is not true.
<>	If the value of two operands is not equal, then the condition becomes true.	(A<>B) Is true

Example	**Output**

a=10
b=5
print(a==b)
print(a!=b)
print(a>b)
print(a<b)
print(a>=b)
print(a<=b)
print(a<>b)
	False
	True
	True
	False
	True
	False
	True

Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =.

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and	C %= A is equivalent

	assigns the result to the left operand.	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

Example	**Output**

a = 10
b = 5
c = 0
 	
c = a + b
print "c=a+b=", c
 	
c += a
print "c=c+a=", c
 	
c *= a
print "c=c*a=", c
 	
c /= a
print "c=c/a=", c
 	
c %= a
print "c=c%a=", c
	c=a+b= 15
	c=c+a= 25
	c=c*a= 250
	c=c/a= 25
	c=c%a= 5
	c=c**a= 1024
	c=c//a= 102

c=2 c **= a print "c=c**a=", c c //= a print "c=c//a=", c c	

Logical Operators

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following logical operators.

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Bitwise Operators

The bitwise operators perform bit by bit operation on the values of the two operands.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B)

|	Binary OR Operator copies a bit if it exists in either operand.	(A | B)
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B)
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A)
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2

Membership Operators

Python membership operators are used to check the membership of value inside a data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

Operator	Description
in	The result of this operation becomes True if it finds a value in a specified sequence & False otherwise.
not in	result of this operation becomes True if it doesn't find

	a value in a specified sequence & False otherwise.

Example	Output

a = 10
b = 5
l = [1, 2, 3, 4, 5];
 	
if (a in l):
  print "true"
else:
  print "false"
 	
if (a not in l):
  print "true"
else:
  print "false"
 	
if (b in l):
  print "true"
else:
  print "false"
	false
	true
	true

Identity Operators

Identity operators compare the memory locations of two objects.

Operator	Description
is	It is evaluated to be true if the reference present at

	both sides point to the same object.
is not	It is evaluated to be true if the reference present at both side do not point to the same object.

Example	**Output**

a = 10
b = 5
 	
if (a is b):
   print "true"
else:
   print "false"
 	
if (a is not b):
   print "true"
else:
   print "false"
	false
	true