Logical Operators

Logical operators are used to combine conditional statements:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>Returns True if both statements are true</td>
<td>x &lt; 5 and x &lt; 10</td>
</tr>
<tr>
<td>or</td>
<td>Returns True if one of the statements is true</td>
<td>x &lt; 5 or x &lt; 4</td>
</tr>
<tr>
<td>not</td>
<td>Reverse the result, returns False if the result is true</td>
<td>not(x &lt; 5 and x &lt; 10)</td>
</tr>
</tbody>
</table>

Example #1:

```python
# Python program to demonstrate
# logical and operator
a = 10
b = 10
c = -10

if a > 0 and b > 0:
    print("The numbers are greater than 0")

if a > 0 and b > 0 and c > 0:
    print("The numbers are greater than 0")
else:
    print("Atleast one number is not greater than 0")
```

Output:

```
The numbers are greater than 0
Atleast one number is not greater than 0
```
Example #2:

# Python program to demonstrate
# logical and operator

a = 10
b = 12
c = 0

if a and b and c:
    print("All the numbers have boolean value as True")
else:
    print("Atleast one number has boolean value as False")

Output:

Atleast one number has boolean value as False

Example #3:

# Python program to demonstrate
# logical not operator

a = 10

if not a:
    print("Boolean value of a is True")

if not (a%3 == 0 or a%5 == 0):
    print("10 is not divisible by either 3 or 5")
else:
    print("10 is divisible by either 3 or 5")

Output:

10 is divisible by either 3 or 5
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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>Returns True if both variables are the same object</td>
<td>x is y</td>
</tr>
<tr>
<td>is not</td>
<td>Returns True if both variables are not the same object</td>
<td>x is not y</td>
</tr>
</tbody>
</table>

Example

```python
a = 20
b = 20

if ( a is b):
    print("Line 1 - a and b have same identity")
else:
    print("Line 1 - a and b do not have same identity")

if (id(a) == id(b)):
    print("Line 2 - a and b have same identity")
else:
    print("Line 2 - a and b do not have same identity")

b = 30
if ( a is b):
    print("Line 3 - a and b have same identity")
else:
    print("Line 3 - a and b do not have same identity")

if ( a is not b):
    print("Line 4 - a and b do not have same identity")
else:
    print("Line 4 - a and b have same identity")
```

When you execute the above program it produces the following result —

Line 1 - a and b have same identity
Line 2 - a and b have same identity
Line 3 - a and b do not have same identity
Line 4 - a and b do not have same identity
Membership Operators

Membership operators are used to test if a sequence is presented in an object:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>Returns True if a sequence with the specified value is present in the object</td>
<td>x in y</td>
</tr>
<tr>
<td>not in</td>
<td>Returns True if a sequence with the specified value is not present in the object</td>
<td>x not in y</td>
</tr>
</tbody>
</table>

Example

```python
a = 10
b = 20
list = [1, 2, 3, 4, 5];

if (a in list):
    print("Line 1 - a is available in the given list")
else:
    print("Line 1 - a is not available in the given list")

if (b not in list):
    print("Line 2 - b is not available in the given list")
else:
    print("Line 2 - b is available in the given list")

a = 2
if (a in list):
    print("Line 3 - a is available in the given list")
else:
    print("Line 3 - a is not available in the given list")
```

When you execute the above program it produces the following result –

Line 1 - a is not available in the given list

Line 2 - b is not available in the given list

Line 3 - a is available in the given list
Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>AND</td>
<td>Sets each bit to 1 if both bits are 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
</tr>
<tr>
<td>^</td>
<td>XOR</td>
<td>Sets each bit to 1 if only one of two bits is 1</td>
</tr>
<tr>
<td>~</td>
<td>NOT</td>
<td>Inverts all the bits</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Zero fill left shift</td>
<td>Shift left by pushing zeros in from the right and let the leftmost bits fall off</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Signed right shift</td>
<td>Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off</td>
</tr>
</tbody>
</table>

Example

```plaintext
a = 60  # 60 = 0011 1100
b = 13  # 13 = 0000 1101
c = 0

c = a & b;  # 12 = 0000 1100
print("Line 1 - Value of c is ", c)

c = a | b;  # 61 = 0011 1101
print("Line 2 - Value of c is ", c)

c = a ^ b;  # 49 = 0011 0001
print("Line 3 - Value of c is ", c)

c = ~a;  # -61 = 1100 0011
print("Line 4 - Value of c is ", c)

c = a << 2;  # 240 = 1111 0000
print("Line 5 - Value of c is ", c)

c = a >> 2;  # 15 = 0000 1111
print("Line 6 - Value of c is ", c)
```

When you execute the above program it produces the following result−

Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15