04-Python Basic

Operator in python

Posted on September 8, 2021
Last updated on February 1, 2023

4 Operators

Operator is a symbol that performs certain operations. Python provides the following set of operators

1. Arithmetic Operators
2. Relational Operators or Comparison Operators
3. Logical operators
4. Bitwise oeprators
5. Assignment operators
6. Special operators

4.1 Arithmetic Operators

Operator	Meaning	Syntax
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y

Operator	Meaning	Syntax
**	Exponentiation	x ** y
//	Floor division	x // y

4.2 Comparison Operators

Operator	Meaning	Syntax
==	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

4.3 Logical Operators:

```
# Returns True if both statements are true
and

# Returns True if one of the statements is true
or

# Reverse the result, returns False if the result is true
not
```

```
True and True  # True

True or False  # True

not True  # False
not 0     # True
not 1000  # False
```

[4.4 Bitwise Operators]

The bitwise operators are used to perform bitwise calculations on integers. The integers are first `converted into binary` and `then operations are performed on each bit` or corresponding pair of bits, hence the name bitwise operators. The result is then returned in decimal format.

These operators are applicable `only for int and boolean` types. By mistake if we are trying to apply for any other type then we will get Error.

Operator	Meaning	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
»	Bitwise right shift	x»
«	Bitwise left shift	x«

```
# Example
a   = 10 = 1010 (Binary)
b   = 4  = 0100 (Binary)
10&4= 0  = 0000 (Binary)
```

```
# Bitwise right shift

Example 1:
a = 10 = 0000 1010 (Binary)
a >> 1 = 0000 0101 = 5

Example 2:
a = -10 = 1111 0110 (Binary)
a >> 1  = 1111 1011 = -5
```

```
# Bitwise left shift
Example 1:
a = 5  = 0000 0101 (Binary)
a << 1 = 0000 1010 = 10
a << 2 = 0001 0100 = 20

Example 2:
b = -10 = 1111 0110 (Binary)
b << 1  = 1110 1100 = -20
b << 2  = 1101 1000 = -40
```

```
print(True & False) # False
print(True | False) # True
print(True ^ False) # True
print(~True) # -2
print(True<<2) # 4
print(True>>2) # 0
```

4.5 Assignment Operators:

We can use assignment operator to assign value to the variable.

Operator	Meaning	Syntax
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3

Operator	Meaning	Syntax
^=	x ^= 3	x = x ^ 3
»=	x »= 3	x = x » 3
«=	x «= 3	x = x « 3

4.6 Identity Operators

```
# True if both var are the same object
is -->  x is y

# True if both var are not the same object
is not  x is not y
```

```
p = [1,2,3]
q = [1,2,3]
p == q  # True coz val is same
p is q  # False coz not same obj

a = "first"
b = "first"
a is b # --> True
```

- NOTE:
  - Immutable object (str, int, tuple) `refers to same obj if values are same`
  - Mutable obj always create new object

4.7 Membership Operators

We can use Membership operators to check whether the given object present in the given collection. It may be String, List, Set, Tuple or Dict

```
# True if the sequence x present in y
in      # x in y

# True if the sequence x not present in y
not in # x not in y
```

```
x="hello learning Python is very easy!!!"
'h' in x      # True
'Python' in x  # True

list1=["sunny","bunny","chinny","pinny"]
"sunny" in list1    # True
"tunny" not in list1 # True
```

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.