

Lesson: Python Development

DS260222A

Operators

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operator

Operator	Description
+ (Addition)	It is used to add two operands. For example, if $a = 20$, $b = 10 \Rightarrow a + b = 30$
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if $a = 20$, $b = 10 \Rightarrow a - b = 10$
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if $a = 20$, $b = 10 \Rightarrow a / b = 2.0$
* (Multiplication)	It is used to multiply one operand with the other. For example, if $a = 20$, $b = 10 \Rightarrow a * b = 200$
% (reminder)	It returns the reminder after dividing the first operand by the second operand. For example, if $a = 20$, $b = 10 \Rightarrow a \% b = 0$
** (Exponent)	It is an exponent operator represented as it calculates the first operand power to the second operand.
// (Floor division)	It gives the floor value of the quotient produced by dividing the two operands.

Example for Arithmetic Operator

Operators	Meaning	Example	Result
+	Addition	$4 + 2$	6
-	Subtraction	$4 - 2$	2
*	Multiplication	$4 * 2$	8
/	Division	$4 / 2$	2
%	Modulus operator to get remainder in integer division	$5 \% 2$	1
**	Exponent	$5 ** 2 = 5^2$	25
//	Integer Division/ Floor Division	$5 // 2$ $-5 // 2$	2 -3

Comparison Operator

Operator	Description
==	If the value of two operands is equal, then the condition becomes true.
!=	If the value of two operands is not equal, then the condition becomes true.
<=	If the first operand is less than or equal to the second operand, then the condition becomes true.
>=	If the first operand is greater than or equal to the second operand, then the condition becomes true.
>	If the first operand is greater than the second operand, then the condition becomes true.
<	If the first operand is less than the second operand, then the condition becomes true.

Example for Comparison Operator

Operators	Meaning	Example	Result
<	Less than	$5 < 2$	False
>	Greater than	$5 > 2$	True
<=	Less than or equal to	$5 \leq 2$	False
>=	Greater than or equal to	$5 \geq 2$	True
==	Equal to	$5 == 2$	False
!=	Not equal to	$5 != 2$	True

Logical Operator

Operator	Description
and	If both the expression are true, then the condition will be true. If a and b are the two expressions, $a \rightarrow \text{true}$, $b \rightarrow \text{true} \Rightarrow a \text{ and } b \rightarrow \text{true}$.
or	If one of the expressions is true, then the condition will be true. If a and b are the two expressions, $a \rightarrow \text{true}$, $b \rightarrow \text{false} \Rightarrow a \text{ or } b \rightarrow \text{true}$.
not	If an expression a is true, then not (a) will be false and vice versa.

Python Logical Operators

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	Not A
True	False
False	True

Example for Logical Operator

Operator	Meaning	Example	Result
and	Logical and	$(5 < 2)$ and $(5 > 3)$	False
or	Logical or	$(5 < 2)$ or $(5 > 3)$	True
not	Logical not	not $(5 < 2)$	True

Bitwise

Operator	Description
& (binary and)	If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied.
(binary or)	The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1.
^ (binary xor)	The resulting bit will be 1 if both the bits are different; otherwise, the resulting bit will be 0.

Operator	Meaning
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR / Bitwise XOR

BitWise – > XOR , AND, OR

Bitwise Xor Operations

Operation	Result
$0 \wedge 0$	0
$1 \wedge 0$	1
$0 \wedge 1$	1
$1 \wedge 1$	0

Implementation of And Operation on Binary Digits

Operation	Result
$0 \& 0$	0
$1 \& 0$	0
$0 \& 1$	0
$1 \& 1$	1

Bitwise Or Operations

Operation	Result
$0 \mid 0$	0
$1 \mid 0$	1
$0 \mid 1$	1
$1 \mid 1$	1

325 × 238

Example BitWise

Example :

$X = 5$ # binary $\rightarrow 0101$

$Y = 1$ # binary $\rightarrow 0001$

Output = $X \& Y \rightarrow 0001$

Python Membership Operators

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

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Example

```
1 #Membership Operators
2 a = 5
3 b = 10
4 c = [1,2,3,4,5]
5
6 print(a in c)
7 print(a not in c)
8 print(b in c)
9 print(b not in c)
```

Python 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:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

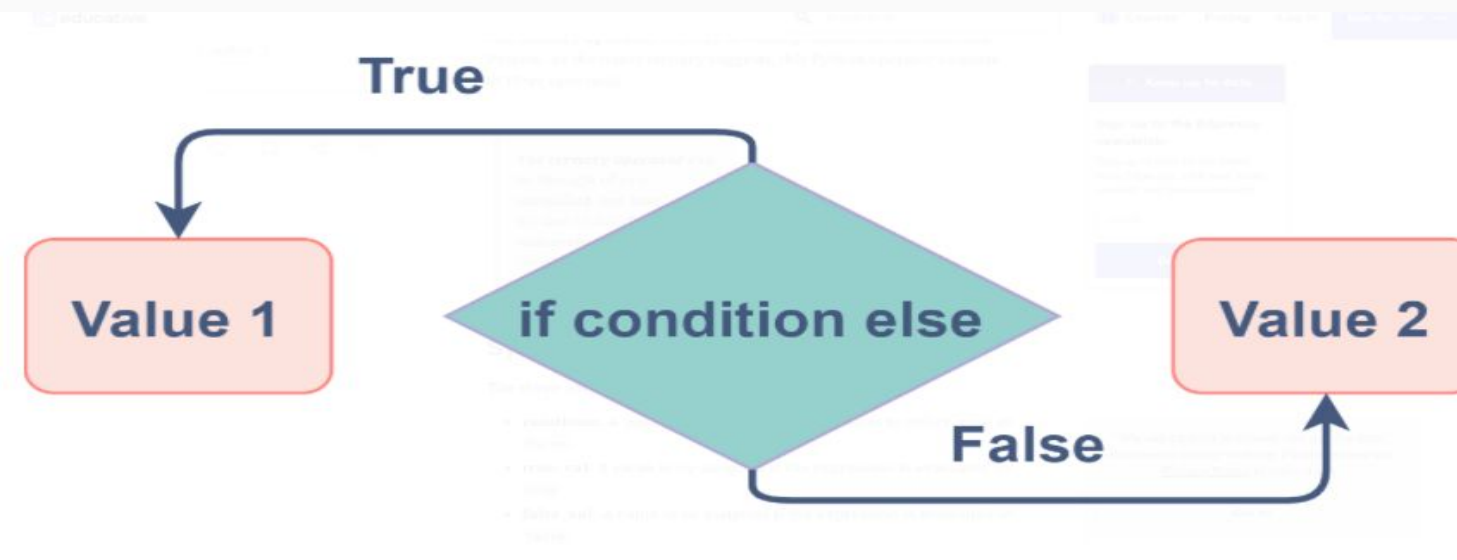
Example

```
1 #Identity Operators
2 a = [1,2,3,4,5]
3 b = [1,2,3,4,5]
4 c = a
5
6 print(a is c)
7 print(a is b)
8 print(a is not c)
9 print(a is not b)
```


Ternary Operator in Python

Syntax:

```
[on_true] if [expression] else [on_false]
```



Ternary Operator or Conditional Operator

```
# Program to demonstrate conditional operator
a, b = 10, 20

# Copy value of a in min if a < b else copy b
min = a if a < b else b

print(min)
```

OUTPUT:

10

Assignment Operator Example

```
# Examples of Assignment Operators
```

```
a = 10
```

```
# Assign value
```

```
b = a
```

```
print(b)
```

```
# Add and assign value
```

```
b += a
```

```
print(b)
```

```
# Subtract and assign value
```

```
b -= a
```

```
print(b)
```

```
# multiply and assign
```

```
b *= a
```

```
print(b)
```

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`