Python Operators

In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for the purpose of logical and arithmetic operations. In this article, we will look into different types of **Python operators.**

- OPERATORS: These are the special symbols. Eg- + , * , /, etc.
- OPERAND: It is the value on which the operator is applied.

Types of Operators in Python

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Identity Operators and Membership Operators

Arithmetic Operators in Python

Python Arithmetic operators are used to perform basic mathematical operations like **addition, subtraction, multiplication**, and **division**.

In Python 3.x the result of division is a floating-point while in Python 2.x division of 2 integers was an integer. To obtain an integer result in Python 3.x floored (// integer) is used.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Example of Arithmetic Operators in Python

Division Operators

Division Operators allow you to divide two numbers and return a quotient, i.e., the first number or number at the left is divided by the second number or number at the right and returns the quotient.
There are two types of division operators:

1. Float division
2. Floor division

Float division

The quotient returned by this operator is always a float number, no matter if two numbers are integers. For example:

- Python3

```
# python program to demonstrate the use of "/"

print(5/5)

print(10/2)

print(-10/2)

print(20.0/2)
```

Output:

```
1.0

5.0

-5.0

10.0
```

Integer division(Floor division)

The quotient returned by this operator is dependent on the argument being passed. If any of the numbers is float, it returns output in float. It is also known as Floor division because, if any number is negative, then the output will be floored. For example:

- Python3

```
# python program to demonstrate the use of "//"

print(10//3)

print (-5//2)

print (5.0//2)

print (-5.0//2)
```

Output:

```
3

-3
```

```
2.0
-3.0
```

Precedence of Arithmetic Operators in Python

The precedence of Arithmetic Operators in python is as follows:

1. P – Parentheses
2. E – Exponentiation
3. M – Multiplication (Multiplication and division have the same precedence)
4. D – Division
5. A – Addition (Addition and subtraction have the same precedence)
6. S – Subtraction

The modulus operator helps us extract the last digit/s of a number. For example:

- x % 10 -> yields the last digit
- x % 100 -> yield last two digits

Arithmetic Operators With Addition, Subtraction, Multiplication, Modulo and Power

Here is an example showing how different Arithmetic Operators in Python work:

- Python3

```python
# Examples of Arithmetic Operator
a = 9
b = 4

# Addition of numbers
add = a + b

# Subtraction of numbers
sub = a - b

# Multiplication of number
mul = a * b

# Modulo of both number
mod = a % b

# Power
```

```
p = a ** b


# print results
print(add)
print(sub)
print(mul)
print(mod)
print(p)
```

Output:

```
13
5
36
1
6561
```

Note: *Refer to Differences between / and // for some interesting facts about these two operators.*

Comparison Operators in Python

In Python Comparison of Relational operators compares the values. It either returns **True** or **False** according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to True if the left operand is greater than or equal to the right	x >= y
<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y

= is an assignment operator and == comparison operator.

Precedence of Comparison Operators in Python

In python, the comparison operators have lower precedence than the arithmetic operators. All the operators within comparison operators have same precedence order.

Example of Comparison Operators in Python

Let’s see an example of Comparison Operators in Python.

- Python3

```
# Examples of Relational Operators

a = 13

b = 33


# a > b is False

print(a > b)


# a < b is True

print(a < b)


# a == b is False

print(a == b)


# a != b is True

print(a != b)


# a >= b is False

print(a >= b)


# a <= b is True

print(a <= b)
```

Output

```
False

True

False
```

```
True
False
True
```

Logical Operators in Python

Python Logical operators perform **Logical AND**, **Logical OR**, and **Logical NOT** operations. It is used to combine conditional statements.

Operator	**Description**	**Syntax**
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

Precedence of Logical Operators in Python

The precedence of Logical Operators in python is as follows:

1. Logical not
2. logical and
3. logical or

Example of Logical Operators in Python

The following code shows how to implement Logical Operators in Python:

- Python3

```python
# Examples of Logical Operator
a = True
b = False

# Print a and b is False
print(a and b)

# Print a or b is True
print(a or b)
```

```python
# Print not a is False
print(not a)
```

Output

```
False
True
False
```

Bitwise Operators in Python

Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Precedence of Bitwise Operators in Python

The precedence of Bitwise Operators in python is as follows:

1. Bitwise NOT
2. Bitwise Shift
3. Bitwise AND
4. Bitwise XOR
5. Bitwise OR

Bitwise Operators in Python

Here is an example showing how Bitwise Operators in Python work:

- Python3

```python
# Examples of Bitwise operators

a = 10

b = 4



# Print bitwise AND operation

print(a & b)



# Print bitwise OR operation

print(a | b)



# Print bitwise NOT operation

print(~a)



# print bitwise XOR operation

print(a ^ b)



# print bitwise right shift operation

print(a >> 2)



# print bitwise left shift operation

print(a << 2)
```

Output

```
0

14

-11

14

2

40
```

Assignment Operators in Python

Python Assignment operators are used to assign values to the variables.

Operator	Description	Syntax
=	Assign the value of the right side of the expression to the left side operand	x = y + z
+=	Add AND: Add right-side operand with left-side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b a=a&b
|=	Performs Bitwise OR on operands and assign value to left operand	a|=b a=a|b
^=	Performs Bitwise xOR on operands and assign value to left operand	a^=b a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b a=a>>b

Operator	Description	Syntax
<<=	Performs Bitwise left shift on operands and assign value to left operand	a <<= b a= a << b

Assignment Operators in Python

Let's see an example of Assignment Operators in Python.

- Python3

```
# 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)


# bitwise lishift operator
b <<= a
print(b)
```

Output

```
10
20
```

```
10

100

102400
```

Identity Operators in Python

In Python, **is** and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

is	True if the operands are identical
is not	True if the operands are not identical

Example Identity Operators in Python

Let’s see an example of Identity Operators in Python.

- Python3

```
a = 10

b = 20

c = a


print(a is not b)

print(a is c)
```

Output

```
True

True
```

Membership Operators in Python

In Python, **in** and **not in** are the membership operators that are used to test whether a value or variable is in a sequence.

in	True if value is found in the sequence
not in	True if value is not found in the sequence

Examples of Membership Operators in Python

The following code shows how to implement Membership Operators in Python:

- Python3

```
# Python program to illustrate

# not 'in' operator

x = 24

y = 20

list = [10, 20, 30, 40, 50]
```

```
if (x not in list):
    print("x is NOT present in given list")
else:
    print("x is present in given list")

if (y in list):
    print("y is present in given list")
else:
    print("y is NOT present in given list")
```

Output

```
x is NOT present in given list
y is present in given list
```

Ternary Operator in Python

in Python, Ternary operators also known as conditional expressions are operators that evaluate something based on a condition being true or false. It was added to Python in version 2.5.
It simply allows testing a condition in a **single line** replacing the multiline if-else making the code compact.
Syntax : *[on_true] if [expression] else [on_false]*

Examples of Ternary Operator in Python

Here is a simple example of Ternary Operator in Python.

- Python3

```
# 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
```

Precedence and Associativity of Operators in Python

In Python, Operator precedence and associativity determine the priorities of the operator.

Operator Precedence in Python

This is used in an expression with more than one operator with different precedence to determine which operation to perform first.

Let's see an example of how Operator Precedence in Python works:

- Python3

```python
# Examples of Operator Precedence


# Precedence of '+' & '*'

expr = 10 + 20 * 30

print(expr)


# Precedence of 'or' & 'and'

name = "Alex"

age = 0


if name == "Alex" or name == "John" and age >= 2:

    print("Hello! Welcome.")

else:

    print("Good Bye!!")
```

Output

```
610

Hello! Welcome.
```

Operator Associativity in Python

If an expression contains two or more operators with the same precedence then Operator Associativity is used to determine. It can either be Left to Right or from Right to Left.

The following code shows how Operator Associativity in Python works:

- Python3

```python
# Examples of Operator Associativity
# Left-right associativity
# 100 / 10 * 10 is calculated as
# (100 / 10) * 10 and not
# as 100 / (10 * 10)
print(100 / 10 * 10)

# Left-right associativity
# 5 - 2 + 3 is calculated as
# (5 - 2) + 3 and not
# as 5 - (2 + 3)
print(5 - 2 + 3)

# left-right associativity
print(5 - (2 + 3))

# right-left associativity
# 2 ** 3 ** 2 is calculated as
# 2 ** (3 ** 2) and not
# as (2 ** 3) ** 2
print(2 ** 3 ** 2)
```

Output

```
100.0

6

0

512
```