Types of Python Operators
Here's a list of different types of Python operators that we will learn in this tutorial.

Arithmetic operators
Assignment Operators
Comparison Operators
Logical Operators
Bitwise Operators
Special Operators
1. Python Arithmetic Operators
Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc. For example,

```
sub = 10 - 5 # 5
```

Here, - is an arithmetic operator that subtracts two values or variables.

Operator	Operation	Example
+	Addition	5 + 2 = 7
-	Subtraction	4 - 2 = 2
*	Multiplication	2 * 3 = 6
/	Division	4 / 2 = 2
//	Floor Division	10 // 3 = 3
%	Modulo	5 % 2 = 1
**	Power	4 ** 2 = 16

Example 1: Arithmetic Operators in Python

```
a = 7
b = 2

# addition
print ('Sum: ', a + b)

# subtraction
print ('Subtraction: ', a - b)

# multiplication
print ('Multiplication: ', a * b)

# division
print ('Division: ', a / b)

# floor division
print ('Floor Division: ', a // b)

# modulo
```

```
print ('Modulo: ', a % b)

# a to the power b
print ('Power: ', a ** b)
```

Run Code

Output

```
Sum: 9
Subtraction: 5
Multiplication: 14
Division: 3.5
Floor Division: 3
Modulo: 1
Power: 49
```

In the above example, we have used multiple arithmetic operators,

+ to add a and b
- to subtract b from a
* to multiply a and b
/ to divide a by b
// to floor divide a by b
% to get the remainder
** to get a to the power b

2. Python Assignment Operators

Assignment operators are used to assign values to variables. For example,

```
# assign 5 to x
var x = 5
```

Here, = is an assignment operator that assigns 5 to x.

Here's a list of different assignment operators available in Python.

Operator	Name	Example
=	Assignment Operator	a = 7
+=	Addition Assignment	a += 1 # a = a + 1
-=	Subtraction Assignment	a -= 3 # a = a - 3
*=	Multiplication Assignment	a *= 4 # a = a * 4
/=	Division Assignment	a /= 3 # a = a / 3
%=	Remainder Assignment	a %= 10 # a = a % 10
**=	Exponent Assignment	a **= 10 # a = a ** 10

Example 2: Assignment Operators

```
# assign 10 to a
a = 10
```

```
# assign 5 to b
b = 5

# assign the sum of a and b to a
a += b      # a = a + b

print(a)

# Output: 15
```

Run Code

Here, we have used the += operator to assign the sum of a and b to a.

Similarly, we can use any other assignment operators according to the need.

3. Python Comparison Operators

Comparison operators compare two values/variables and return a boolean result: True or False. For example,

```
a = 5
b =2

print (a > b)    # True
```

Run Code

Here, the > comparison operator is used to compare whether a is greater than b or not.

Operator	Meaning	Example
==	Is Equal To	3 == 5 gives us False
!=	Not Equal To	3 != 5 gives us True
>	Greater Than	3 > 5 gives us False
<	Less Than	3 < 5 gives us True
>=	Greater Than or Equal To	3 >= 5 give us False
<=	Less Than or Equal To	3 <= 5 gives us True

Example 3: Comparison Operators

```
a = 5

b = 2

# equal to operator
print('a == b =', a == b)

# not equal to operator
print('a != b =', a != b)
```

```
# greater than operator
print('a > b =', a > b)

# less than operator
print('a < b =', a < b)

# greater than or equal to operator
print('a >= b =', a >= b)

# less than or equal to operator
print('a <= b =', a <= b)
```
Run Code
Output

```
a == b = False
a != b = True
a > b = True
a < b = False
a >= b = True
a <= b = False
```
Note: Comparison operators are used in decision-making and loops. We'll discuss more of the comparison operator and decision-making in later tutorials.

4. Python Logical Operators
Logical operators are used to check whether an expression is True or False. They are used in decision-making. For example,

```
a = 5
b = 6

print((a > 2) and (b >= 6))    # True
```
Run Code
Here, and is the logical operator AND. Since both a > 2 and b >= 6 are True, the result is True.

Operator	Example	Meaning
and	a and b	Logical AND: True only if both the operands are True
or	a or b	Logical OR: True if at least one of the operands is True
not	not a	Logical NOT: True if the operand is False and vice-versa.

Example 4: Logical Operators

```
# logical AND
print(True and True)     # True
print(True and False)    # False

# logical OR
print(True or False)     # True

# logical NOT
print(not True)          # False
```

Run Code

Note: Here is the truth table for these logical operators.

5. Python Bitwise operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

For example, 2 is 10 in binary and 7 is 111.

In the table below: Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

6. Python Special operators

Python language offers some special types of operators like the identity operator and the membership operator. They are described below with examples.

Identity operators

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

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Example 4: Identity operators in Python

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
```

```
x3 = [1,2,3]
y3 = [1,2,3]

print(x1 is not y1)  # prints False

print(x2 is y2)  # prints True

print(x3 is y3)  # prints False
```
Run Code
Here, we see that x1 and y1 are integers of the same values, so they are equal as well as identical. Same is the case with x2 and y2 (strings).

But x3 and y3 are lists. They are equal but not identical. It is because the interpreter locates them separately in memory although they are equal.

Membership operators
In Python, in and not in are the membership operators. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Example 5: Membership operators in Python
```
x = 'Hello world'
y = {1:'a', 2:'b'}

# check if 'H' is present in x string
print('H' in x)  # prints True

# check if 'hello' is present in x string
print('hello' not in x)  # prints True

# check if '1' key is present in y
print(1 in y)  # prints True

# check if 'a' key is present in y
print('a' in y)  # prints False
```
Run Code
Output

```
True
True
```

True
False
Here, 'H' is in x but 'hello' is not present in x (remember, Python is case sensitive).

Similarly, 1 is key and 'a' is the value in dictionary y. Hence, 'a' in y returns False.