



SNS COLLEGE OF TECHNOLOGY



Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC –
UGC with 'A+' Grade Approved by AICTE, New Delhi &
Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19AMB302-FULL STACK AI

M.POORNIMA DEVI,AP/AIML

MATH OPERATORS AND EXPRESSIONS



- 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](#)



Operators in Python

Operators	Type
<code>+, -, *, /, %</code>	Arithmetic operator
<code><, <=, >, >=, ==, !=</code>	Relational operator
<code>&&, , !</code>	Logical operator
<code>&, , <<, >>, -, ^</code>	Bitwise operator
<code>=, +=, -=, *=, %=</code>	Assignment operator

1.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	<code>x + y</code>
-	Subtraction: subtracts two operands	<code>x - y</code>
*	Multiplication: multiplies two operands	<code>x * y</code>
/	Division (float): divides the first operand by the second	<code>x / y</code>
//	Division (floor): divides the first operand by the second	<code>x // y</code>
%	Modulus: returns the remainder when the first operand is divided by the second	<code>x % y</code>
**	Power: Returns first raised to power second	<code>x ** y</code>



Example of Arithmetic Operators in Python

2.Division Operators

In [Python programming](#) language **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.

- Example:** The code performs division operations and prints the results. It demonstrates that both integer and floating-point divisions return accurate results.

- For example, '10/2' results in '5.0', and '-10/2' results in '-5.0'.

EXAMPLE:Pythonprint(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:

Example: The code demonstrates integer (floor) division operations using the ‘//’ Python operators. It provides results as follows: ‘10//3’ equals ‘3’, ‘-5//2’ equals ‘-3’, ‘5.0//2’ equals ‘2.0’, and ‘-5.0//2’ equals ‘-3.0’.

- Integer division returns the largest integer less than or equal to the division result.

EXAMPLE:

```
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:

P – Parentheses

E – Exponentiation

M – Multiplication (Multiplication and division have the same precedence)

D – Division

A – Addition (Addition and subtraction have the same precedence)

S – Subtraction

•The modulus of Python operators 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:

Example: The code performs basic arithmetic operations with the values of 'a' and 'b'. It adds ('+'), subtracts ('-'), multiplies ('*'), computes the remainder ('%'), and raises a to the power of 'b (**)''. The results of these operations are printed.



1. Comparison of Python Operators

EXAMPLE:

```
a = 13 b = 33 print(a > b) print(a < b) print(a == b) print(a != b)
print(a >= b) print(a <= b)
```

OUTPUT:

False True False True False

2. Logical Operators in Python

EXAMPLE:

```
a = True b = False print(a and b) print(a or b) print(not a)
```

Output: False True False

3. Bitwise Operators in Python

EXAMPLE:

```
a = 10 b = 4 print(a & b) print(a | b) print(~a) print(a ^ b) print(a
>> 2) print(a << 2)
```

OUTPUT :0 14 -11 14 2 40



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

EXAMPLE:

```
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]*

EXAMPLE:

a, b = 10, 20 min = a **if** a < b **else** b print(min)

Output:

10

Q1. Code to implement basic arithmetic operations on integers

```
num1 = 5
```

```
num2 = 2
```

```
sum = num1 + num2
```

```
difference = num1 - num2
```

```
product = num1 * num2
```

```
quotient = num1 / num2
```

```
remainder = num1 % num2
```

```
print("Sum:", sum)
```

```
print("Difference:", difference)
```

```
print("Product:", product)
```

```
print("Quotient:", quotient)
```

```
print("Remainder:", remainder)
```

Output:

?





THANK YOU