



Python Bitwise Operators



nikhilaggarwal3



Read

Discuss

Courses

Practice

[Operators](#) are used to perform operations on values and variables. These are the special symbols that carry out arithmetic and logical computations. The value the operator operates on is known as Operand.

Table of Content:

- [Bitwise operators:](#)
 - [Bitwise AND operator](#)
 - [Bitwise OR operator](#)
 - [Bitwise not operator](#)
 - [Bitwise XOR operator](#)
- [Shift Operators:](#)
 - [Bitwise right shift](#)
 - [Bitwise left shift](#)
- [Bitwise Operator Overloading](#)

Bitwise operators

In Python, 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.

Note: Python bitwise operators work only on integers.

AD

e-GMAT

Strategies for 760 GMAT: balancing work, life, study.

OPEN >

OPERATOR	NAME	DESCRIPTION	SYNTAX
&	Bitwise AND	Result bit 1,if both operand bits are 1;otherwise results bit 0.	x & y
	Bitwise OR	Result bit 1,if any of the operand bit is 1; otherwise results bit 0.	x y
~	Bitwise NOT	inverts individual bits	~x
^	Bitwise XOR	Results bit 1,if any of the operand bit is 1 but not both, otherwise results bit 0.	x ^ y
>>	Bitwise right shift	The left operand's value is moved toward right by the number of bits specified by the right operand.	x>>
<<	Bitwise left shift	The left operand's value is moved toward left by the number of bits specified by the right operand.	x<<

Let's understand each operator one by one.

Bitwise AND operator Returns 1 if both the bits are 1 else 0.

Example:

```
a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)
```

```

a & b = 1010
      &
      0100
      = 0000
      = 0 (Decimal)

```

Bitwise or operator Returns 1 if either of the bit is 1 else 0.

Example:

```

a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)

a | b = 1010
      |
      0100
      = 1110
      = 14 (Decimal)

```

Bitwise not operator: Returns one's complement of the number.

Example:

```
a = 10 = 1010 (Binary)
```

In computers we usually represent numbers using 32 bits,
so binary representation of 10 is (...0000 1010)[32 bits]

~a is basically 1's complement of a

i.e ~a should be ~10 = ~(...0000 1010) = (...1111 0101) = intermediate-result

Since bitwise negation inverts the sign bit,
we now have a negative number. And we represent a negative number
using 2's complement.

2's complement of intermediate-result is:

```
intermediate-res = 0101      //...1111 0101
```

```

      1010      //...0000 1010 -(1's complement)
      +1
      -----
      = 1011      //...0000 1011
      -----
      = -11 (Decimal)

```

```
thus ~a = -11
```

Bitwise xor operator: Returns 1 if one of the bits is 1 and the other is 0 else returns false.

Example:

```
a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)

a ^ b = 1010
      ^
      0100
      = 1110
      = 14 (Decimal)
```

Python3

```
# Python program to show
# bitwise operators

a = 10
b = 4

# Print bitwise AND operation
print("a & b =", a & b)

# Print bitwise OR operation
print("a | b =", a | b)

# Print bitwise NOT operation
print("~a =", ~a)

# print bitwise XOR operation
print("a ^ b =", a ^ b)
```

Output:

```
a & b = 0
a | b = 14
~a = -11
a ^ b = 14
```

Shift Operators

These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively. They can be used when we have to multiply or divide a number by two.

Bitwise right shift: Shifts the bits of the number to the right and fills 0 on voids left(fills 1 in the case of a negative number) as a result. Similar effect as of dividing the number with some power of two.

Example:

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: Shifts the bits of the number to the left and fills 0 on voids right as a result. Similar effect as of multiplying the number with some power of two.

Example:

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`

Python3

```
# Python program to show
# shift operators

a = 10
b = -10

# print bitwise right shift operator
print("a >> 1 =", a >> 1)
print("b >> 1 =", b >> 1)

a = 5
b = -10

# print bitwise left shift operator
print("a << 1 =", a << 1)
print("b << 1 =", b << 1)
```

Output:

```
a >> 1 = 5
b >> 1 = -5
a << 1 = 10
b << 1 = -20
```

Bitwise Operator Overloading

[Operator Overloading](#) means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because the '+' operator is overloaded by int class and str class. You might have noticed that the same built-in operator or function shows different behavior for objects of different classes, this is called **Operator Overloading**.

Below is a simple example of Bitwise operator overloading.

Python3

```
# Python program to demonstrate
# operator overloading

class Geek():
    def __init__(self, value):
        self.value = value

    def __and__(self, obj):
        print("And operator overloaded")
        if isinstance(obj, Geek):
            return self.value & obj.value
        else:
            raise ValueError("Must be a object of class Geek")

    def __or__(self, obj):
        print("Or operator overloaded")
        if isinstance(obj, Geek):
            return self.value | obj.value
        else:
            raise ValueError("Must be a object of class Geek")

    def __xor__(self, obj):
        print("Xor operator overloaded")
        if isinstance(obj, Geek):
            return self.value ^ obj.value
        else:
            raise ValueError("Must be a object of class Geek")

    def __lshift__(self, obj):
        print("lshift operator overloaded")
        if isinstance(obj, Geek):
            return self.value << obj.value
        else:
            raise ValueError("Must be a object of class Geek")
```

```
def __rshift__(self, obj):
    print("rshift operator overloaded")
    if isinstance(obj, Geek):
        return self.value >> obj.value
    else:
        raise ValueError("Must be a object of class Geek")

def __invert__(self):
    print("Invert operator overloaded")
    return ~self.value

# Driver's code
if __name__ == "__main__":
    a = Geek(10)
    b = Geek(12)
    print(a & b)
    print(a | b)
    print(a ^ b)
    print(a << b)
    print(a >> b)
    print(~a)
```

Output:

```
And operator overloaded
8
Or operator overloaded
14
Xor operator overloaded
8
lshift operator overloaded
40960
rshift operator overloaded
8
Invert operator overloaded
-11
```

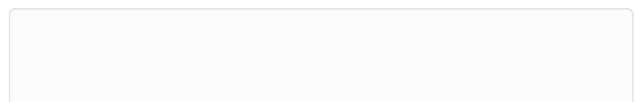
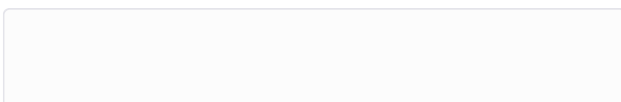
Note: To know more about operator overloading [click here](#).











Last Updated : 04 Apr, 2023

 78








Similar Reads




 G-Fact 19 (Logical and Bitwise Not	 Increment and Decrement
 Logical Operators on String in Python	 Inplace Operators in Python Set 1 (iadd(), isub(), iconcat(...))
 Inplace Operators in Python Set 2 (ixor(), iand(), ipow(),...)	 Inplace vs Standard Operators in Python
 Python Operators Question 1	 Python Operators Question 2
 Python Operators Question 3	 Python Operators Question 4

Related Tutorials

 Pandas AI: The Generative AI Python Library	 OpenAI Python API - Complete Guide
 Python for Kids - Fun Tutorial to Learn Python Programming	 Data Analysis Tutorial
 Flask Tutorial	

[< Previous](#)[Next >](#)

Article Contributed By :



nikhilagggarwal3
nikhilagggarwal3
[+ Follow](#)

Vote for difficulty

Current difficulty : [Medium](#)

Easy	Normal	Medium	Hard	Expert
------	--------	--------	------	--------

Improved By : [RajuKumar19](#), [andreasschlapbach](#), [raunakraj232](#), [sagartomar9927](#), [shaswatadas](#), [iammanish041](#), [joshuagornall](#), [sadiyakumar9211](#), [fahiar10](#), [gautamchaurasia501](#), [sijinsebastian](#), [alexandreo7ol](#), [nikita200ytqw](#)

Article Tags : [Python-Operators](#), [Python](#)

Practice Tags : [python](#), [python-operators](#)

[Improve Article](#)[Report Issue](#)**GeeksforGeeks**

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305



feedback@geeksforgeeks.org



Company

[About Us](#)
[Legal](#)
[Careers](#)
[In Media](#)
[Contact Us](#)
[Advertise with us](#)

Languages

[Python](#)
[Java](#)
[C++](#)
[PHP](#)
[GoLang](#)
[SQL](#)
[R Language](#)
[Android Tutorial](#)

DSA Roadmaps

[DSA for Beginners](#)

Explore

[Job-A-Thon For Freshers](#)
[Job-A-Thon For Experienced](#)
[GfG Weekly Contest](#)
[Offline Classes \(Delhi/NCR\)](#)
[DSA in JAVA/C++](#)
[Master System Design](#)
[Master CP](#)

DSA Concepts

[Data Structures](#)
[Arrays](#)
[Strings](#)
[Linked List](#)
[Algorithms](#)
[Searching](#)
[Sorting](#)
[Mathematical](#)
[Dynamic Programming](#)

Web Development

[HTML](#)

Basic DSA Coding Problems
Complete Roadmap To Learn DSA
DSA for FrontEnd Developers
DSA with JavaScript
Top 100 DSA Interview Problems

CSS
JavaScript
Bootstrap
ReactJS
AngularJS
NodeJS

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning Tutorial
Maths For Machine Learning
Pandas Tutorial
NumPy Tutorial
NLP Tutorial
Deep Learning Tutorial

Competitive Programming

Top DSA for CP
Top 50 Tree Problems
Top 50 Graph Problems
Top 50 Array Problems
Top 50 String Problems
Top 50 DP Problems
Top 15 Websites for CP

Interview Corner

Company Wise Preparation
Preparation for SDE
Experienced Interviews

Python

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
OpenCV Python Tutorial
Python Interview Question

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP

System Design

What is System Design
Monolithic and Distributed SD
Scalability in SD
Databases in SD
High Level Design or HLD
Low Level Design or LLD
Top SD Interview Questions

GfG School

CBSE Notes for Class 8
CBSE Notes for Class 9
CBSE Notes for Class 10

[Internship Interviews](#)

[CBSE Notes for Class 11](#)

[Competitive Programming](#)

[CBSE Notes for Class 12](#)

[Aptitude Preparation](#)

[English Grammar](#)

Commerce

UPSC

[Accountancy](#)

[Polity Notes](#)

[Business Studies](#)

[Geography Notes](#)

[Economics](#)

[History Notes](#)

[Management](#)

[Science and Technology Notes](#)

[Income Tax](#)

[Economics Notes](#)

[Finance](#)

[Important Topics in Ethics](#)

[UPSC Previous Year Papers](#)

SSC/ BANKING

Write & Earn

[SSC CGL Syllabus](#)

[Write an Article](#)

[SBI PO Syllabus](#)

[Improve an Article](#)

[SBI Clerk Syllabus](#)

[Pick Topics to Write](#)

[IBPS PO Syllabus](#)

[Write Interview Experience](#)

[IBPS Clerk Syllabus](#)

[Internships](#)

[Aptitude Questions](#)

[SSC CGL Practice Papers](#)

@geeksforgeeks , Some rights reserved