

Bitwise Operators

This lesson showcases all the different bitwise operators available in Python.

We'll cover the following ^

- Examples
- Explanation

In programming, all data is actually made up of 0s and 1s known as *bits*. Bitwise operators allow us to perform bit-related operations on values.

Operator	Purpose	Notation
&	Bitwise AND	In-fix
	Bitwise OR	In-fix
^	Bitwise XOR	In-fix
~	Bitwise NOT	Prefix
<<	Shift Bits Left	In-fix
>>	Shift Bits Right	In-fix

Examples



```
num1 = 10 # Binary value = 01010  
num2 = 20 # Binary Value = 10100
```

```
print(num1 & num2) # 00000  
print(num1 | num2) # 11110  
print(num1 ^ num2) # 11110  
print(~num1) # 1111 0101  
print(num1 << 3) # 0101 0000  
print(num2 >> 3) # 0010
```



Explanation

In **line 4**, we perform the bitwise AND. This operation takes a bit from `num1` and the corresponding bit from `num2` and performs an AND between them.

In simple terms, AND can be thought of as a multiplication between the two operands.

Now, let's visualize this example:

- `num1` is 01010 in binary and `num2` is 10100 .
- At the first step, the first binary digits of both numbers are taken:
 - 01010
 - 10100
- 0 & 1 would give 0 (again, think of it as multiplication).
- Next, we take the second digits:
 - 01010
 - 10100
- These two will once again give us 0 .
- Doing this for all pairs, we can see that the answer is 0 each time.
- Hence, the output is 00000 .

The OR operation in **line 5** will work in the same principle except that instead of multiplication, we will perform addition between the two binary numbers.

0 OR 1 gives us 1. 1 OR 1 also produces 1 (binary numbers do not go beyond 1). However, 0 OR 0 will give us 0 (0 + 0 is still 0).

Bitwise XOR and NOT will work on each bit as well. You can play with the code to get a better idea.

The bitshift operations (>> and <<) simply move the bits to either the right or the left. When a binary number is shifted, a 0 also enters at the opposite end to keep the number of the same size.

Let's suppose we have a binary number 0110 (6 in decimal). The operation we perform is 0110 >> 2 :

- 0110 >> 2
- 0011 (move one step to the right)
- 0001 (move one more step to the right)
- Operation complete

Similarly, we can move 0110 twice to the left with the following operation 0110 << 2 :

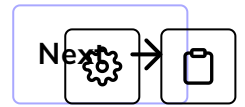
- 0110 << 2
- 01100 (move one step to the left)
- 011000 (move one more step to the left)
- Operation complete

We have studied all the operators of Python. As you'll see, they'll come in handy throughout the course.


The next lesson will teach us how we can group values together.



Logical Operators



String Operations

 Completed



Report an
Issue



Ask a Question

(https://discuss.educative.io/tag/bitwise-operators__data-types-and-variables__learn-python-3-from-scratch)