Bitwise Operators in Python

AND (&)

Operation: Performs a binary AND operation.

Examples:

1 & 1 = 1 (binary: 01 & 01 = 01)

1 & 0 = 0 (binary: 01 & 00 = 00)

0 & 1 = 0 (binary: 00 & 01 = 00)

0 & 0 = 0 (binary: 00 & 00 = 00)

Resulting integers: 1, 0, 0, 0

OR (|)

Operation: Performs a binary OR operation.

Examples:

1 | 1 = 1 (binary: 01 | 01 = 01)

1 | 0 = 1 (binary: 01 | 00 = 01)

0 | 1 = 1 (binary: 00 | 01 = 01)

0 | 0 = 0 (binary: 00 | 00 = 00)

Resulting integers: 1, 1, 1, 0

NOT (~)

Operation: Performs a binary NOT operation (bitwise inversion).

Bitwise Operators in Python

Examples:

~1 = -2 (binary: ~0001 = -0010)

~0 = -1 (binary: ~0000 = -0001)

Note: In Python, the result of ~ is -(n+1).

Resulting integers: -2, -1

XOR (^)

Operation: Performs a binary XOR operation.

Examples:

1 ^ 1 = 0 (binary: 01 ^ 01 = 00)

1 ^ 0 = 1 (binary: 01 ^ 00 = 01)

0 ^ 1 = 1 (binary: 00 ^ 01 = 01)

0 ^ 0 = 0 (binary: 00 ^ 00 = 00)

Resulting integers: 0, 1, 1, 0

Left Shift (<<)

Operation: Shifts the bits of the number to the left by the specified number of positions.

Examples:

1 << 1 = 2 (binary: 01 << 1 = 10)

Bitwise Operators in Python

1 << 2 = 4 (binary: 01 << 2 = 100)

Resulting integers: 2, 4

Right Shift (>>)

Operation: Shifts the bits of the number to the right by the specified number of positions.

Examples:

4 >> 1 = 2 (binary: 100 >> 1 = 010)

4 >> 2 = 1 (binary: 100 >> 2 = 001)

Resulting integers: 2, 1