🧠 Bitwise Operations in Python

Bitwise operators operate on the **binary (bit-level)** representation of integers. They're commonly used for:

- Low-level device control
- Data compression
- Cryptography
- Performance-critical applications

Each bitwise operation affects individual bits in a binary number.

🔷 1. `&` – Bitwise AND

python Copy Edit

```python
5 & 3 # 0101 & 0011 = 0001 → 1
```

How it works:

- Compares each bit of two numbers.
- Returns `1` only if both bits are `1`.

Bit position	5 (`0101`)	3 (`0011`)	Result
3	0	0	0
2	1	0	0
1	0	1	0
0	1	1	1

➡️ Result: `0001 = 1`

🔷 2. `|` – Bitwise OR

```python
5 | 3 # 0101 | 0011 = 0111 → 7
```

Copy Edit

How it works:

- Compares each bit.
- Returns `1` if **at least one** bit is `1`.

Bit position	5 (`0101`)	3 (`0011`)	Result
3	0	0	0
2	1	0	1
1	0	1	1
0	1	1	1

➡️ Result: `0111 = 7`

🔷 3. `^` – Bitwise XOR (Exclusive OR)

```python
5 ^ 3 # 0101 ^ 0011 = 0110 → 6
```

Copy Edit

How it works:

- Returns `1` **only if the bits are different**.

Bit position	5 (`0101`)	3 (`0011`)	Result
3	0	0	0
2	1	0	1
1	0	1	1
0	1	1	0

Bit position	5 (`0101`)	3 (`0011`)	Result

➡️ Result: `0110 = 6`

🔷 4. `~` – Bitwise NOT (One's Complement)

```python
~5 # ~00000101 = 11111010 → -6
```

How it works:

- Inverts every bit: `0 → 1` , `1 → 0`
- For integers, the result is in **two's complement** form (used for negative numbers).

Python uses **infinite-length signed integers**, so:

```python
~n == -n - 1
```

So:

```python
~5 = -6
```

🔷 5. `<<` – Left Shift

```python
5 << 1 # 00000101 << 1 = 00001010 → 10
```

How it works:

- Shifts bits to the **left**, adding `0` s at the right.
- Effectively **multiplies the number by 2^n**.

python Copy Edit

```
5 << 2 = 5 × 2² = 20
```

🔷 6. >> – Right Shift

python Copy Edit

```
5 >> 1 # 00000101 >> 1 = 00000010 → 2
```

How it works:

- Shifts bits to the **right**, dropping bits off the end.
- Effectively **divides the number by 2^n**, discarding remainder.

python Copy Edit

```
5 >> 2 = 5 // 4 = 1
```

🔧 Example with Binary Output in Python

python Copy Edit

```
a = 5 # 0b0101 b = 3 # 0b0011 print(bin(a & b)) # 0b1 → AND print(bin(a | b)) # 0b111 → OR print(bin(a ^ b)) # 0b110 → XOR print(bin(~a)) # -0b110 → NOT (Two's complement) print(bin(a << 2)) # 0b10100 → Left shift (×4 = 20) print(bin(a >> 1)) # 0b10 → Right shift (//2 = 2)
```

🔍 Summary Table

Operator	Name	Description
&	AND	1 only if both bits are 1

Operator	Name	Description
`	`	OR
^	XOR	1 if bits are different
~	NOT	Invert all bits
<<	Left Shift	Shift bits left (multiply by 2^n)
>>	Right Shift	Shift bits right (divide by 2^n)