Bitwise Operators in Python – Full Explanation

■ Complement Operator (~)

Example: ~10

Explanation: Binary of 10 → 00001010
Complement flips all bits → 11110101
Result (decimal) = -11

Output: -11

■ AND Operator (&)

Example: 10 & 11

Explanation: Bitwise AND compares each bit; only 1 if both bits are 1.
Binary 10→1010, 11→1011 → Result 1010 (10)

Output: 10

■ OR Operator (|)

Example: 10 | 11

Explanation: Bitwise OR sets bit to 1 if any bit is 1.
Binary 10→1010, 11→1011 → Result 1011 (11)

Output: 11

■ XOR Operator (^)

Example: 15 ^ 20

Explanation: Bits different → 1.
15→01111, 20→10100 → Result 11011 (27)

Output: 27

■ Left Shift (<<)

Example: 11<<1, 11<<2, 11<<3

Explanation: Shifts bits left and fills with zeros.
Results: 22, 44, 88

Output: 22, 44, 88

■ Right Shift (>>)

Example: 11>>1, 11>>2, 11>>3, 11>>4

Explanation: Shifts bits right and removes bits from the right.
Results: 5, 2, 1, 0

Output: 5, 2, 1, 0

■ Summary Table

Operator	Symbol	Description	Example	Output
Complement	~	Flips all bits	~10	-11
AND	&	Both bits 1 → 1	10 & 11	10
OR	|	Any bit 1 → 1	10 | 11	11
XOR	^	Bits different → 1	15 ^ 20	27
Left Shift	<<	Shift bits left	11 << 2	44
Right Shift	>>	Shift bits right	11 >> 3	1

■ **Conclusion:** Bitwise operators are used for low-level operations, masking, encryption, and optimization. They work directly on binary bits (0s and 1s).