Bitwise operators

Bitwise operators are used to perform operations on **binary (bit-level) representations of integers**. They manipulate individual bits (0 or 1) of numbers.

- **Types of Bitwise Operators**

Operator	Name	Description
&	Bitwise AND	Returns 1 if both bits are 1
|	Bitwise OR	Returns 0 only if both bits are 0
^	Bitwise XOR	Returns 1 if bits are different
~	Bitwise NOT	Inverts all bits (1 → 0, 0 → 1)
<<	Left Shift	Shifts bits to the left (adds zeros on the right)
>>	Right Shift	Shifts bits to the right (drops bits on the right)

- **Syntax**

result = operand1 <operator> operand2

For NOT (~):

result = ~operand

- **Examples**

a = 10 # (1010 in binary)

b = 4 # (0100 in binary)

print(a & b) # 0 -> (1010 & 0100 = 0000)

print(a | b) # 14 -> (1010 | 0100 = 1110)

print(a ^ b) # 14 -> (1010 ^ 0100 = 1110)

print(~a) # -11 -> Bitwise NOT (2's complement)

print(a << 1) # 20 -> (1010 << 1 = 10100)

print(a >> 1) # 5 -> (1010 >> 1 = 0101)

- **Important Points**

1. Works **only on integers** (floats, strings, etc. are not supported).
2. Negative numbers use **2's complement representation** in bitwise operations.
3. Left shift (<<) multiplies by 2^n, Right shift (>>) divides by 2^n.
    - Example: 10 << 1 = 20 (10 × 2), 10 >> 1 = 5 (10 ÷ 2).
4. ~x is equivalent to -(x+1) in Python (due to 2's complement).
5. Bitwise operators are commonly used in:
    - **Low-level programming**
    - **Cryptography**
    - **Masking & Flag operations**
    - **Data compression**