Bitwise Operators

Bitwise operators in Python are used to perform operations on individual bits of integers. These operators work at the binary level, manipulating the bits that represent the numbers.

Here are the bitwise operators in Python:

1. `&` (Bitwise AND): This operator performs a bitwise AND operation on corresponding bits of two numbers.

Example:

```
a = 5   # binary representation: 0101
b = 3   # binary representation: 0011

result = a & b
print(result)  # Output: 1
```

Explanation:

- `a` is represented in binary as `0101`.
- `b` is represented in binary as `0011`.
- Performing a bitwise AND operation on these two binary representations gives `0001`, which is `1` in decimal.

2. `|` (Bitwise OR): This operator performs a bitwise OR operation on corresponding bits of two numbers.

Example:

```
a = 5   # binary representation: 0101
b = 3   # binary representation: 0011

result = a | b
print(result)  # Output: 7
```

Explanation:

- `a` is represented in binary as `0101`.
- `b` is represented in binary as `0011`.
- Performing a bitwise OR operation on these two binary representations gives `0111`, which is `7` in decimal.

3. `^` (Bitwise XOR): This operator performs a bitwise XOR (exclusive OR) operation on corresponding bits of two numbers.

Example:

```
a = 5   # binary representation: 0101
b = 3   # binary representation: 0011

result = a ^ b
print(result)  # Output: 6
```

Explanation:

- `a` is represented in binary as `0101`.
- `b` is represented in binary as `0011`.
- Performing a bitwise XOR operation on these two binary representations gives `0110`, which is `6` in decimal.

4. `~` (Bitwise NOT): This operator inverts the bits of a number, turning `0` to `1` and `1` to `0`.

Example:

```
a = 5   # binary representation: 0101


result = ~a
print(result)  # Output: -6
```

Explanation:

- `a` is represented in binary as `0101`.
- Performing a bitwise NOT operation inverts the bits to `1010`, which is `-6` in decimal due to two's complement representation.

5. `<<` (Left Shift): This operator shifts the bits of a number to the left by a specified number of positions.

Example:

```
a = 5   # binary representation: 0101
```

```
result = a << 1
print(result)  # Output: 10
```

Explanation:

- Shifting `0101` one position to the left results in `1010`, which is `10` in decimal.

6. `>>` (Right Shift): This operator shifts the bits of a number to the right by a specified number of positions.

Example:

```
a = 5   # binary representation: 0101

result = a >> 1
print(result)  # Output: 2
```

Explanation:

- Shifting `0101` one position to the right results in `0010`, which is `2` in decimal.

These examples demonstrate the basic usage of bitwise operators in Python. Remember that bitwise operations are typically used in low-level programming or situations requiring bit-level manipulation.