**Bitwise**

def is\_even(num):

# Check if the least significant bit is 0 (even)

return num & 1 == 0

def is\_odd(num):

# Check if the least significant bit is 1 (odd)

return num & 1 == 1

def swap\_numbers(a, b):

# Swap two numbers using XOR without using a temporary variable

a = a ^ b

b = a ^ b

a = a ^ b

return a, b

def multiply\_by\_two(num):

# Left shift is equivalent to multiplying by 2

return num << 1

def divide\_by\_two(num):

# Right shift is equivalent to dividing by 2

return num >> 1

**Benefits:**

**Efficient Even/Odd Checking:**

Using the bitwise AND operation (& 1) is a more efficient way to check if a number is even or odd compared to the modulo operation (% 2). Bitwise operations are often faster at the hardware level.

**Swapping Numbers Without Temporary Variable:**

The XOR swap algorithm (a = a ^ b; b = a ^ b; a = a ^ b) allows swapping two variables without using a temporary variable. This can be more memory-efficient and avoids the need for an additional storage location.

**Multiplying/Dividing by Two:**

Left shifting (<<) and right shifting (>>) are more efficient than traditional multiplication and division by 2. They are equivalent operations at the bit level and can be faster, especially for large numbers.

**Bitwise Operations for Bit-Level Manipulation:**

Bitwise operations are crucial for tasks that involve bit-level manipulation, such as setting, clearing, or toggling specific bits within an integer. This is commonly used in low-level programming, network protocols, and certain algorithm optimizations.

**Performance Optimization:**

Bitwise operations are often more efficient at the hardware level, making them useful for performance-critical applications. In scenarios where execution speed is crucial, using bitwise operators can lead to more optimized code.

**Compact Code:**

Bitwise operations often allow for more concise and compact code compared to equivalent operations using traditional arithmetic or logical operators. This can improve code readability and maintainability.