Bitwise Questions

**Things to know:**

* Binary representation
  + General knowledge, conversion from decimal 🡪 binary and vice-versa
* Bitwise operators
  + Much less common, but good to be aware about

**Binary**

The regular numbers we use in day-to-day life are in decimal system (base 10). Each digit goes from 0-9.

Binary is base 2. Each digit is either a 0 or a 1.

The number 13 in decimal:

1\*10^1 + 3\*10^0 = 13

The number 13 in binary:

1\*2^3 + 1\*2^2 + 0\*2^1 + 1\*2^0 = 1101

Python code to go from decimal to binary:

def dec\_to\_binary(num):

    if num > 1:

        dec\_to\_binary(num // 2)

    print(num % 2, end = '')

**Bitwise Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Usage** |
| & | AND - 1 if both bits are set, 0 otherwise | 010**1** (decimal 5)  AND 001**1** (decimal 3)  = 000**1** (decimal 1) | Can be useful with checking if certain bits are set (see bit masking) |
| | | OR – 1 if either bit is set, 0 otherwise | 0**101** (decimal 5)  OR 0**011** (decimal 3)  = 0**111** (decimal 7) | Can be useful with setting certain bits |
| ^ | XOR – 1 if the bits do not match, 0 otherwise | 0**10**1 (decimal 5)  XOR 0**01**1 (decimal 3)  = 0**11**0 (decimal 6) | Can be useful with toggling bits repeatedly |
| ~ | Complement – flip the bit | ~ 0011 (decimal 3)  = 1100 (decimal 12) |  |

**Practice Questions**

1. Given an integer, write a function to return the new integer that would result from reversing the bits.

ex: reverse\_bits(4) 🡪 0100 reversed is 0010 🡪 return 2

1. Given two integers, return the number of positions where they differ in bits (called the *Hamming Distance)*

ex: differing\_bits(2, 4) 🡪 0010 and 0100 differ in 2 locations 🡪 return 2

1. Write a function using bitwise operators to determine if an integer is a power of two.

ex: is\_power\_of\_two(16) 🡪 True

hint: what does the binary representation of a power of two look like?