**探寻Python(C++)中的 ~ （按位取反符）究竟是怎么取的**

**一：原码、反码、补码与计算机是如何进行运算的**

我们首先来了解一下有关原码、反码、补码与计算机如何进行运算的问题，当然如果对这块儿比较熟悉，可以直接跳过，直接看下面的计算机如何进行按位取反的，我之所以在介绍计算机如何进行按位取反的内容之前罗嗦那么多，实际上也是为了大家更好地去理解和掌握计算机如何进行按位取反。

**1.原码、反码、补码：**

**补码是计算机处理有符号数的常用方法**，因为补码在进行加减运算时可以将符号位一起直接参与运算，无需特别对符号位判断后再进行运算，从而简化了运算规则，提高了运算速度。

**原码**就是符号位加上真值的绝对值， 即用第一位表示符号， 其余位表示值。

**反码**的表示方法是:正数的反码是其本身；负数的反码是在其原码的基础上，符号位不变，其余各位逐位取反得到的。

**补码**的表示方法是:正数的补码就是其本身；负数的补码是在其原码的基础上，符号位不变， 其余各位逐位取反， 末位加一得到的。 (即在反码的基础上+1)

因此：正数的原码=正数的反码=正数的补码

另外，要注意到：

**任何一个数的反码的反码就是其原码本身**

**任何一个数的补码的补码就是其原码本身**（这句话是没毛斌的，有兴趣的话可以自己验证）

**2.加减运算**

我们先来看一下定点加法运算（计算机数值中有定点和浮点两种表示方法，这里为了简化问题，只讨论一下定点的运算）

加法运算公式为：[X+Y]补=[X]补+[Y]补

例题1：已知X=-110 0100B,Y=11 0010B（字长为8），求X+Y。

解：因为:      [X]原=1110 0100B、[Y]原=0011 0010B

       所以：    [X]补=1001 1100B、[Y]补=0011 0010B

       根据公式[X+Y]补=[X]补+[Y]补=1001 1100B+0011 0010B=1100 1110B

       由所求得的补码得出X+Y=[[X+Y]补]补=1011 0010B

注意：

X+Y=[[X+Y]补]补是根据上面提到的**任何一个数的补码的补码就是其原码本身**得来的。

上面就是计算机在求解定点数加法运算的过程，先把原码转成补码，然后再进行运算（这时得到的结果仍是补码），最后根据**任何一个数的补码的补码就是其原码本身**再进行一次求补码的运算以求出结果的原码。

做完例题1后我们会对机器是如何进行运算的，有了大致的了解，那么下面再做一个题强化一下：

减法运算公式为： [X-Y]补=[X]补-[Y]补=[X]补+[-Y]补

例题2：已知X=110 0100B,Y=11 0010B（字长为8），求X-Y。

解：因为:      [X]原=0110 0100B,[Y]原=0011 0010B,[-Y]原=1011 0010B

       所以       [X]补=0110 0100B,[-Y]补=1100 1110B

[X-Y]补=[X]补+[-Y]补=01100100B+1100 1110B=0011 0010B（因字长为8位，所以应舍弃最高位的进位）

由所求得的补码得出X-Y=0011 0010B（正数的补码、反码均与原码相同，一个数的补码的补码就是其原码本身）

注意：计算机在求解定点数运算问题时，先把原码转成补码，然后再进行运算（这时得到的结果仍是补码），最后再根据运算中得到的补码求出最终结果的原码。

**二：~是按位取反符，具体取反过程为：**

例1：求~1

1的原码：0000 0001

1的补码：0000 0001

对1的补码按位取反之后得：1111 1110（这是一个新的补码）

然后，求解上步中的补码的原码：1000 0010

最后转成10进制结果为-2

例2：求~2

-2的原码为：1000 0010

-2的补码为：1111 1110

对-2的补码按位取反后得：0000 0001（这是一个新的补码）

然后，求解上步中的补码的原码：0000 0001（正数的补码、反码与原码相同）

最后转成10进制结果为1

注意：**按位取反，是指对某数的补码进行按位取反。**

另外，也可以简单地记一下：按位取反运算符是对数据的补码的每个二进制位进行取反，即把1变为0，把0变为1 。~x 类似于 -x-1。