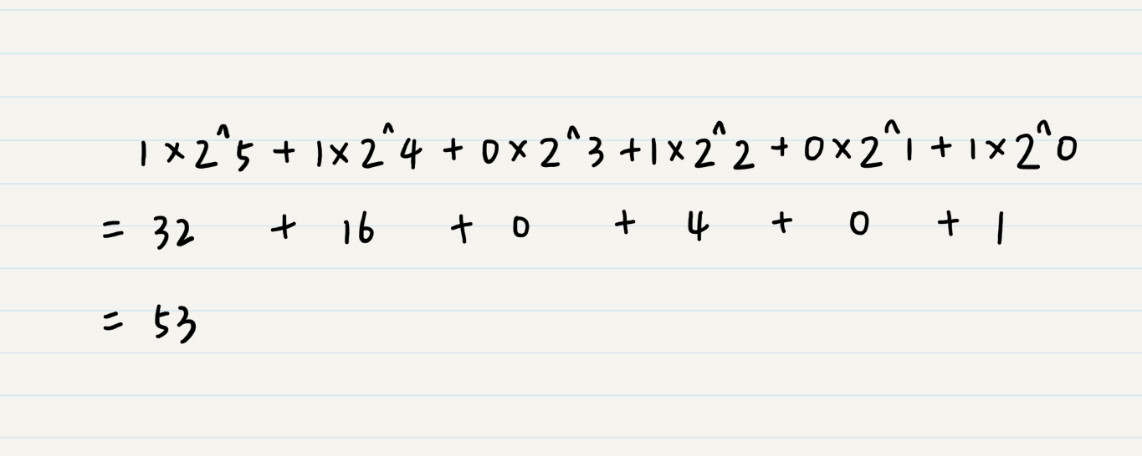
二进制 110101 -> 二进制的数位就是 2^n 的形式.



**计算机为什么使用二进制？**

我觉得，计算机使用二进制和现代计算机系统的硬件实现有关。组成计算机系统的逻辑电路通常只有两个状态，即开关的接通与断开。

断开的状态我们用“0”来表示，接通的状态用“1”来表示。由于每位数据只有断开与接通两种状态，所以即便系统受到一定程度的干扰时，它仍然能够可靠地分辨出数字是“0”还是“1”。因此，在具体的系统实现中，二进制的数据表达具有抗干扰能力强、可靠性高的优点。

相比之下，如果用十进制设计具有 10 种状态的电路，情况就会非常复杂，判断状态的时候出错的几率就会大大提高。

另外，二进制也非常适合逻辑运算。**逻辑运算中的“真”和“假”，正好与二进制的“0”和“1”两个数字相对应**。逻辑运算中的加法（“或”运算）、乘法（“与”运算）以及否定（“非”运算）都可以通过“0”和“1”的加法、乘法和减法来实现。

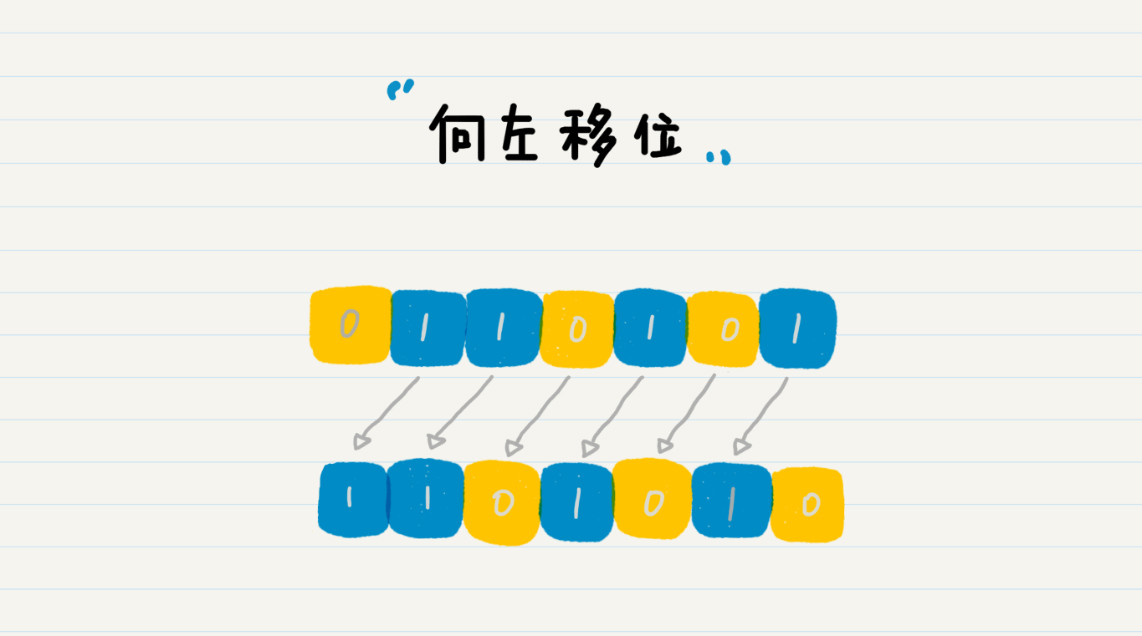
**计算机语言中针对二进制的位操作。**

这里的位操作，也叫作位运算，就是直接对内存中的二进制位进行操作。常见的二进制位操作包括向左移位和向右移位的移位操作，以及“或”“与”“异或”的逻辑操作。

**向左移位**

二进制 110101 向左移一位，就是在末尾添加一位 0，因此 110101 就变成了 1101010。请注意，这里讨论的是数字没有溢出的情况。

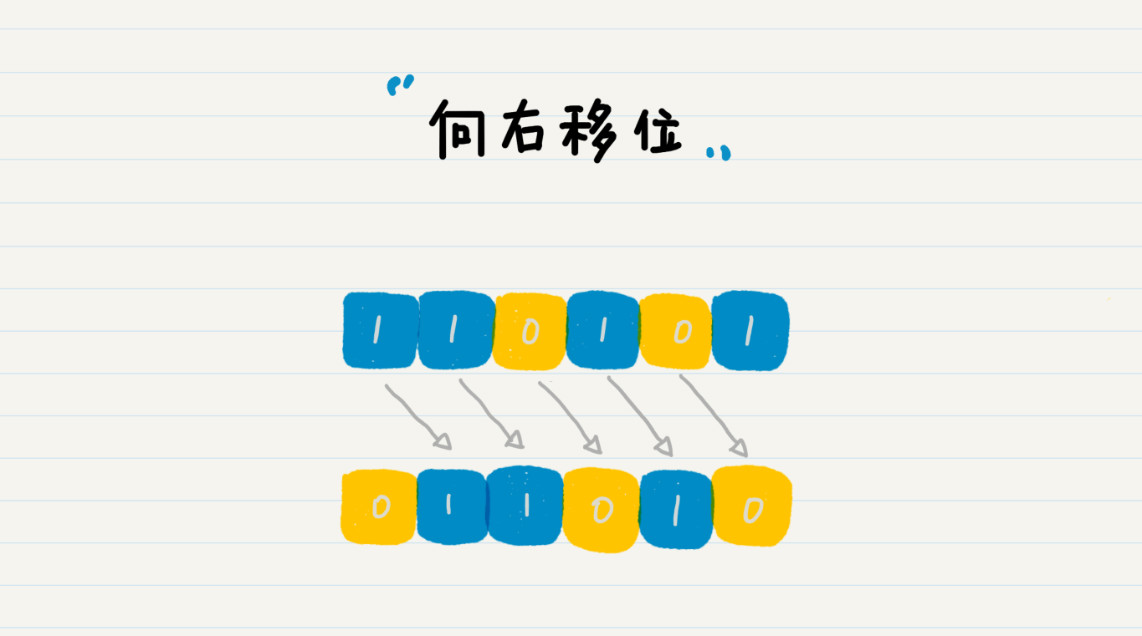
所谓数字溢出，就是二进制数的位数超过了系统所指定的位数。目前主流的系统都支持至少 32 位的整型数字，而 1101010 远未超过 32 位，所以不会溢出。如果进行左移操作的二进制已经超出了 32 位，左移后数字就会溢出，需要将溢出的位数去除。



如果将 1101010 换算为十进制就是106，106 正好是 53 的 2 倍。所以，我们可以得出一个结论：**二进制左移一位，其实就是将数字翻倍。**

**向右移位**

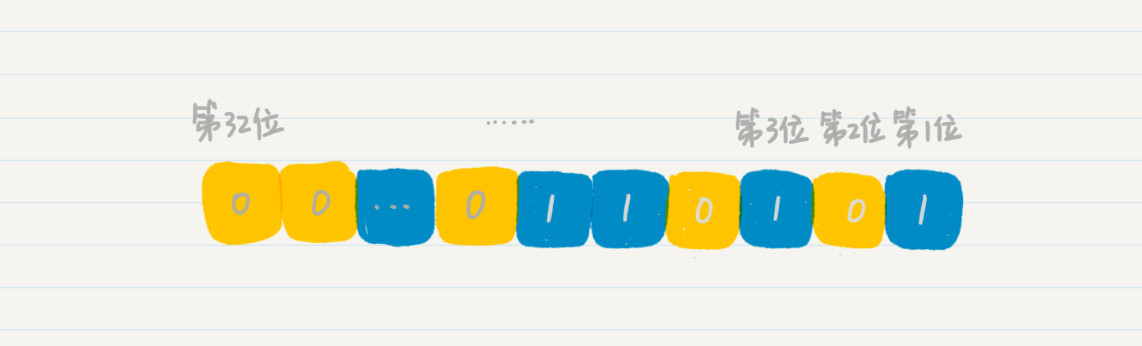
二进制 110101 向右移一位，就是去除末尾的那一位，因此 110101 就变成了 11010（最前面的 0 可以省略）。我们将 11010 换算为十进制，就是 26，正好是 53 除以 2 的整数商。所以**二进制右移一位，就是将数字除以 2 并求整数商的操作**。



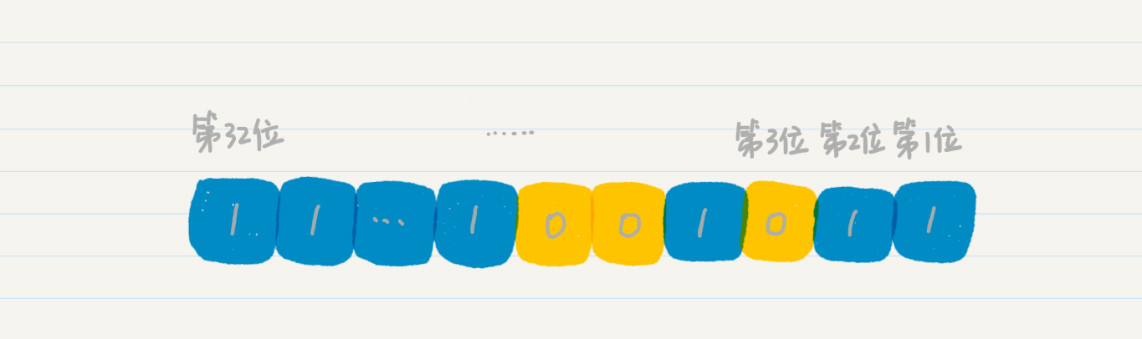
移位 1 次相当于乘以或除以 2，而移位 3 次就相当于乘以或除以 8（即 2 的 3 次方）。

左移位是 <<，那右移位为什么是 >>> 而不是 >> 呢？

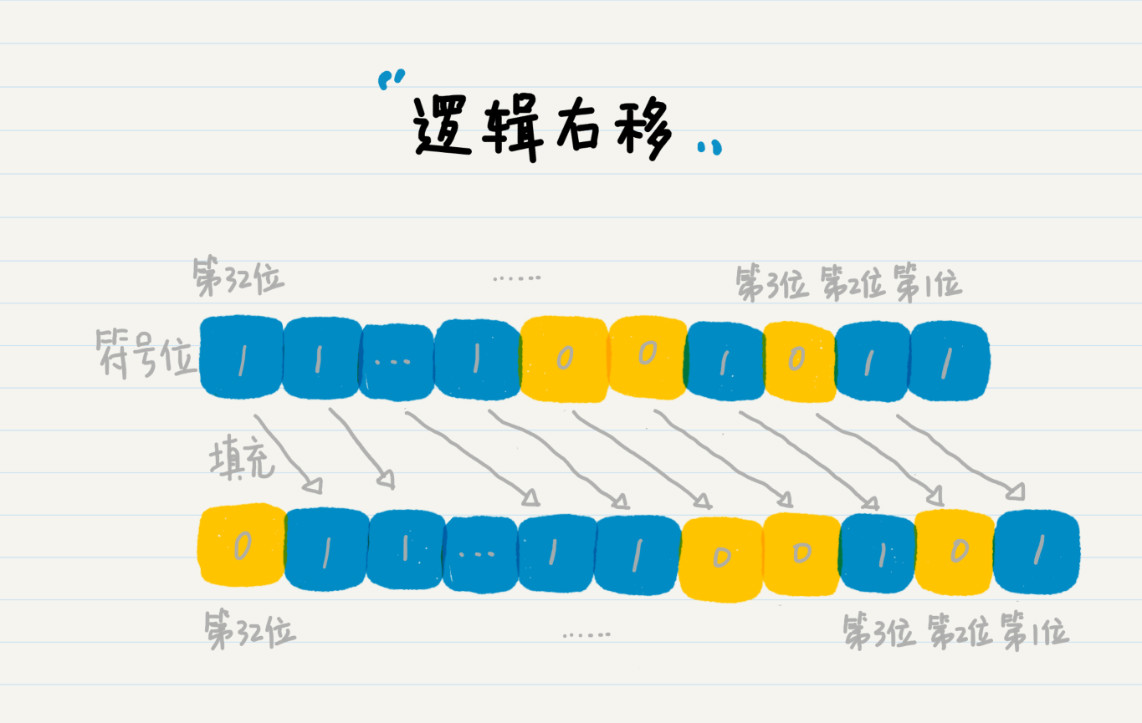
**当符号位为 0 时，表示该数值为正数**；当符号位为 1 时，表示该数值为负数。我们以 32 位 Java 为例，数字 53 的二进制为 110101，从右往左数的第 32 位是 0，表示该数是正数，只是通常我们都将其省略。



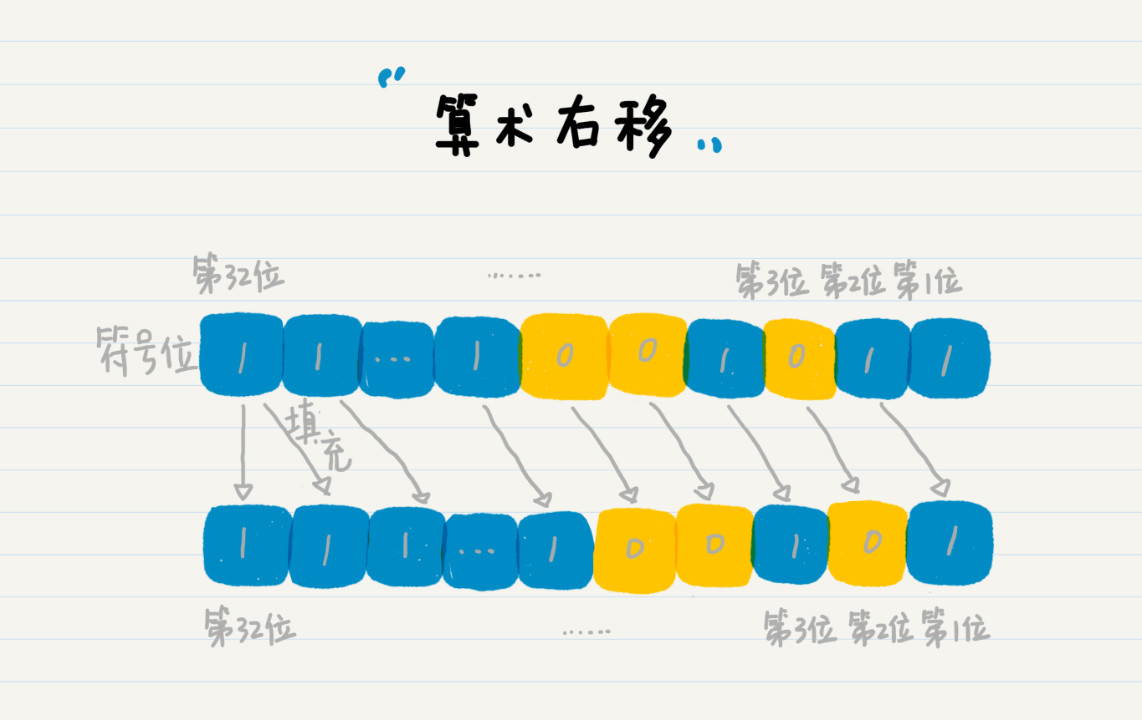
如果数字是 -53 呢？那么第 32 位就不是 0，而是 1。请注意我这里列出的是补码



那么这个时候向右移位，就会产生一个问题：对于符号位（特别是符号位为 1 的时候），我们是否也需要将其右移呢？因此，Java 里定义了两种右移，**逻辑右移和算术右移**。**逻辑右移 1 位，左边补 0 即可**。

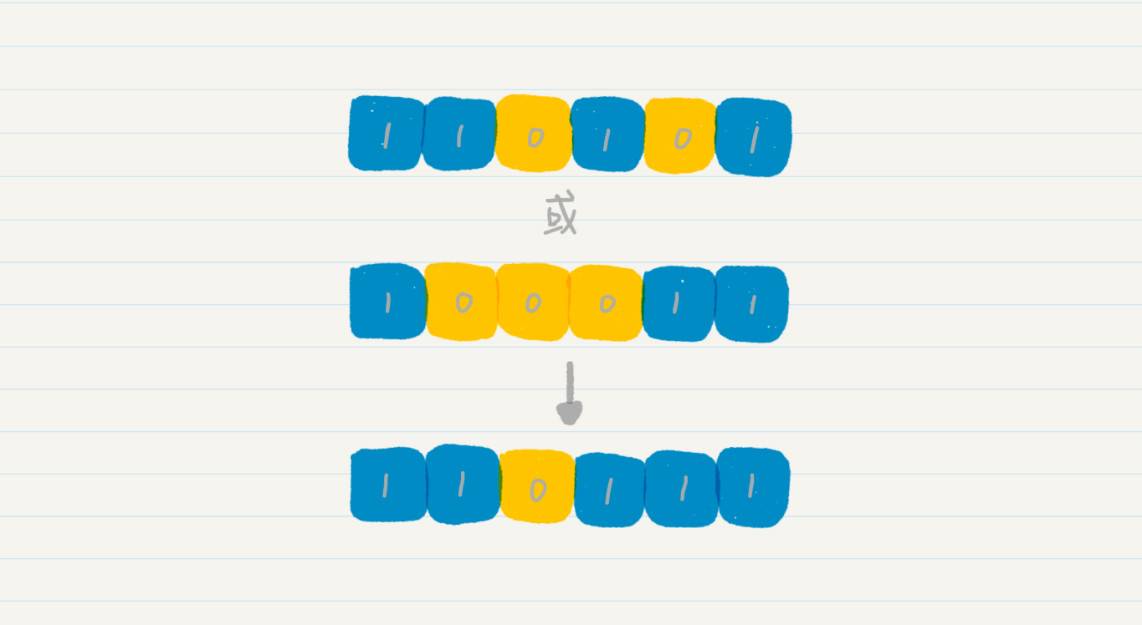


算术右移时保持符号位不变，除符号位之外的右移一位并补符号位 1。补的 1 仍然在符号位之后。



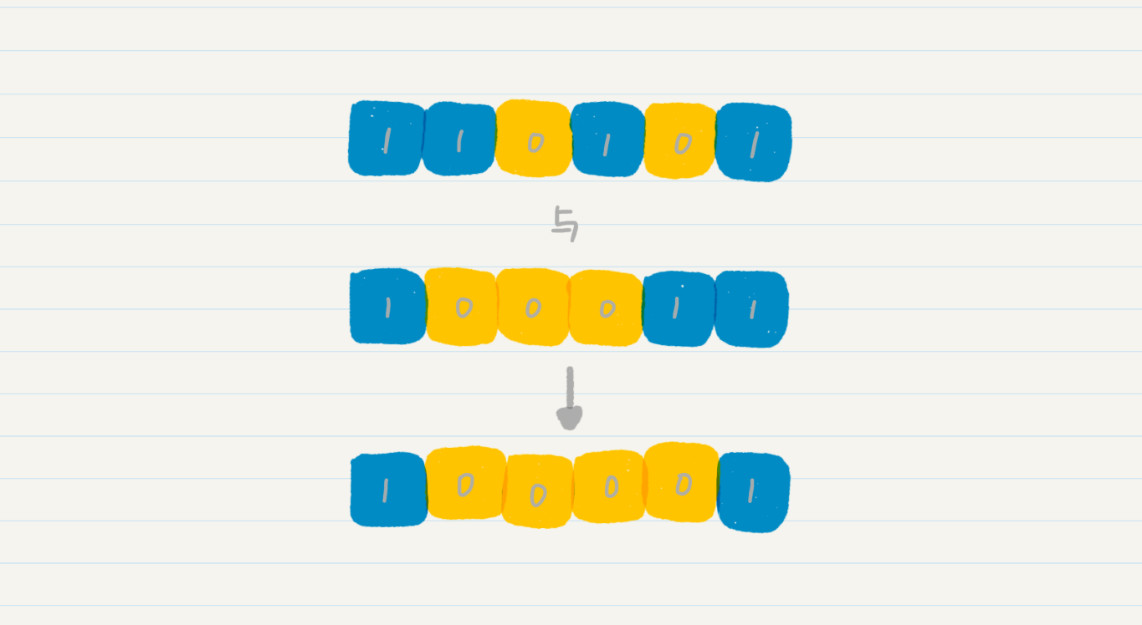
位的“或”

逻辑“或”的意思是，参与操作的位中只要有一个位是 1，那么最终结果就是 1，也就是“真”。如果我们将二进制 110101 和 100011 的每一位对齐，进行按位的“或”操作，就会得到 110111。



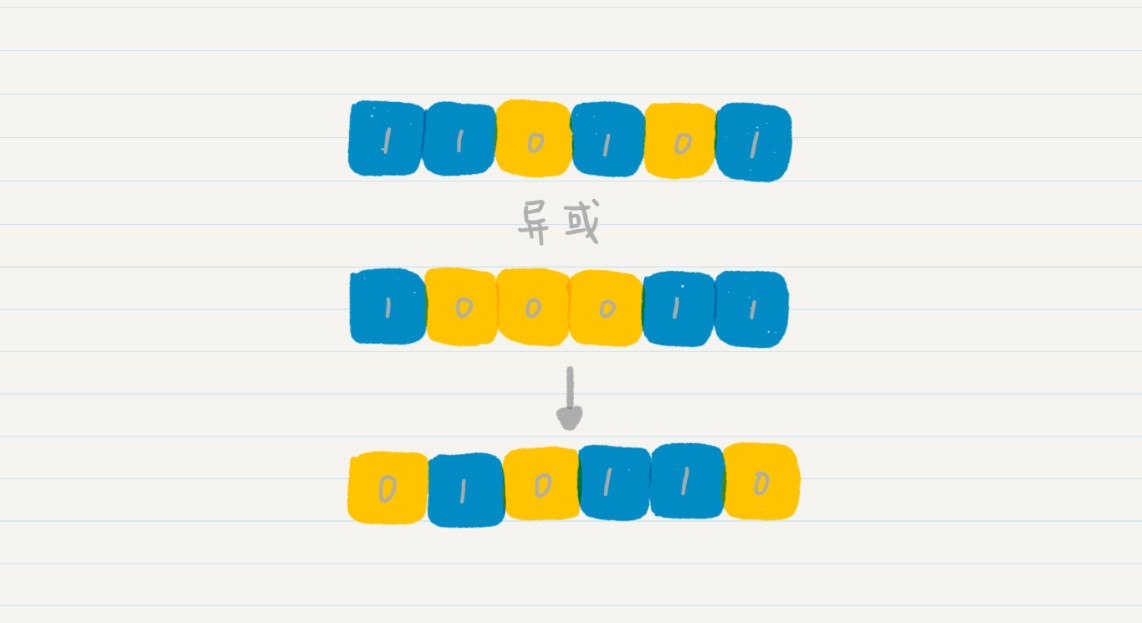
位的“与”

同理，我们也可以针对位进行逻辑“与”的操作。“与”的意思是，参与操作的位中必须全都是 1，那么最终结果才是 1（真），否则就为 0（假）。如果我们将二进制 110101 和 100011 的每一位对齐，进行按位的“与”操作，就会得到 100001。



位的“异或”

逻辑“异或”和“或”有所不同，它具有排异性，也就是说如果**参与操作的位相同，那么最终结果就为 0（假），否则为 1（真）**。**所以如果要得到 1，参与操作的两个位必须不同，这就是此处“异”的含义**。我们将二进制 110101 和 100011 的每一位对齐，进行按位的“异或”操作，可以得到结果是 10110。



“异或”操作的本质其实就是，所有数值和自身进行按位的“异或”操作之后都为 0。而且要通过“异或”操作得到 0，也必须通过两个相同的数值进行按位“异或”。这表明了**两个数值按位“异或”结果为 0**，**是这两个数值相等的必要充分条件**，可以作为判断两个变量是否相等的条件。

Java 中使用|表示按位的“或”，& 表示按位“与”，^ 表示按位“异或”。

