二进制转化为十进制

注：以下正文部分用了几种不同颜色的字体标注，特此说明：

灰色字体 – 如段落**标题及括号中的内容**，为解释性信息，不是正文内容，不需要读出来

红色字体 – 需要重点强调的地方

蓝色字体 – 为讲解思路

完整代码附在文章最后

===============================================

一、从介绍什么是十进制和二进制开始

从小妈妈就教我们数数，来来来，伸出10个手指头，1、2、3、4、5、6、7、8、9、10，凑够10就进1。凑够10个10再进1，就是100。满10进1的进位制度就是十进制，这种计数方法我们已经司空见惯了。

但在计算机系统的信息中都只用两个数值来储存，那就是0和1，早期计算机是以电晶体管的开关状态来表示数值，开代表1，关代表0，凑够2个就进1，借1当2，基数是2，这就是二进制，所以我们称计算机系统是二进制系统。

举个栗子：

每10个鸡蛋装一盒，十盒鸡蛋装一箱，这是十进制

每2个西瓜装一盒，每两盒装一箱，这就是二进制

我们在数外加上括号，然后在括号的右下角加上2来表示二进制数，就是酱事儿的：

(1111)2  (1010101)2

(1111)2（读幺幺幺幺）可不是一千一百一十一，而是代表二进制的1111（读幺幺幺幺），二进制满2进1，所以从后往前1111第一位代表1个1，1111这个位置1个顶2个，满2个就再向前进一位；那1111这个位置1个就代表2个2，也就是2的平方；同理1111这个位置1个代表2个2的平方，就是2的3次方

其实二进制的(1111)2就是十进制的15

那15是怎么来的？11111又是十进制的多少呢？10101011010101又是十进制的多少呢？。。。。。

二、进制换算

厄…..这个问题嘛，咱们得先找到二进制转换十进制的规律——

先对比来看下你最熟悉的十进制的1111（读一千一百一十一）：

1111 个位的1就是1

1111 十位的1则代表10，分解一下就是1\*10

1111 百位的1则代表100，分解一下就是1\*100

1111 千位的1则代表1000，分解一下就是1\*1000

虽然都是1，但在不同的位置代表的大小却不一样，数制中每一固定位置对应的单位值称为位权，可不是“吃饱了来一瓶”那个味全哦, 数字乘对应位权再相加就等于这个十进制数

所以，一千一百一十一就等于：

1\*103+1\*102+1\*101+1\*100 🡪 1111

同理，二进制的1111（读幺幺幺幺）：

1 1 1 1

23 22 2 1

1111 第一位的1就是1，分解一下就是1\*20

1111 第二位的1则代表2，分解一下就是1\*21

1111 第三位的1则代表2\*2，分解一下就是1\*22

1111 第四位的1则代表2\*2\*2，分解一下就是1\*23

所以二进制的幺幺幺幺就等于：

1\*23+1\*22+1\*21+1\*20 = 15

这下我们知道了二进制数转换成十进制数的方法就是：二进制数从右向左每位数乘以2的次方（从0开始，从右向左依次+1），然后相加求和即可

好了，现在我们不但知道了15是怎么来的，还可以算出11111、10101011010101…又分别是十进制的多少。不过，别着急，反复做同样的事情很无聊，还是让Python代码来帮我们继续做吧

三、代码实现

1、整体介绍

我们现在就来做一个二进制转化为十进制的程序，跟把大象塞进冰箱里一样，需要三步：

第一步：输入(input)--告诉程序你的二进制数

第二步：处理(process) --将二进制数转化为十进制数

第三步：输出(output) –将十进制数的结果打印到屏幕上

2、先讲转换处理

1）讲讲变量

既然需要输入输出，计算机就得找个地方把它们存上。在Python中，如果希望计算机记住某个东西，以便你以后使用，那就要给这个“东西”起个名字，这个名字就是变量

在我们的程序里我们先定义一个叫binary的变量来代表输入的二进制数，一个叫decimal的变量代表要输出的十进制数

binary —> 二进制变量

decimal—> 十进制变量

这里还是先用我们最喜欢的1111（读幺幺幺幺）来举栗子：

binary = '1111'  
decimal = 0

假设你在键盘上敲了一串字符1111，把1111赋值给代表二进制的变量binary。目前我们还不知道decimal它是几，所以先假定它是0，咦，既然不知道它是几，为什么还要让他等于0呢？

decimal = 0是变量的初始化，Python没有强制要求所有的变量在定义的时候做初始化，但是所有的变量在使用之前，也就是出现在赋值号的右边之前，应该被赋值一次，否则就可能出错。

2）从公式转化成Python语句

上面我们已经知道了二进制转化成十进制的方法：从右向左每位数乘以2的次方，然后相加求和，所以decimal可以通过以下算式得到：

decimal= 1\*23+1\*22+1\*21+1\*20

这里，每一位上的数都要和2的次方相乘，那么问题来了，在Python里如何表示几次方呢？

来来来，认识一下这个方法---pow()，写作：

pow(x,y)—它的作用是返回 xy（x的y次方） 的值

所以，这一句在Python中要写成：

decimal= 1\*pow(2,3)+1\*pow(2,2)+1\*pow(2,1)+1\*pow(2,0)

在这里有没有发现一个规律，从左往右：

1111 第1位上的1 对应的次方是3

1111 第2位上的1 对应的次方是2

1111 第3位上的1 对应的次方是1

1111 第4位上的1 对应的次方是0

1+3、2+2、3+1、4+0 都等于4，也就是1111的长度，所以在这里：

**位数+次方=二进制数的长度**

但Python是0-base的语言，就是标记数组或者字符串的下标从0开始，据Python语言的创始人吉多.范罗苏姆说这样做是为了让Python的切片语法看起来更优雅。切片我们先用不着，优雅不优雅我们也先不说，单说我们这里：

1111从左开始的第1位，在Python中其实是第0位，也就变成了：

1111 第0位上的1 ，对应的次方是3

1111 第1位上的1 ，对应的次方是2

1111 第2位上的1 ，对应的次方是1

1111 第3位上的1 ，对应的次方是0

位数和次方的对应关系也成了

**位数+次方=二进制数的长度-1**

**由此得出：**

**次方=二进制数的长度-1-位数**

所以，1111的长度是4，减去1就是3，那么decimal也就可以表示成：

decimal= 1\*pow(2,3-0)+1\*pow(2,3-1)+1\*pow(2, 3-2)+1\*pow(2, 3-3)

在这里要注意了，1111是代表用户从键盘上输入的值，所以他是一串字符串，而不是数字。不信？你可以试试，Python提供了一个叫type()的函数，它可以明确的告诉我们变量的类型

print(type(binary))

我们在编辑器的运行窗口里可以得到：<class 'str'>这样的提示，’str’就是代表字符串类型。字符串可以通过索引--也就是刚才我们说的那个从0开始的编号，来获取指定的字符，于是：

1111 第0位上的1 ，可以表示成binary[0]

1111 第1位上的1 ，可以表示成binary[1]

1111 第2位上的1 ，可以表示成binary[2]

1111 第3位上的1 ，可以表示成binary[3]

（将decimal等式中的1用变量替换掉）也就是下面这样：

decimal= binary[0]\*pow(2,3-0)+ binary[1]\*pow(2,3-1)+ binary[2]\*pow(2,3-2)+ binary[3]\*pow(2,3-3)

3）循环

如果用i来代表当前数的位置，那么第**i**位乘以2的几次方就可以表示成：

binary[**i**] \* pow(2, len(binary) - 1 - **i**)

要把每一位的计算结果相加才能得到最后的十进制数，所以：

decimal = binary[**i**] \* pow(2, len(binary) - 1 - **i**)

+ binary[**i**] \* pow(2, len(binary) - 1 - **i**)

+ binary[**i**] \* pow(2, len(binary) - 1 - **i**)

+ binary[**i**] \* pow(2, len(binary) - 1 - **i**)

咦，每次加的都一样啊？！加了4次，i分别代表0、1、2、3，就是一个重复了4次的循环啊！

bingo，就是循环，重复一定次数的循环称为计数循环（counting loop）,也叫for循环，在Python的语法里长这样：

for i in [0,1,2,3]:  
 decimal += binary[**i**] \* pow(2, len(binary) - 1 - **i**))

因为是4次，从0开始，所以in后面是一个从0到3的数组[0,1,2,3]，可以用range(0,4)代替，这里4是1111的长度，也就是len(binary)，range()提供了一个数字列表，从给定的第一个数开始，在给定的最后一个数之前结束

因为每次加的内容都一样，所以这里decimal累加求和使用了+=自增运算符(incrementing)，类似于a = a+1可以简写成a += 1

for i in range(0, len(binary)):

decimal += binary[**i**] \* pow(2, len(binary) - 1 - **i**))

4）优化一下

为了让程序看起来更清晰简洁，我们再定义两个变量：power代表2的几次方，binaryLen代表程序中二进制数的长度

power = 0----------2的几次方  
binaryLen = len(binary)------------ 二进制数的长度

于是：

power = binaryLen - 1 – **i**

decimal += binary[i] \* pow(2, power)

3、输出

程序进行到这里，将二进制转化成十进制的计算基本就处理完了，那到底对不对呢？还需要验证一下，让结果显示在屏幕上看看

binary = '1111'  
decimal = 0

power = 0  
binaryLen = len(binary)  
  
for i in range(0, binaryLen):  
 power = binaryLen - 1 - i  
 decimal += binary[i] \* pow(2, power)

print("对应的十进制数是：", decimal)

TypeError: unsupported operand type(s) for +=: 'int' and 'str'

What?居然报错了！我们不是明明白白的一步步写出来的嘛，为什么会这样？

别急，我们先来看看错误的提示信息，它告诉我们+=自增运算符不支持将int整型和str字符串连接在一起。因为decimal的初始值是0，所以它是整型Int，而binary[i]是字符串，\*乘以2的次方后仍然还是字符串。

这就好比我们买了2盒酸奶3个苹果，你既不能说买了5个苹果，也不能说买了5盒酸奶，不能把完全不同的两个东西加在一起。

所以这里我们需要把binary[i]也转化成整型int(binary[i])，这样计算出的类型一致就可以相加了

binary = '1111'  
decimal = 0  
power = 0  
binaryLen = len(binary)  
  
for i in range(0, binaryLen, 1):  
 power = binaryLen - 1 - i  
 decimal += int(binary[i]) \* pow(2, power)  
  
print("对应的十进制数是：", decimal)

这一次结果是不是对了？等于15，跟我们刚才用1\*23+1\*22+1\*21+1\*20 相加算出来的和一样，对不对

4、加上输入

那回到前面的问题11111、10101011010101…又分别是十进制的多少呢？这个简单，把binary的值改成11111，10101011010101再运行两次就知道了。

这样当然也可以，但必须修改程序的代码，如果你希望用户在程序运行时输入他想转换的二进制数呢？

之前我们说过，一个程序有3大部分：输入、处理和输出。我们前面已经做了处理和输出了，还没有输入。现在该向程序加入输入了。这样一来我们就能写出与用户交互的程序，这就有趣多了。

把1111换成这一句：

binary = input("请输入一个二进制数：")

再来试一下：

binary = input("请输入一个二进制数：")  
decimal = 0  
power = 0  
binaryLen = len(binary)  
  
for i in range(0, binaryLen):  
 power = binaryLen - 1 - i  
 decimal += int(binary[i]) \* pow(2, power)  
  
print("对应的十进制数是：", decimal)

运行🡪

输入11111，得到对应的十进制数31；

再输入10101011010101，得到10965。

恩，这下就得到我们想要的结果了。

其实，程序无论多复杂，无论多大，做的事情多重要，最终都可以理解为这样的一种模型：读一些输入 —>  做一些计算  —>  最后输出，当然计算可能复杂，输出输出之间可能各种各样的关系。

这就是编程的基本套路，我们这段代码就是按照这个套路完成的。是不是很简单？

四、结束语&扩展延伸

除了十进制和二进制生活中还有各种各样的计数方式：

比如，一个星期有7天，每满7天就是1个星期，所以星期是七进制的；钟表的秒针走一圈60秒，就是1分钟，同样分针走一圈60分又是一小时，所以分秒是60进制的；而时针却是走12步才转一圈，它是十二进制的；选班干部时唱票用的“正”字计数法是五进制的；算盘是十六进制的；而和计算机八竿子打不着的八卦图，却也是二进制的。

有这么一句话：世界上有10（注：这里读作一零）种人，一种是懂二进制的，一种是不懂二进制的。你不但懂了，还会用代码实现了，给自己一个赞吧

===============================================

以下为完整的程序代码：

# 第一步：输入(input)--告诉程序你的二进制数  
binary = input("请输入一个二进制数：")  
  
# 第二步：处理(process) --将二进制数转化为十进制数  
decimal = 0  
power = 0  
binaryLen = len(binary)  
  
for i in range(0, binaryLen):  
 power = binaryLen - 1 - i  
 decimal += int(binary[i]) \* pow(2, power)  
   
# 第三步：输出(output) –将十进制数的结果打印到屏幕上  
print("对应的十进制数是：", decimal)

有几点说明

1、有可能是针对从来没有系统学过Python的孩子，所以对变量、赋值、循环…这些基本概念都做了详细讲解，整体字数较多

2、没有将处理转换部分写到独立的函数中，因为篇幅有限，不想在这个示例中再花费笔墨去讲函数的定义和调用

3、没有考虑二进制带小数点的情况，因为四年级的孩子还没有学负数，带小数点需要计算2-1，2-2…

感谢阅读！