一、数的进制

进制是一种记数方式，亦称进位计数法或位值计数法。利用这种记数法，可以使用有限种数字符号来表示所有的数值。任何一种数制都包含两个基本要素：基和权。基又叫基数，是组成该数制的数码个数，一般来说，k进制的基数就是k，包含k个数字；权又叫权值，是指每一个数位上的1对应的数值，可以表示为基数的若干次幂。十进制数的基数为10，十进制数234中2的权值是10^2，3的权值是10^1，4的权值是10^0，所以十进制数234还可表示为：$2\times10^2+3\times10^1+4\times10^0$

除了生活中常见的10进制，计算机还有二进制、十六进制等，我们通常用一个下标来表示该数的进制(十进制数可以忽略)，也可以在该数的最后以字母来表示，见下表。

进位制	二进制	八进制	十进制	十六进制
标识	B	O	D	H

二、十进制

我们先来看6785: 3位数的每一位 输入一个三位数的自然数，然后一次输出这个数的每位上的数字，并用逗号分隔。
10进制的123为什么代表123呢，$123=1*100+2*10+3*1$，即123有1个100，2个10和3个1，也就是逢10进1，10进制中每一位只会出现0～9。

1. 求个位最简单，直接%10即可；
2. 求十位呢，123有12个10，但是10个10是百位(10进制中每一位只会出现0～9，不能出现10)，所以任意一个数我们//10后再求次个位也就是%10就可以求出十位；
3. 求百位也就是有几个100，直接//100即可。

给定任意一个数，%10得到当前位，//10抛弃当前位，不断重复下去，即可以得到其每一位。
参考代码

```
n=int(input())
# 求出个位
c=n%10
# 求出十位（抛弃个位后再求个位）
b=n//10%10
# 求出百位（抛弃个位和十位后）
a=n//100
print(a,b,c,sep=',')
```

三、二进制

数据在计算机内部是以二进制方式进行存储和处理的。计算机的内部有无数个负责开关的半导体元件，0代表开关的断，1代表开关的合。生活中的二进制有逻辑上的“真”与“假”，黑白图像中的“黑”与“白”。二进制由德国数学家莱布尼茨发明，到20世纪40年代应用在电子计算机中。二进制是计算机的核心。

二进制数的特点是

1. 有两个基本数码：0，1。
2. 采用逢二进一的进位规则。

二进制用字母B标识，例如$1101.01B = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2}$ 。
$2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}$是不同位置上的权值。

二进制中的0和1可以认为是有或无，可以尝试下6831: 苹果装箱问题

四、十进制转二进制

十进制和二进制之间关系

十进制	二进制		十进制	二进制
0	0		8	1000
1	1		9	1001
2	10		10	1010
3	11		11	1011
4	100		12	1100
5	101		13	1101
6	110		14	1110
7	111		15	1111

十进制数存在和二进制数以上的转换关系，15以内的我们可以搞定了。若给定十进制数超过15求出其2进制值表示怎么做呢?
因为二进制仅存0和1，所以若可以求出位数，可以直接分别求出每一位是0还是1。十进制正整数n的二进制位数为$\lfloor \log_2 n \rfloor + 1$，$\lfloor \quad \rfloor$表示向下取整。

参考代码1

```
import math
n=int(input())
# num_of_digits 表示位数
num_of_digits = math.floor((math.log2(n))) + 1
# 用对应的权值分别求出0和1，最高位为2^(位数-1)，最低位为2^0
for i in range(num_of_digits-1,-1,-1):
    # 当前位所对应的权值为2^i
    current_digit = n // (2**i)
    print(current_digit,end="")
    # 求完之后减去当前位影响
    n -=  current_digit * (2**i)
```

6882: 十进制转二进制提供了一种更为简洁的方法，我们可以将其%2获得当前位，然后//2抛弃当前位，直到0为止。
100/2 = 50…0
50 / 2 = 25…0
25 / 2 = 12…1

12 / 2 = 6…0
3 / 2 = 1…1
1 / 2 = 0…1
将所得的余数倒过来即得到答案1100100，如下图所示。

直到0为止也就是while语句，我们可以写出如下代码：
参考代码2

```
n=int(input())
# n_base_2 代表2进制数，初始为空
n_base_2 = ''
# 直到n为0结束
while n:
    # n%2获得当前位，为int，需要转换为str字符串
    # 先求出低位，最后求得高位。所以是当前位+之前求的
    n_base_2 = str(n%2) + n_base_2
    # 抛弃当前位
    n = n // 2

print(n_base_2)
```

先求出的是低位，最后是高位，也就是先进后出的关系。满足选修一第三章“栈”的性质，所以也可以用栈去完成。
参考代码3

```
st = [-1]*100
# top代表顶端的位置，初始没有值，所以顶端不存在为-1
top = -1
n = int(input())
# 直到0结束
while n:
    # 要放在栈顶，栈元素个数+1
    top = top+1
    # 当前位进栈
```

```
    st[top] = n % 2
    # 抛弃当前位
    n = n//2
# 按照栈从后往前输出
while top >= 0:
    print(st[top], end="")
    top = top-1
```

Python的format函数非常强大，存在更简单的写法。

参考代码4

```
n=int(input())
print(format(n,'b'))
```

五、二进制转十进制

6883: 二进制转十进制 给定二进制数，让我们转10进制。
第1位索引 i 为0，其对应为$2^{位数-1}$，也就是2^{n-1-i}，每一位依次累加即可。
参考代码1

```
s = input()
# n存储结果
n = 0
# 求出长度
l = len(s)
# 对每一位分别进行处理
for i in range(l):
    # s[i]为当前位，类型为str，注意转换
    n += int(s[i])*2**(l-1-i)
print(n)
```

当然还存在其他做法，我们从前到后依次处理，每次都将上次得到的结果乘2+当前位，这样正好每一位乘上的位权是正确的。
参考代码2

```
s = input()
# n存储十进制数
n = 0
# 对每一位分别进行处理
# 由于和索引无关，可以直接遍历字符串
for i in s:
    # 把上次结果*2+当前位
    n = n * 2 + int(i)
print(n)
```

Python的int函数自带进制的处理，直接利用代码很短

```
n = int(input(),2)
print(n)
```

六、十六进制

二进制数在实际使用中，由于位数太长，不便于书写和记忆，所以人们常采用十六进制数来表示。
十六进制数的特点是

1. 由十六个基本数码组成，即0，1，2，...，9，A，B，C，D，E，F
2. 采用逢十六进一的进位规则。

十六进制用字母H标识，$B574H = 11\times16^3 + 5\times16^2 + 7\times16^1 + 4\times16^0$。与二进制相类似，$16^3$，$16^2$，$16^1$，$16^0$是不同位置上的权值。

七、十六进制和其他进制转换

十进制和十六进制之间关系

十进制	十六进制		十进制	十六进制
0	0		8	8
1	1		9	9
2	2		10	A
3	3		11	B
4	4		12	C
5	5		13	D
6	6		14	E
7	7		15	F

7031: 十进制转十六进制。求10进制数的每一位我们用了%10得到当前位，//10抛弃当前当前位，不断进行直到0为止。十进制转二进制也是这样，十六进制当然也同理，换为16就可以了，不过10～15还是需要我们特殊处理下。
参考代码1

```
n=int(input())
# n_base_2 代表2进制数，初始为空
n_base_16 = ''
# 直到n为0结束
while n:
    # n%16获得当前位，为int，需要转换为str字符串
    # 先求出低位，最后求得高位。所以是当前位+之前求的
    if n%16>=10:
```

```
        # 求出其距离'A'的值，使用chr函数转换过去
        n_base_16 = chr(ord('A')+n%16-10) + n_base_16
    else:
        # 不变
        n_base_16 = str(n%16) + n_base_16
    # 抛弃当前位
    n = n // 16

print(n_base_16)
```

上面的代码却错了，为什么呢，因为数据包含了0，但是我们程序不能对0进行处理，所以需要将其特判掉。

```
if n==0:
    n_base_16 = '0'
```

当然我们也可以提前建好索引表。
参考代码

```
n=int(input())
base_16_table = "0123456789ABCDEF"
# n_base_2 代表2进制数，初始为空
n_base_16 = ''
if n==0:
    n_base_16 = '0'
# 直到n为0结束
while n:
    # n%16获得当前位，为int，直接利用索引表找到对应字母
    # 先求出低位，最后求得高位。所以是当前位+之前求的
    n_base_16 = base_16_table[n%16] + n_base_16
    # 抛弃当前位
    n = n // 16

print(n_base_16)
```

7284: 十六进制转十进制 与二进制转十进制同理。
可以尝试下以下题目
7207: 二进制转十六进制 我们可以将二进制数先转换为十进制，再将其转换为十六进制。也可以从后往前，四位一组将其转换为十六进制的一位，余下的单独算。
7285: 十六进制转二进制我们可以把十六进制的每一位单独转换，但是第一位要去除前导0，其他不能去，模拟起来要判断一下，可以尝试一下。也可以把十进制当作中间进制进行转换。

*八、任意进制转换

可以先尝试下以下题目
7030: 十进制转八进制 、7019: 二进制转八进制简易版、1386: 十转换转R进制
1386: 十转换转R进制 涉及读到文件末尾结束和一行多个数字，可以参照以下读入方式。
1386读入参考代码

```
# 捕捉异常，读取不到就结束
try:
    while True:
        # split对字符串分割，然后转为int对应给n和r
        n, r = map(int, input().split())
        ### 你的代码
except:
    pass
```

6198: Alice与进制转换进阶版

给定R1进制的n让我们转换为r2进制的，我们需要以十进制作为中间进制，先转为十进制，最后再转回r2进制。

参考代码

```
base_16_table = "0123456789ABCDEF"
def convert_to_base_10(n_base_r1,r1):
    num = 0
    if n_base_r1[0] == '-':
        # 是负数，把负号去除，结果乘上-1即可
        sign = -1
        n_base_r1 = n_base_r1[1:]
    else:
        sign = 1
    for i in n_base_r1:
        # find函数可以找到出现i的下标
        num = num * r1 + base_16_table.find(i)
    return sign * num
def convert_to_base_r2(n,r2):
    if n<0:
        # 记下符号，将其当正数处理
        sign = -1
        n = -n
    else:
        sign = 1
    n_base_r2 = ''
    # 0特别处理
    if n == 0:
        n_base_r2 = '0'
    while n:
        n_base_r2 = base_16_table[n%r2] + n_base_r2
        n = n // r2
    if sign == -1:
        return  '-'+n_base_r2
    return n_base_r2
# 捕捉异常，读取不到就结束
try:
    while True:
        # split对字符串分割，然后转为int对应给n和r
        n_base_r1, r1, r2 = map(str, input().split())
        # 把r1进制的n转换为十进制的
        n = convert_to_base_10(n_base_r1,int(r1))
```

```
        # 把十进制的n转换为r2进制的
        n_base_r2 = convert_to_base_r2(n,int(r2))
        print(n_base_r2)
except:
    pass
```