native birthday attack

Project: implement the naïve birthday attack of reduced SM3

生日攻击其实是一个概率论的问题，也就是说一个看起来很难发生的事情，事实上它发生的概率却很大。

只有一个人时，生日不同的概率是 365/365,两个人生日不同的概率是 365/365 * 364/365，n（n<365）个人生日不同的概率是 365/365 * 364/365*……*(366-n) /365。

当有 23 个人时，每个人生日不同的概率大概就是 0.492703，23 个人中有两个人生日相同的概率可以大于 50%。

代码说明：

在 sm3 加密算法中，对消息进行哈希，得到 256 位的固定输出。输出范围为 2^{256}，根据概率论的公式，我们想要达到 50%的几率，那么需要尝试的次数是：$\sqrt{\frac{\pi}{2} * 2^{256}}$。

由于电脑计算能力及数据大小对计算时间的影响，找到长度为 28bit (7*4)的碰撞。

首先声明随机函数 Random 取随机数据。

```
def Random():#取长度为128*4字节的随机数字符
    alphabet = '0123456789'
    data=[]
    for i in range(128):
        data.append(random.choice(alphabet))
    return (''.join(data))
```

定义 birth 函数，先取一个值及其加密后的值存入列表中。继续取随机数据进行加密，并与之前存入列表的加密值比较，若出现前 28bit 相同的加密值，则找到碰撞，输出碰撞对；反之则存入列表并继续取随机值加密。

```
def birth():#生日攻击
    lst=[]
    temp=Random()
    temph=hash(temp)
    temph=temph[0:7]
    lst.append(temph)
    lst.append(temp)
    while 1:
        temp=Random()
        temph=hash(temp)
        temph=temph[0:7]
        i=0
        for i in range(0,len(lst),2):
            if (lst[i]==temph)&(temp!=lst[i+1]):
                print("找到碰撞，碰撞为\n%s\n%s"%(temp,lst[i+1]))
                return 0
        lst.append(temph)
        lst.append(temp)
```

运行指导：

可改变 temph 切片长度找到其他长度加密值相等的碰撞。

运行结果：

```
====
找到碰撞，碰撞为
468694581075634750805286205243926675507129393532804067569934103353397431706112618642836125411345009477262858235667983527034526801697170208459749282266886007993591955328941184202913950150316994814779831710436274209632941963060793008557338281823250047625170 5
```