1. 输入一个自然数(<90000), 分别用递归法和非递归法求其二机制表示.

思路：递归方法边界条件是当商为1时，返回1，即余数是1，递归是result=function(n//2)

非递归方法是使用循环，当商大于0是不断除以2取整

源代码：

递归方法

n=int(input("please input a number(<90000):")) #input输出的是字符

def function(n):

result = '1'

if n==1:

return result

else:

result=function(n//2) #取整

return result+str(n%2) #字符串相连

print(function(n))

非递归方法

def function(n):

s=''

while n>0:

s=str(n%2)+s#字符串相加

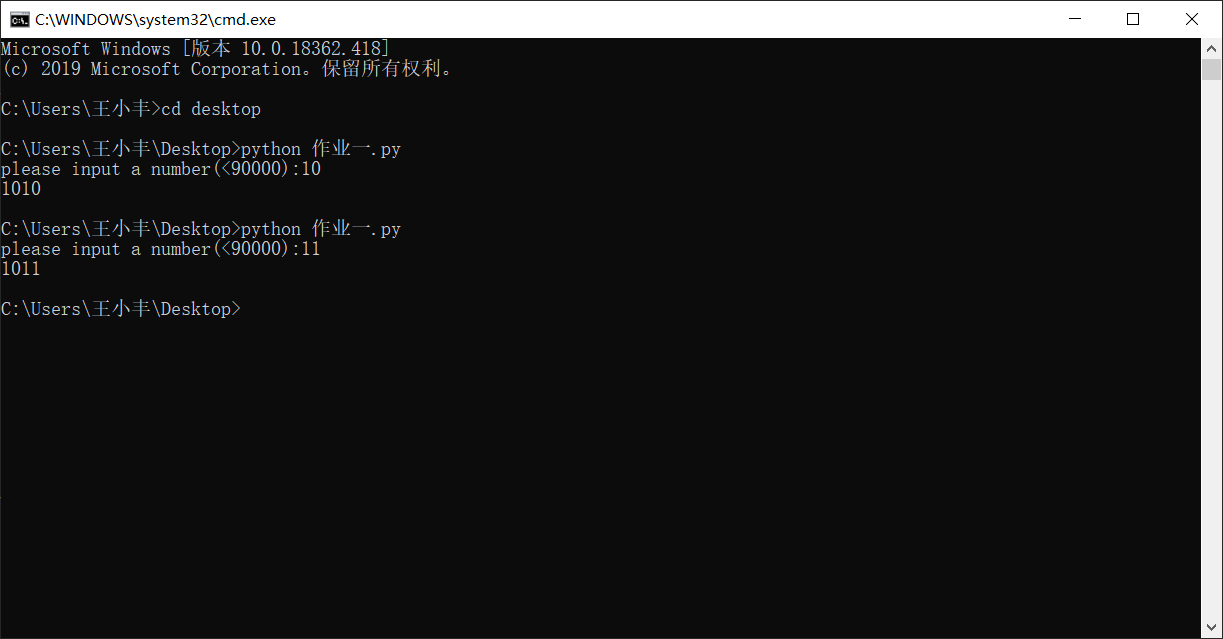
n=n//2#取整

return s

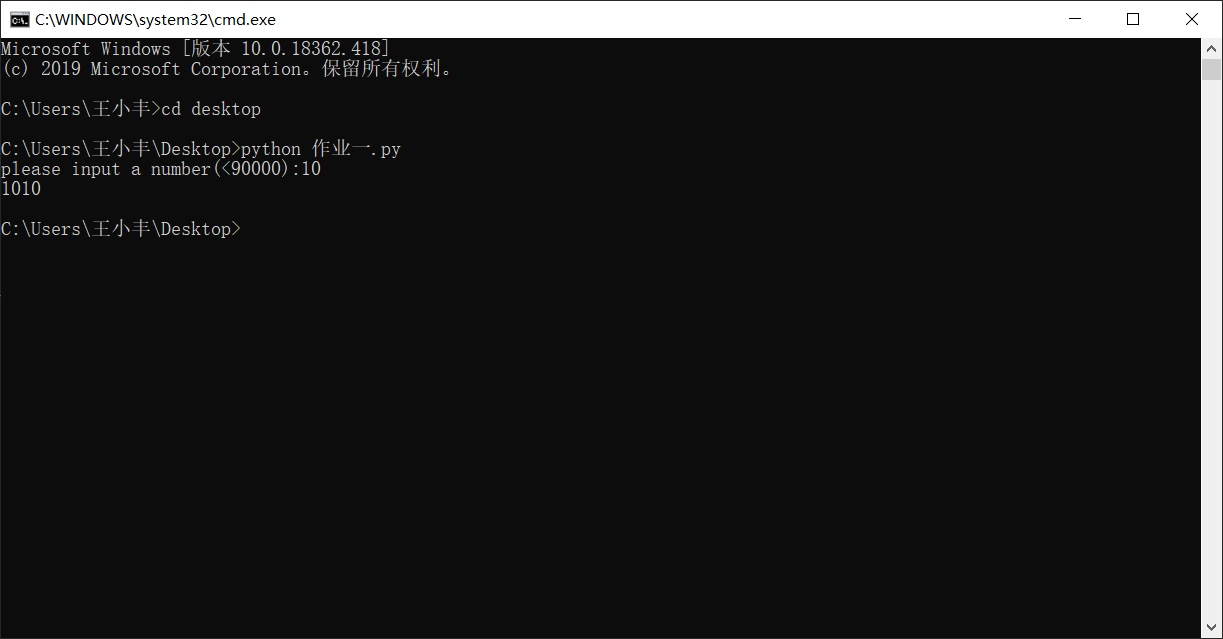
n=int(input("please input a number(<90000):"))

print(function(n))

递归截图



非递归截图



1. 分别用递归法和非递归法求Fibonacci数列的前1000位，并比较计算时间的差异.

思路：递归方法边界条件是第一个和第二个数都是1，递归是fibonacci(n-1)=fibonacci(n-1)+fibonacci(n-2)，非递归方法是先输出第一个和第二个数，然后让第三个数等于第一个和第二个数之和，第四个数等于第二个和第三个数之和，使用for循环48次

源代码：

#递归方法

def fibonacci(n):

if n<=1:

return 1

else:

return fibonacci(n-1)+fibonacci(n-2)

for i,x in enumerate(list(range(50))):

print(i,fibonacci(x))

非递归方法

fib0,fib1 = 1,1

for x in range(2,50):

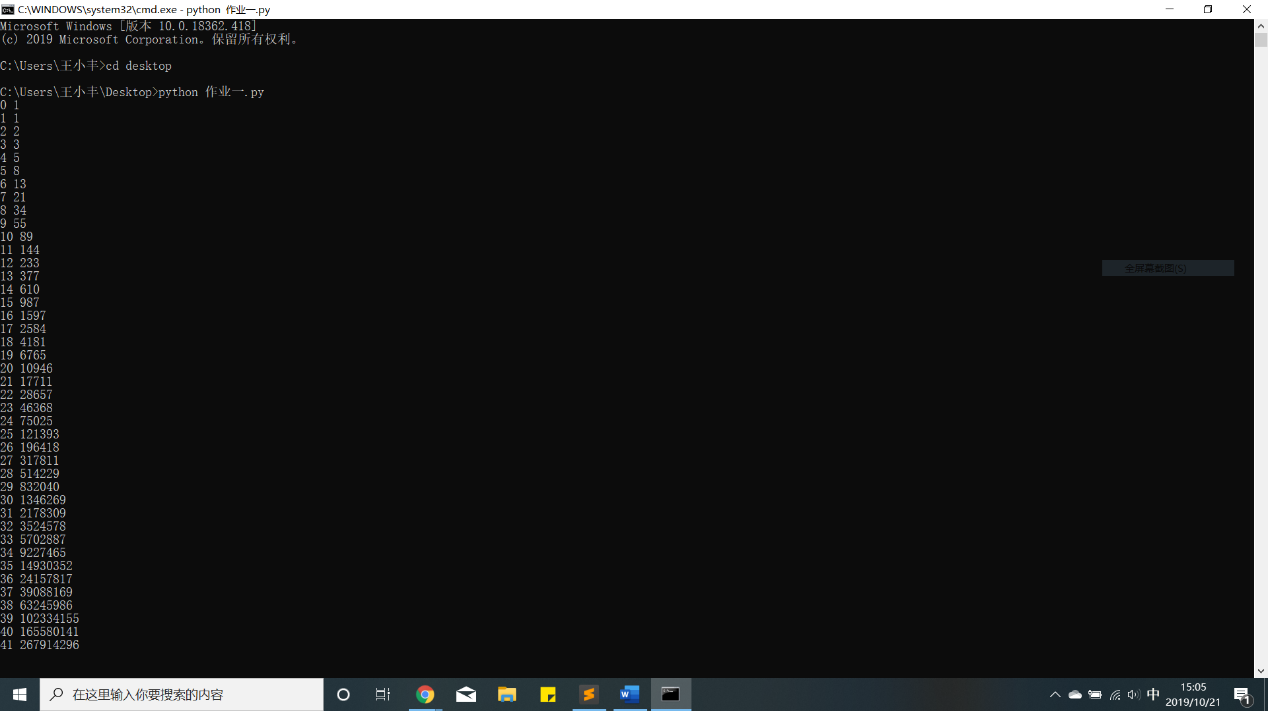
print(fib0, fib1)

fib0 += fib1

fib1 += fib0

非递归截图



递归截图

1. 用递归算法完成如下问题：有52张牌，使它们全部正面朝上，第一轮是从第2张开始，凡是2的倍数位置上的牌翻成正面朝下；第二轮从第3张牌开始，凡是3的倍数位置上的牌，正面朝上的翻成正面朝下,正面朝下的翻成正面朝上；第三轮从第4张牌开始，凡是4的倍数位置上的牌按上面相同规则翻转，以此类推，直到第一张要翻的牌超过52为止。统计最后有几张牌正面朝上，以及它们的位置号.

思路：先设一个含有52个元素的列表，列表每一个值初始化为0，也就是说列表第一个元素c[0]为第一张牌，c[1]为第二张牌，以此类推，因为是从第二张牌开始翻牌，所以c[x]应该从1开始，并判断(x+1)%(n+1)==0:如果成立，接下来判断c[x]是否为零，为零则置为1，为1则置为0.

源代码：

c=[0]\*52

def card(n):

if n>52:

return

else:

for x in range(n,52):

if (x+1)%(n+1)==0:

if c[x]==0:

c[x]=1

else:

c[x]=0

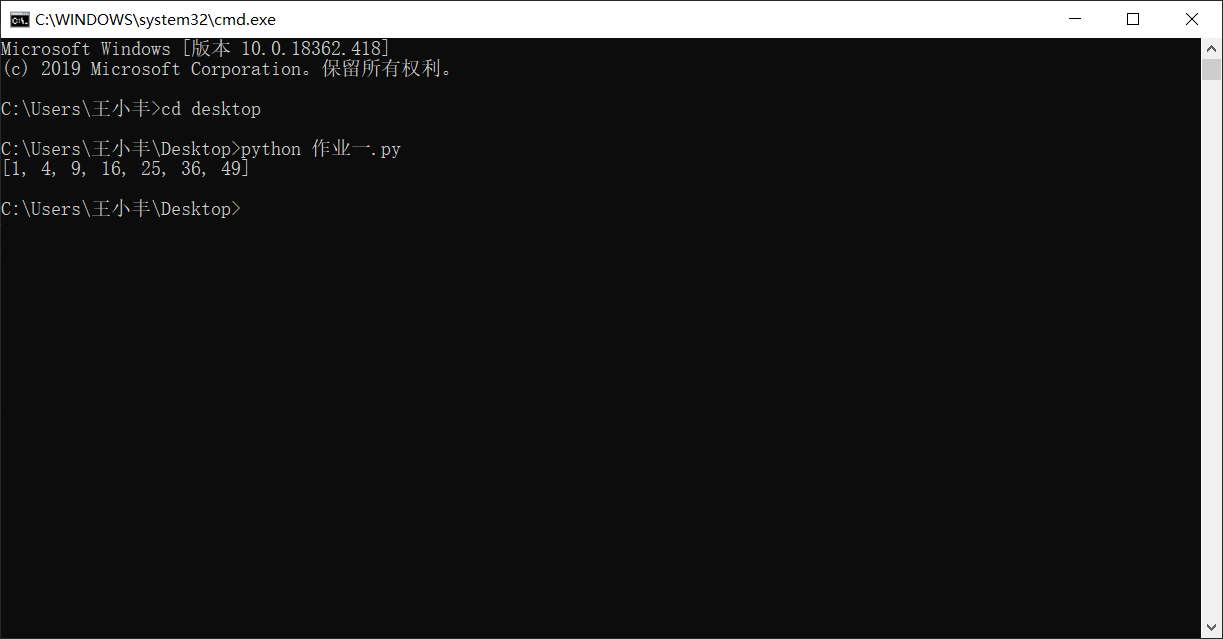
card(n+1)

card(1)

card=[x+1 for x in range(52) if c[x]==0]

print(card)

运行结果截图



1. 一个射击运动员打靶，靶一共有10环，连开6枪打中45环的可能性有多少种? （每一枪最少是0环，最多是10环）

思路：输出的条件是打靶次数n=0且环数sum=45, 首先应先判断剩余打靶次数即使都是10换，总环数能否达到45环，即(45-sum)>(10\*(n)):然后设x为每次打靶的环数在0到10之间， 每打一次，sum=sum+x,n=n-1,

源代码：

times=0

def function(n,sum):

global times

if n<0:

return

elif n==0 and sum==45: #不可用&&

times+=1

return

elif (45-sum)>(10\*(n)):#剩余的次数即使都是10环也不够45

return

for x in range(11):

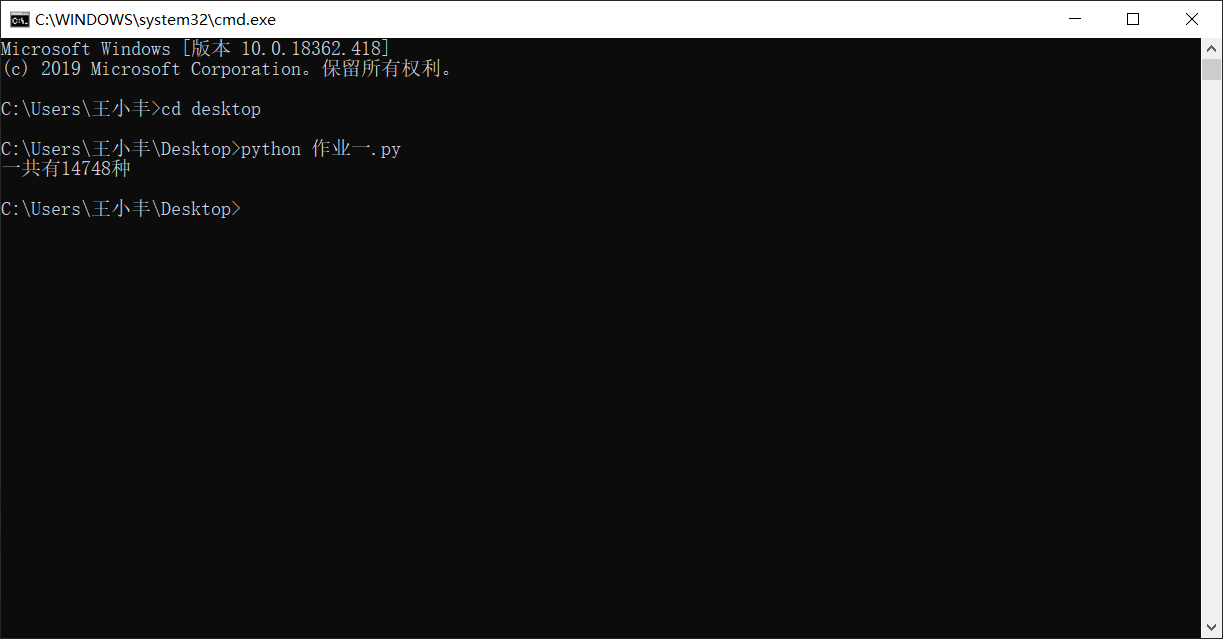
function(n-1,sum+x)

return times

function(6,0)

print('一共有%d种'%times)

运行结果截图



1. 在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，输出所有摆法。

思路：state[i] 表示列 nextx 表示下一个皇后的列 nexty 表示行或者是已经有几个皇后

源代码：

#冲突条件判断

def conflict(state,nextX):

nextY=len(state)

for i in range(nextY):

if abs(state[i]-nextX) in (0,nextY-i):#state[i]-nextX=0在同一列 and abs(state[i]-nextX)=nextY-i在斜线上

return True

return False

def queens(num=8,state=()):

for position in range(0,num):#range(0.num)

if not conflict(state,position):

if len(state)==num-1:#只剩一个皇后

yield(position,)#只含一个元素的元组

else:#前七个皇后

for result in queens(num,state+(position,)):

yield(position,)+result#元组相加

# for i in list(queens(8)):

# print(i)

#元组转化为列表输出

for i,x in enumerate(list(queens(8))):

print(i,x)

运行结果截图

