1.what will be the output of the fllowing code?

List=[‘a’,’b’,’c’,’d’,’e’]

print(list[10:]

#[]

2.such integers in a file.在一个文件总和整数

Input :

Filename

Output:

a number

#def sumint(filename):

sum=0

with open (filename,’r’) as fp:

for line in fp:

num\_str=line\_readline()

num=int(num\_str)

sum+=num

return sum

3.implement binary search.实现二进制搜索

def binary\_search(find, list1) :

low = 0

high = len(list1)

while low <= high :

mid = (low + high) / 2

if list1[mid] == find :

return mid

#左半边

elif list1[mid] > find :

high = mid -1

#右半边

else :

low = mid + 1

#未找到返回-1

return -1

4.查找字符串的第一个不重复的字符

#

class Solution:

# 返回对应char

def \_\_init\_\_(self):

self.string\_all = {}

self.ch = []

def FirstAppearingOnce(self):

# write code here

if self.string\_all is None:

return '#'

for i in self.ch:

if self.string\_all[i] == 1:

return i

return '#'

def Insert(self, char):

# write code here

self.ch.append(char)

if char in self.string\_all:

self.string\_all[char] = self.string\_all[char] + 1

else:

self.string\_all[char] = 1

5.reverse a string.反转字符串

string = 'abcdef'

def string\_reverse1(string):

return string[::-1]

def string\_reverse2(string):

t = list(string)

l = len(t)

for i,j in zip(range(l-1, 0, -1), range(l//2)):

t[i], t[j] = t[j], t[i]

return "".join(t)

def string\_reverse3(string):

if len(string) <= 1:

return string

return string\_reverse3(string[1:]) + string[0]

from collections import deque

def string\_reverse4(string):

d = deque()

d.extendleft(string)

return ''.join(d)

def string\_reverse5(string):

#return ''.join(string[len(string) - i] for i in range(1, len(string)+1))

return ''.join(string[i] for i in range(len(string)-1, -1, -1))

print(string\_reverse1(string))

print(string\_reverse2(string))

print(string\_reverse3(string))

print(string\_reverse4(string))

print(string\_reverse5(string))

6.print all combinations of a string.打印字符串的所有组合

#

1

import itertools

print list(itertools.permutations(['a','d','e','g']))

2

string = ['a','b','c','d','\n']

def Permutation(string,i):

if string == None:

return

if string[i] == '\n':

print("%s\n"%string)

else:

for j in range(i,len(string)-1):

string[j],string[i] = string[i],string[j]

Permutation(string,i+1)

string[j],string[i] = string[i],string[j]

Permutation(string,0)

7.implement a hash table.实现一个哈希表

#

class HashMap(object):

def \_\_init\_\_(self):

# 初始化总表为，容量为2的表格（含两个子表）

self.maps = BetterMap(2)

self.num = 0 # 表中数据个数

def get(self,k):

return self.maps.get(k)

def add(self, k, v):

# 若当前元素数量达到临界值（子表总数）时，进行重排操作

# 对总表进行扩张，增加子表的个数为当前元素个数的两倍！

if self.num == len(self.maps.maps):

self.resize()

# 往重排过后的 self.map 添加新的元素

self.maps.add(k, v)

self.num += 1

def resize(self):

""" 重排操作，添加新表, 注意重排需要线性的时间 """

# 先建立一个新的表,子表数 = 2 \* 元素个数

new\_maps = BetterMap(self.num \* 2)

for m in self.maps.maps: # 检索每个旧的子表

for k,v in m.items: # 将子表的元素复制到新子表

new\_maps.add(k, v)

self.maps = new\_maps # 令当前的表为新表

8.what will be the output of the following code in each step?

Class c:

Dangerous=2

C1=c()

C2=c()

Print(c1.dangerous)

#2