# 无限输入 python3

for line in sys.stdin:

s = list(line.strip())

while True:

try:

temp = list(map(int,input().split()))

except Exception as e:

break

while True:

try:

line = input()

if not line:

break

except:

break

# heapq 优先队列

import heapq

h = []

heapq.heappush(h,item)

heapq.heappop(h)

>>>h

[1, 6, 3, 9, 7]

// h 变为小顶堆

# Python sort函数

使用 Timsort方法进行排序

TimSort算法是一种起源于归并排序和插入排序的混合排序算法，设计初衷是为了在真实世界中的各种数据中可以有较好的性能。基本工作过程是：

1.扫描数组，确定其中的单调上升段和严格单调下降段，将严格下降段反转；

2.定义最小基本片段长度，短于此的单调片段通过插入排序集中为长于此的段；

3.反复归并一些相邻片段，过程中避免归并长度相差很大的片段，直至整个排序完成，所用分段选择策略可以保证O(n log n)时间复杂性。

可以看到，原则上TimSort是归并排序，但小片段的合并中用了插入排序。

# 单例

\_\_new\_\_是真正创建实例对象的方法，所以重写基类的\_\_new\_\_方法，以此来保证创建对象的时候只生成一个实例

class Singleton(object):

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not hasattr(cls, '\_instance'):

cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls, \*args, \*\*kwargs)

return cls.\_instance

class Foo(Singleton):

pass

foo1 = Foo()

foo2 = Foo()

print foo1 is foo2

# 多线程单例

import threading

class Singleton(object):

\_instance\_lock = threading.Lock()

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

pass

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not hasattr(Singleton, "\_instance"):

with Singleton.\_instance\_lock:

if not hasattr(Singleton, "\_instance"):

Singleton.\_instance = object.\_\_new\_\_(cls)

return Singleton.\_instance

class SingleTonTest(object):

def create(self):

for i in range(10):

t = threading.Thread(target=SingleTonTest.PrintAddress,args=[i,])

t.start()

def PrintAddress(self):

print(Singleton())

if "\_\_main\_\_" == \_\_name\_\_:

sig0 = Singleton()

sig1 = Singleton()

print(sig0)

print(sig1)

test = SingleTonTest();

test.create()

# bisect包

一旦决定使用二分搜索时，立马要想到使用这个模块

import bisect

L = [1,3,3,6,8,12,15]

x = 3

x\_insert\_point = bisect.bisect\_left(L,x)#在L中查找x，x存在时返回x左侧的位置，x不存在返回应该插入的位置..这是3存在于列表中，返回左侧位置１

print x\_insert\_point

x\_insert\_point = bisect.bisect\_right(L,x) #在L中查找x，x存在时返回x右侧的位置，x不存在返回应该插入的位置..这是3存在于列表中，返回右侧位置３

print x\_insert\_point

x\_insort\_left = bisect.insort\_left(L,x) #将x插入到列表L中，x存在时插入在左侧

print L

x\_insort\_rigth = bisect.insort\_right(L,x) #将x插入到列表L中，x存在时插入在右侧

print L

结果：

1

3

[1, 3, 3, 3, 6, 8, 12, 15]

[1, 3, 3, 3, 3, 6, 8, 12, 15]

# Filter

filter() 函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。

该接收两个参数，第一个为函数，第二个为序列，序列的每个元素作为参数传递给函数进行判，然后返回 True 或 False，最后将返回 True 的元素放到新列表中。

注意: Pyhton2.7 返回列表，Python3.x 返回迭代器对象，

def is\_odd(n):

return n % 2 == 1

newlist = filter(is\_odd, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

print(newlist)

[1, 3, 5, 7, 9]

# Python多线程