数据结构分为：线性结构、树结构、图结构

线性结构：数据结构中的元素存在一对一的相互关系

树结构：数据结构中的元素存在一对多的相互关系

图结构：数据结构中的元素存在多对多的相互关系

数组与列表不同：

1. 数组元素类型要相同
2. 数组长度固定

**列表是存地址**

插入删除,需要**全部移动**

**栈是一个数据集合，只能在一端进行插入或删除操作列表**

进栈：push

出栈：pop

取栈顶：gettop

class stack:  
 def \_\_init\_\_(self):  
 self.stack = []  
 def push(self, element):  
 self.stack.append(element)  
 def pop(self):  
 return self.stack.pop()  
 def getpop(self):  
 if len(self.stack) > 0:  
 return self.stack[-1]  
 else:  
 return None

**队列：仅允许在列表的一端进行插入，另一端进行删除**

进行插入的一端为队尾（rear）

进行删除的一端为对头（front）

队列的性质：先进先出

环形队列：当队尾指针front==Maxsize + 1 时，再前进一个位置就自动到0

队首指针前进1：front=(front + 1) % Maxsize

队尾指针前进1：rear=(rear + 1) % Maxsize

队空条件：rear = front

队满条件：(rear + 1) % Maxsize == front

class Queue:  
 def \_\_init\_\_(self , size = 100):  
 self.queue = [0 for \_ in range(size)]  
 self.size = size  
 self.rear = 0 #队尾  
 self.front = 0 #队首  
  
 def push(self , element):  
 if not self.isFilled():  
 self.rear = (self.rear + 1) % self.size  
 self.queue[self.rear] = element  
 else:  
 raise IndexError("Queue is filled.")  
  
 def pop(self):  
 if not self.isEmpty():  
 self.front = (self.front + 1) % self.size  
 return self.queue[self.front]  
 else:  
 raise IndexError("Queue is empty.")  
 #判断队空  
 def isEmpty(self):  
 return self.rear == self.front  
 #判断队满  
 def isFilled(self):  
 return (self.rear + 1) % self.size == self.front

双向队列：两边能进能出

rom collections import deque  
"""  
q = deque([1,2,3], 5)  
#单向  
q.append(4)  
#队尾进队  
q.popleft()  
#队首出队  
  
#双向队列  
q.appendleft(1)   
#队首进队  
q.pop()  
#队尾出队  
"""  
def tail(n):  
 with open('test.txt','r') as file:  
 q = deque(file , n) #进队test然后队尾出  
 return q