**链表**：由一系列节点组成元素集合。每个节点包含数据与和指向下一个节点的指针

class Node:  
 def \_\_init\_\_(self, item):  
 self.item = item  
 self.next = None

**头插法：**

def creatLinkList(dataSet):  
 head = Node(dataSet[0])  
  
 for element in dataSet[1:]:  
 node = Node(element)  
 node.next = head  
 head = node  
 return head

**尾插法：**

def creatLinkListTail(dataSet):  
 head = Node(dataSet[0])  
 tail = head  
 for elemnent in dataSet[1:]:  
 node = Node(elemnent)  
 tail.next = node  
 tail = node  
 return head

插入、删除：

def insLinkList(data, dataSet):  
 curNode = dataSet  
 data.next = curNode.next  
 curNode.next = data  
   
def delLinkList(data, dataSet):  
 curNode = dataSet  
 data = curNode.next  
 curNode.next = curNode.next.next  
 del data

哈希表：

通过一个哈希函数来计算数据存储位置的数据结构：

Insert(key,value)插入键值对

Get(key)判断是存在键key值

Delete(key)删除键为key的键

构建大小为m的寻址表T

Key为k的元素放到h(k)位置上

H（k）是一个函数，其将域U映射到表T

哈希冲突：存储的值的总数量是无限的，因此对于任何哈希函数都会出现两个不同元素映射到同一个位置的情况，这种情况叫做哈希冲突

解决哈希冲突：

开放寻址法：冲突向后探查新的位置来存储这个值

拉链法：当冲突时，冲突的元素将被加到该位置链表的最后

除法哈希法：h(k)=k%m

乘法哈希法：h(k)=floor(m\*(A\*key%1))

全域哈希法：h(k)=((a\*key+b)mod p)mod m