**顺序查找**：从列表第一个元素开始，到最后一个

def linearSearch(dataSet ,value ):  
 for i in range(range(dataSet)):  
 if dataSet[i] == value:  
 return i  
 return

**二分查找**：通过对待查找的值与候选区中间值比较，可以使候选区减少一半

def binarySearch(dataSet ,value):  
 left = 0  
 right = len(dataSet ) - 1  
 while left <= right: #候选区有值  
 mid = (left + right) // 2  
 if dataSet[mid] == value:  
 return mid  
 elif dataSet[mid] > value: #带查找的值mid左侧  
 right = mid - 1  
 else: #带查找的值mid右侧  
 left = mid + 1  
 else:  
 return None

列表排序：将无序列表变为有序列表

**冒泡排序**：列表每两个相邻的数，比较交换

def bubbleSort(dataSet):  
 for i in range(len(dataSet) - 1):  
 exchange = False  
 for j in range(len(dataSet) - i - 1):  
 if dataSet[j] > dataSet[j+1]:  
 dataSet[j] , dataSet[j+1] = dataSet[j+1], dataSet[j]  
 exchange = True  
 print(dataSet)  
 if not exchange:  
 return

**选择排序：**每趟都记录最小值

def selectSort(dataSet):  
 for i in range(len(dataSet) - 1):  
 min\_loc = i  
 for j in range(i, len(dataSet)):  
 if dataSet[j] < dataSet[min\_loc]:  
 min\_loc = j  
 dataSet[i],dataSet[min\_loc] = dataSet[min\_loc],dataSet[i]  
 print(dataSet)

**插入排序：**

摸牌，然后再手上的牌遍历排序

def insertSort(dataSet):  
 for i in range(1, len(dataSet)): #i 表示摸到的牌  
 temp = dataSet[i]  
 j = i - 1 #j 表示手里的牌  
 while j >= 0 and dataSet[j] > temp:  
 dataSet[j+1] = dataSet[j]  
 j -= 1  
 dataSet[j+1] = temp  
 print(dataSet)

**快速排序：**

分成临界值的左边，右边递归排序

def partition(dataSet ,left , right):  
 temp = dataSet[left]  
 while left < right:  
 while left <right and dataSet[right] >= temp: #从右边找比temp小的数  
 right -= 1 #往左走一步  
 dataSet[left] = dataSet[right] #右边的值写到左边的空位  
 while left <right and dataSet[left] <= temp:  
 left += 1  
 dataSet[right] = dataSet[left]  
 dataSet[left] = temp #temp归位  
 return left  
  
def quickSort(dataSet , left, right):  
 if left < right: #至少两个元素  
 mid = partition(dataSet , left, right)  
 quickSort(dataSet , left, mid - 1)  
 quickSort(dataSet , mid + 1, right)