```
#堆排序的实现  完全按照c++的算法写的
a=[1,3,5,7,9,2,4,6,8,0]
len=10
#完成对一个父节点位置的放置 放置为还满足最小堆
def adjust_heap(list,p,len): #p为父节点 len为长度
    lchild=2*p+1   #左孩
    currentp=list[p]   #保存父节点 避免被替换  方便比较
    while lchild<len:
        if lchild+1<len and list[lchild]>list[lchild+1]  :
            lchild+=1
        if currentp>list[lchild]:
            list[p]=list[lchild]
            p=lchild   #更新父节点
            lchild=2*p+1  #更新子节点
        else:
            break   #这个不能省去
    list[p]=currentp  #把父节点放在正确的位置上
    return list

def heap_sort(list,len):
    #初始化堆为最小堆
    for i in range(len//2)[::-1]:  #加这个就可以逆序 从最后一个父节点开始
        list=adjust_heap(list,i,len)
    print(list)
```

```
    #交换首尾值  这样得到的序列为逆序 及最小堆 得到的结果是从大到小的排序
    for j in range(len)[::-1]:
        list[0],list[j]=list[j],list[0]
        list=adjust_heap(list,0,j)
    print(list)

heap_sort(a,len)
```

2.

```
def bubble_sort(lists):
    # 冒泡排序
    #如果i > j就交换 就可以保证i的值是最小的
    count = len(lists)
    for i in range(0, count):
        for j in range(i + 1, count):
            if lists[i] > lists[j]: #注意这里是i 与j 比较 这样就可以把最小的值放到最前面
                lists[i], lists[j] = lists[j], lists[i]
    return lists
def my_bubble_sort():
    #按照c的方法实现 一样的效果
    a = [9, 10, 5, 6, 1, 5, 9, 12, 9, 5]
    len = len(a)
    for i in range(len - 1):
        for j in range(len - i - 1):
```

```
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
    print(a)

#快速排序
def quick_sort(list,left,right):  #right=len-1
    if left>=right:
        return list
    key=list[left]
    #这里记得把left right存起来 因为后面会改变其值 而递归调用会用到这个没有改变的值
    low=left
    high=right

    while left<right:
        while left < right and list[right] >= key:   #这里必须加等号 否则会死循环
            right -= 1
        list[left] = list[right]

        while left < right and list[left] <= key:
            left += 1
        list[right] = list[left]
    list[right]=key

    quick_sort(list,low,left-1)
    quick_sort(list,left+1,high)
```

```
return list
```