练习

有一个无序序列[37, 99, 73, 48, 47, 40, 40, 25, 99, 51]，对其先排序输出新列表。
分别尝试插入20、40、41到这个新序列中合适的位置，保证其有序。

思路
排序后二分查找到适当位置插入数值。
排序使用sorted解决，假设升序输出。
查找插入点，使用二分查找完成。
假设全长为n，首先在大致的中点元素开始和待插入数比较，如果大则和右边的区域的中点继续比较，如果小则和左边的区域的中点进行比较，以此类推。
直到中点就是

```
def insert_sort(orderlist, i):
    ret = orderlist[:]
    low = 0
    high = len(ret) - 1
    while low < high:
        mid = (low + high) // 2
        if ret[mid] < i:
            low = mid + 1 # 说明i大，往右找，调整下限
        else:
            high = mid # 说明i小于等于，往左找，调整上限
    print(low, i) # low为插入点
    ret.insert(low, i)
    return ret

# 测试
lst = [37, 99, 73, 48, 47, 40, 40, 25, 99, 51]

newlst = sorted(lst) # 升序
print(newlst) #[25, 37, 40, 40, 47, 48, 51, 73, 99, 99]
for x in (40, 20, 41):
    newlst = insert_sort(newlst, x)
    print(newlst)
```

看似上面代码不错，请测试插入100。
问题来了，100插入的位置不对，为什么?

```
def insert_sort(orderlist, i):
    ret = orderlist[:]
    low = 0
    high = len(ret) # 去掉减1
    while low < high:
        mid = (low + high) // 2
        if ret[mid] < i:
            low = mid + 1 # 说明i大，往右找，调整下限
        else:
```

```
            high = mid # 说明i小于等于，往左找，调整上限
    print(low, i) # low为插入点
    ret.insert(low, i)
    return ret

# 测试
lst = [37, 99, 73, 48, 47, 40, 40, 25, 99, 51]

newlst = sorted(lst) # 升序
print(newlst) #[25, 37, 40, 40, 47, 48, 51, 73, 99, 99]
for x in (40, 20, 41, 100):
    newlst = insert_sort(newlst, x)
    print(newlst)
```

`high = len(orderlist)`，去掉减1，不影响整除2，但影响下一行判断。
`while low < high` 这一句low索引可以取到length-1了，原来只能取到length-2，所以一旦插入元素到尾部就出现问题了。

算法的核心，就是折半至重合为止。

二分

二分前提是有序，否则不可以二分。

二分查找算法的时间复杂度O(log n)

bisect模块

bisect模块提供的函数有：

- `bisect.bisect_left(a,x, lo=0, hi=len(a))`
  查找在有序列表a中插入x的index。lo和hi用于指定列表的区间，默认是使用整个列表。如果x已经存在，在其左边插入。返回值为index。
- `bisect.bisect_right(a,x, lo=0, hi=len(a))` 或 `bisect.bisect(a, x,lo=0, hi=len(a))`
  和bisect_left类似，但如果x已经存在，在其右边插入。
- `bisect.insort_left(a,x, lo=0, hi=len(a))`
  在有序列表a中插入x。等同于a.insert(bisect.bisect_left(a,x, lo, hi), x) 。
- `bisect.insort_right(a,x, lo=0, hi=len(a))` 或者 `bisect.insort(a, x,lo=0, hi=len(a))`
  和insort_left函数类似，但如果x已经存在，在其右边插入。

函数可以分2类：
bisect系，用于查找index。
Insort系，用于实际插入。
默认重复时从右边插入。

```
import bisect

lst = [37, 99, 73, 48, 47, 40, 40, 25, 99, 51, 100]

newlst = sorted(lst) # 升序
print(newlst) # [25, 37, 40, 40, 47, 48, 51, 73, 99, 99, 100]
print(list(enumerate(newlst)))
```

```
print(20, bisect.bisect(newlst, 20))
print(30, bisect.bisect(newlst, 30))
print(40, bisect.bisect(newlst, 40))

print(20, bisect.bisect_left(newlst, 20))
print(30, bisect.bisect_left(newlst, 30))
print(40, bisect.bisect_left(newlst, 40))

for x in (20, 30, 40, 100):
    bisect.insort_left(newlst, x)
    print(newlst)
```

应用

判断学生成绩，成绩等级A~E。其中，90分以上为'A'，80~89分为'B'，70~79分为'C'，60~69分 为'D'，60分以下为'E'

```
import bisect

def get_grade(score):
    breakpoints = [60, 70, 80, 90]
    grades = 'EDCBA'

    return grades[bisect.bisect(breakpoints, score)]

for x in (91, 82, 77, 65, 50, 60, 70, 80, 90):
    print('{} => {}'.format(x, get_grade(x)))
```