list.append(x) #在列表的末端添加一个新的元素

Add an item to the end of the list; equivalent to a[len(a):] = [x].

list.extend(L)#将两个 list 中的元素合并到一起

Extend the list by appending all the items in the given list; equivalent to a[len(a):] = L.

list.insert(i, x)#将元素插入到指定的位置（位置为索引为 i 的元素的前面一个）

Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).

list.remove(x)#删除 list 中第一个值为 x 的元素（即如果 list 中有两个 x , 只会删除第一个 x ）

Remove the first item from the list whose value is x. It is an error if there is no such item.

list.pop([i])#删除 list 中的第 i 个元素并且返回这个元素。如果不给参数 i ,将默认删除 list  中最后一个元素

Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.）

list.index(x)#返回 list 中 ， 值为 X 的元素的索引  
　　　Return the index in the list of the first item whose value is x. It is an error if there is no such item.list.count(x)#返回 list 中 ， 值为 x 的元素的个数

Return the number of times x appears in the list.

demo:

#-\*-coding:utf-8-\*-  
L = [1,2,3]      #创建 list   
L2 = [4,5,6]

print L  
L.append(6)      #添加  
print L  
L.extend(L2)  #合并  
print L  
L.insert(0,0) #插入  
print L  
L.remove(6)      #删除  
print L  
L.pop()          #删除  
print L  
print L.index(2)#索引  
print L.count(2)#计数  
L.reverse()　　　#倒序  
print L

result:

[1, 2, 3]  
[1, 2, 3, 6]  
[1, 2, 3, 6, 4, 5, 6]  
[0, 1, 2, 3, 6, 4, 5, 6]  
[0, 1, 2, 3, 4, 5, 6]  
[0, 1, 2, 3, 4, 5]  
2  
1  
[5, 4, 3, 2, 1, 0]

list.sort(cmp=None, key=None, reverse=False)

　　Sort the items of the list in place (the arguments can be used for sort customization, see [sorted()](https://docs.python.org/2/library/functions.html" \l "sorted" \t "http://www.linuxidc.com/Linux/2017-04/_blank)for their explanation).

1.对一个 list 进行排序。默认按照从小到大的顺序排序

L = [2,5,3,7,1]  
L.sort()  
print L

==>[1, 2, 3, 5, 7]

L = ['a','j','g','b']  
L.sort()  
print L

==>['a', 'b', 'g', 'j']

2.reverse 是一个 bool 值. 默认为 False , 如果把它设置为 True, 那么这个 list 中的元素将会被按照相反的比较结果(倒序)排列.

#  reverse is a boolean value. If set to True, then the list elements are sorted as if each comparison were reversed.

L = [2,5,3,7,1]  
L.sort(reverse = True)  
print L

==>[7, 5, 3, 2, 1]

L = ['a','j','g','b']  
L.sort(reverse = True)  
print L

==>['j', 'g', 'b', 'a']

3.key 是一个函数 ， 它指定了排序的关键字 ， 通常是一个 lambda 表达式 或者 是一个指定的函数

#key specifies a function of one argument that is used to extract a comparison key from each list element: key=str.lower. The default value is None (compare the elements directly).

#-\*-coding:utf-8-\*-  
#创建一个包含 tuple 的 list 其中tuple 中的三个元素代表名字 ， 身高 ， 年龄  
students = [('John', 170, 15), ('Tom', 160, 12), ('Dave', 180, 10)]  
print students

==>[('John', 170, 15), ('Tom', 160, 12), ('Dave', 180, 10)]

students.sort(key = lambda student:student[0])  
print students

==>[('Dave', 180, 10), ('John', 170, 15), ('Tom', 160, 12)]#按名字(首字母)排序

students.sort(key = lambda student:student[1])  
print students

==>[('Tom', 160, 12), ('John', 170, 15), ('Dave', 180, 10)]#按身高排序

students.sort(key = lambda student:student[2])  
print students

==>[('Dave', 180, 10), ('Tom', 160, 12), ('John', 170, 15)]#按年龄排序

4.cmp 是一个指定了两个参数的函数。它决定了排序的方法。

#cmp specifies a custom comparison function of two arguments (iterable elements) which should return a negative, zero or positive number depending on whether the first #argument is considered smaller than, equal to, or larger than the second argument: cmp=lambda x,y: cmp(x.lower(), y.lower()). The default value is None.

#-\*-coding:utf-8-\*-  
students = [('John', 170, 15), ('Tom', 160, 12), ('Dave', 180, 10)]  
print students

==>[('John', 170, 15), ('Tom', 160, 12), ('Dave', 180, 10)]

#指定 用第一个字母的大写(ascii码)和第二个字母的小写(ascii码)比较  
students.sort(cmp=lambda x,y: cmp(x.upper(), y.lower()),key = lambda student:student[0])  
print students

==>[('Dave', 180, 10), ('Tom', 160, 12), ('John', 170, 15)]

#指定 比较两个字母的小写的 ascii 码值  
students.sort(cmp=lambda x,y: cmp(x.lower(), y.lower()),key = lambda student:student[0])  
print students

==>[('Dave', 180, 10), ('John', 170, 15), ('Tom', 160, 12)]

#cmp(x,y) 是python内建立函数，用于比较2个对象，如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1

cmp 可以让用户自定义大小关系。平时我们认为 1 < 2 ， 认为 a < b。

现在我们可以自定义函数，通过自定义大小关系（例如 2 < a < 1 < b） 来对 list 进行指定规则的排序。

当我们在处理某些特殊问题时，这往往很有用。