下面列出列表常用的方法操作列表以及小例子：  
  
1.  Append  
     在列表末尾添加元素，需在列表末尾添加元素，需要注意几个点:  
     A. append中添加的参数是作为一个整体

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name.append(list(" tiger"))  
 >>> name  
 ['s', 'c', 'o', 't', 't', [' ', 't', 'i', 'g', 'e', 'r']]

得到的值不是：['s', 'c', 'o', 't', 't', ' ', 't', 'i', 'g', 'e', 'r']  
如果想要这种的追加方式，可以试试分片赋值(或者下面说到的extend方法)：

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name[len(name):] = list(" tiger")      #从末尾追加  
 >>> name  
 ['s', 'c', 'o', 't', 't', ' ', 't', 'i', 'g', 'e', 'r']

B.append一次性只能添加一个元素

复制代码代码如下:

>>> name = list("scott")  
>>> name  
['s', 'c', 'o', 't', 't']  
>>> name.append("A","B")        #添加多个元素即将报错  
Traceback (most recent call last):  
  File "<stdin>", line 1, in ?  
TypeError: append() takes exactly one argument (2 given)  
>>> name.append("A")  
>>> name  
['s', 'c', 'o', 't', 't', 'A']

2. Count

统计某个元素在列表中出现的次数

复制代码代码如下:

>>> name = list("scott")  
>>> name  
['s', 'c', 'o', 't', 't']  
>>> name.count('s')  
1  
>>> name.count("t")  
2  
>>> name.count("A")  
0  
>>> name.append(list("Python"))  
>>> name  
['s', 'c', 'o', 't', 't', ['P', 'y', 't', 'h', 'o', 'n']]  
>>> name.count(['P', 'y', 't', 'h', 'o', 'n'])  
1

3. Extend

   在原列表追加另一个序列的中的多个值

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name.extend(list(" tiger"))  
 >>> name  
 ['s', 'c', 'o', 't', 't', ' ', 't', 'i', 'g', 'e', 'r']

当然，我们可以用分片赋值来实现：

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name[len(name):] = list(" tiger")  
 >>> name  
 ['s', 'c', 'o', 't', 't', ' ', 't', 'i', 'g', 'e', 'r']

这时小伙伴们会想到，我们可以直接用操作符"+"嘛，还方便些：

复制代码代码如下:

 >>> name = list("scott")  
 >>> pwd  = list(" tiger")  
 >>> name + pwd  
 ['s', 'c', 'o', 't', 't', ' ', 't', 'i', 'g', 'e', 'r']  
 >>> name  
 ['s', 'c', 'o', 't', 't']

从这三种方式操作的输出，可以看出：  
extend和分片赋值都是修改原列表，相对而言，extend可读性强些，而操作符"+"是生成一个新的列表，不影响原列表，如果  
我们需要生成新列表而不影响原列表，就可以用操作符"+"。  
  
4.Index  
从列表中找出某个值第一个（注意是第一个）匹配项的索引位置

复制代码代码如下:

>>> name = list("scott")  
>>> name  
['s', 'c', 'o', 't', 't']  
>>> name.index('t')    ##第一个字母t的索引位置是3  
3     
>>> name.index('a')  
Traceback (most recent call last):  
  File "<stdin>", line 1, in ?  
ValueError: list.index(x): x not in list  
>>> 'a' in name  
False  
>>> 'a' not in name  
True

从输出可以看出，index找的是第一个匹配项的索引位置，而如果查找的元素不在列表中，会报错（返回-1会不会好一点呢?），当然如果想避免报  
错，我们可以先用in操作，判断某个元素是否在某个列表中，如果在的话，然后进行index操作。  
  
5. Insert  
    用于将对象插入到列表中，俩个参数，第一个是索引位置，第二个插入的元素对象。

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name.insert(2,'tiger')     ##在索引为2的地方插入字符串tiger    
 >>> name  
 ['s', 'c', 'tiger', 'o', 't', 't']

我们也可以用分片赋值：

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name[2:2] = ['tiger']  
 >>> name  
 ['s', 'c', 'tiger', 'o', 't', 't']  
 >>> name[2:2] = 'tiger'  
 >>> name  
 ['s', 'c', 't', 'i', 'g', 'e', 'r', 'tiger', 'o', 't', 't']

这里需要注意的是，如果是插入一个元素，需要用[]括起来，不然，直接用字符串的话，是插入字符串的列表，在索引位置之后添加。  
当然，用insert的可读性比分片赋值强。  
  
6. Pop  
    移除列表中的一个元素（最后一个元素），并返回该元素的值

复制代码代码如下:

>>> name = list("scott")  
>>> name  
['s', 'c', 'o', 't', 't']  
>>> name.pop()  
't'  
>>> name  
['s', 'c', 'o', 't']  
>>> name.append("t")  
>>> name  
['s', 'c', 'o', 't', 't']

分片赋值模拟pop:

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name[len(name)-1:] = []  
 >>> name  
 ['s', 'c', 'o', 't']

这上面用pop和append模拟了栈的先进先出LIFO。  
  
7. Remove  
   移除列表中某个值的第一匹配项： 如果有俩个相等的元素，就只是移除匹配的一个元素，如果某元素不存在某列表中，便会报错，而且一次性只能  
   移除一个元素。

复制代码代码如下:

>>> name = list("scott")  
>>> name  
['s', 'c', 'o', 't', 't']  
>>> name.remove("t")    #去掉第一个t  
>>> name  
['s', 'c', 'o', 't']  
>>> name.remove("A")    #不存在会报错  
Traceback (most recent call last):  
  File "<stdin>", line 1, in ?  
ValueError: list.remove(x): x not in list  
>>> "A" not in name  
True  
>>> name.remove("s","c")  #一次只能移除一个元素  
Traceback (most recent call last):  
  File "<stdin>", line 1, in ?  
TypeError: remove() takes exactly one argument (2 given)

8.Revense

   将列表中的元素反向

复制代码代码如下:

 >>> name = list("scott")  
 >>> name  
 ['s', 'c', 'o', 't', 't']  
 >>> name.reverse()  
 >>> name  
 ['t', 't', 'o', 'c', 's']

9. Sort & Sorted

    sort方法用于对列表进行排序，修改原列表，不会返回一个已排序的列表副本

复制代码代码如下:

 >>> result = [8,5,5,3,9]  
 >>> result.sort()  
 >>> result  
 [3, 5, 5, 8, 9]

如果我们想要返回一个已排序的列表副本，而不影响原来的列表呢，一种方法，我们可以先赋值原来列表（可以用分片赋值复制），然后  
在复制的列表上做sort操作，另一种方法，就是使用sorted函数，它会返回已排序的列表副本：

复制代码代码如下:

 >>> result = [8,5,5,3,9]  
 >>> result2 = sorted(result)  
 >>> result  
 [8, 5, 5, 3, 9]  
 >>> result2  
 [3, 5, 5, 8, 9]

原文出处：

<http://www.jb51.net/article/46768.htm>