关于Python中列表的综述

面向对象的程序设计，一定程度上可以使程序的维护和扩展变得容易一些，同时可大大提高程序开发效率。Python作为一种面向对象的语言，其内部构建了一些内置对象，包括整数、浮点数、字符串、列表、元组、字典与集合。为了更好地熟悉这门语言，进一步理解“对象”的含义，本文针对内置对象中的“列表”进行简要阐述。

（一）列表的基本知识及特点

在Python中，列表也被称为序列，此外字符串与元组也属于序列，其有着相同的访问模式与操作符。通常，使用中括号“[]”表示列表，并用逗号分隔其中元素。例如通过下面代码创建了一个简单列表。

```
fruit = ['apple','pear','melon','lemon']     #创建一个名为fruit的列表
print(fruit)                                  #输出列表fruit中的信息
```

下面将从性质、操作符、操作方法三方面简述列表相关内容。

1.性质

（1）长度：通过len(列表)获取列表长度，即内部元素个数。

```
ls=[1,2,3]
print(len(ls))
3
```

（2）索引：通过变量名[位置编号]的方式进行索引，一般分为正向与反向索引，正向索引的位置编号从0开始，反向索引则是从列表最后一个元素开始，即位置编号为-1。

```
ls=[1,2,3,5]
print(ls[1])    #正向索引
2
print(ls[-1])    #反向索引
5
```

（3）切片：通过变量名[开始位置：结束位置：间隔]的方式完成切片，其也分为正向与反向。

```
ls=[1,2,3,5,8]
print(ls[0:3:2])    #正向切片
[1,3]
print(ls[-1:-4:-1])    #反向切片
[8, 5, 3]
```

2.操作符

（1）拼接：可直接通过“+”连接。

```
a =[1,2]
b =[3,4]
print(a + b)
[1,2,3,4]
```

（2）成倍复制：通过`n*list`或`list*n`实现。

```
a =[1,2]
print(a*3)
[1,2,1,2,1,2]
```

3.操作方法

（1）增加元素主要有以下三种形式：

a.直接添加元素：列表.append（待增元素）

```
fruit = ['apple','pear']
fruit.append ('lemon')
print(fruit)
['apple','pear','lemon']
```

b.任意位置插入元素：列表.insert（位置编号，待增元素），此处加入的新元素在位置编号前。

```
fruit = ['apple','pear']
fruit.insert (1,'lemon')
print(fruit)
['apple','lemon','pear']
```

c.在末尾整体并入另一列表：列表1.extend（列表2），在此将以下面的例子区别append与extend的用法。

```
fruit = ['apple','pear']
fruit.append (['melon','lemon'])
print(fruit)
['apple','pear',['melon','lemon']]

fruit = ['apple','pear']
fruit.extend (['melon','lemon'])
print(fruit)
['apple','pear','melon','lemon']
```

（2）删除元素主要有以下两种形式：

a.列表.pop(位置)，当括号中的“位置”不填写时，默认删掉最后一个元素。

```
fruit = ['apple','pear','melon']
fruit.pop()
```

```
print(fruit)
['apple', 'pear']
```
b.列表.remove(待删元素)，默认删掉列表中第一次出现的该待删元素。
```
fruit = ['apple','pear','melon','lemon','pear']
fruit.remove('pear')
print(fruit)
['apple','melon','lemon','pear']
```
（3）查找元素：列表.index(待查元素)，默认查找列表中第一次出现的该元素位置。
```
fruit = ['apple','pear','melon','lemon','pear']
idx = fruit.index('pear')
print(fruit)
1
```
（4）修改元素：列表[位置]=新值。
```
fruit = ['apple','pear','melon','lemon']
fruit[1] = 'mango'
print(fruit)
['apple','mango','melon','lemon']
```
（5）列表复制主要有以下两种形式：
a.列表.copy()
```
fruit = ['apple','pear']
fruit_2 = fruit.copy()
print(fruit_2)
['apple','pear']
```
b.列表[:]
```
fruit_3 = fruit[:]
print(fruit_3)
['apple','pear']
```
（6）列表的排序：
a.列表.sort(),该方法直接在列表上进行排序，当括号内为空时，默认升序排序，若希望降序排序，则表述为列表.sort(reverse = True)。
```
ls = [2,5,2,8]
ls.sort()
print(ls)
[2,2,5,8]
```
b.列表*2* = *sorted(列表)，降序排序为列表*2=sorted(列表，reverse = True)。与a不同的是，该方法排序后结果生成新列表，原列表不改变。
```
ls = [2,5,2,8]
ls_2 = sorted(ls)
print(ls_2)
[2,2,5,8]
print(ls)
[2,5,2,8]
```
（7）列表翻转：
a.列表.reverse(),该方法直接在列表上进行操作。
```
ls = [2,5,2,8]
ls.reverse()
print(ls)
[8,2,5,2]
```
b.列表[::-1]，可利用切片得到翻转的列表。
```
ls = [2,5,2,8]
print(ls[::-1])
[8,2,5,2]
```
除了上面列出的内容，列表还有许多便利的操作，因此其应用场景较为广泛。总之，对于列表，首先其可存放多种类型元素，元素可重复出现，并且为可变的有序集合，即可进行增删改查。列表通常通过偏移来索引，从而读取数据，此外还支持嵌套用法。

（二）列表与数组间的异同

列表（list）是由一系列按特定顺序排列的元素组成，且列表中的元素可以是多种类型，其元素间可没有关联，在Python中主要用于顺序存储结构。数组(array)是一个同一类型的数据的有限集合。Python中本没有数组类型，其是numpy库中定义的。

二者均可以存储数据，可根据索引来获取其中的元素。不同的是，列表中的元素的数据类型可以不一样，而数组里元素的数据类型必须一样；列表不可以进行数学四则运算，而数组可以进行多样的数学运算；列表计算元素个数时，通过len()，而数组可用a.shape或np.shape(a)操作；数组存储的空间大小即为数据的大小，而列表存储的是数据的存放地址，因此，相对数组，列表需占据更多的存储空间。此外，使用数组时，需要在Python中导入数组类型（import numpy）。当然，列表与数组之间可以通过np.array(list)实现转换,以下面的例子简单呈现。
```
x =[[1,2,3],[4,5,6]]
y =np.array(x)
print(y)
[[1 2 3]
[4 5 6]]
```

基于本文对Python中列表这一数据结构的简要介绍，可大致明晰列表的定义、用途及其常见用法操作，此外还有比如列表推导式等更多的用法可以去了解练习，进而加深对于“对象”的理解。