文章编号：2095-2163(2019)06-0156-04 中图分类号：TP311.13 文献标志码：A

Python 语言中列表方法的探索

徐建忠[1]，于广浩[1]，吕思宁[2]，苏 奎[1]
（1 牡丹江医学院 医学影像学院,黑龙江 牡丹江 157000；2 牡丹江市田家炳实验中学，黑龙江 牡丹江 157011）

摘 要：本文通过对 python 列表方法的介绍和举例阐明了 python 语言中列表的使用方法,可让读者在理解列表的过程中对其有一个深刻的认识。python 列表提供了最常用的 11 种方法用于对列表的一般操作。列表方法的介绍列出了列表中常用的方法,列表方法的使用通过分析加实例的方式演示了列表的方法使用。

关键词：python；列表；分析

The exploration of the list method in python language

XU Jianzhong[1], YU Guanghao[1], LV Sining[2], SU kui[1]
(1 College of medical imaging, Mudanjiang Medical University, Heilongjiang Mudanjiang 157011, China;
2 Mudanjiang Tin Ka Ping Secondary School, Heilongjiang Mudanjiang 157011, China)

【**Abstract**】This article illustrates the use of list in Python programming language by introducing several examples, which makes the readers more impressed on it. The usually used methods are provided in this paper to shown the operations on list in Python, accompanied with some specific examples guiding the readers to use these methods.

【**Key words**】python; list; analysis

0 引 言

随着开源软件的蓬勃发展,近几年 python 得到了软件行业前所未有的重视[1]。无论是人工智能、机器学习还是大数据等这些热点领域的背后都有 python 的影子[2-3]。Python 以其独有的特点吸引了无数的开发者投入其中,而在 python 众多特点中列表数据结构当仁不让的成为了其最重要的基石[4-7]。以下将通过 python3.6.2 版本对 python 的列表方法进行探索。熟练掌握这些方法是进阶 python 能力的重要一环,而且列表方法的掌握对元组方法的学习也有事半功倍的效果。

1 列表方法的使用

设样本列表结构如下,并根据此样本列表进行使用方法的演示。

Sample _list = [‘c’, ‘a’, ‘f’, ‘b’, ‘d’, ‘g’, ‘e’, ‘q’]

list1 = [1, 2, 3, [‘a’, ‘b’, ‘c’, ‘d’], 4, 5, 6]

1.1 append

该方法的作用是向列表中添加一个指定的元素。以 sample _list 列表为例,当执行 sample _list .append(‘n’)之后,sample _list 列表的尾部就会添加一个字符串 n。该方法多用于对列表内容的扩充。

```
>>>sample _list = [´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>sample _list .append(´n´)
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, ´n´]
```

1.2 clear

该方法的作用是清空列表,使其内容消失成为空列表。以 sample _list 列表为例,当执行 sample _list .clear( )之后,sample _list 列表的内容就都会被删除。该方法多用于对列表内容的全部删除。

```
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>sample _list .clear( )
>>>print(sample _list)
[ ]
```

1.3 copy

该方法的作用是浅复制列表。以 sample _list 为例,当执行 other _list = sample _list .copy( )时 sample _list 会将自身列表的内容复制出来一份传递给 other _list 这个列表。需要特别注意的是:列表的

作者简介：徐建忠(1976-),男,硕士,讲师,主要研究方向:医学影像设备学与自动化处理；苏 奎(1981-),男,硕士,讲师,主要研究方向:智能网络。
通讯作者：苏 奎 Email：suensk@163.com

收稿日期：2019-09-20

copy 方法只执行浅复制,也就意味着复制操作只在列表的第一层起作用。如果嵌套列表执行 copy 方法的话被嵌套的列表虽然会被复制但之后嵌套列表在被修改的时候会影响到复制出来的列表。

```
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>other _list = sample _list .copy( )
>>>print(other _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>sample _list .append(´w´)
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, ´w´]
>>>print(other _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
```

通过以上的代码可以看到,sample _list 列表在执行了 copy 方法后会复制出一个新的列表(暂命名为 other _list)。复制之后,若在 sample _list 列表后添加了一个'w',再次查看两个列表的内容就会发现,sample _list 列表的修改并不会影响到 other _list 列表,也就意味着在这个列子中两个列表互相修改的时候并不会影响到对方。

下面通过一个例子来说明什么叫浅复制。

```
>>>list1 = [1, 2, 3, [´a´, ´b´, ´c´, ´d´], 4, 5, 6]
>>>list2 = list1.copy( )
>>>print(list1)
[1, 2, 3, [´a´, ´b´, ´c´, ´d´], 4, 5, 6]
>>>print(list2)
[1, 2, 3, [´a´, ´b´, ´c´, ´d´], 4, 5, 6]
>>>list1[1] = 100
>>>list1[3][1] = ´w´
>>>print(list1)
[1, 100, 3, [´a´, ´w´, ´c´, ´d´], 4, 5, 6]
>>>print(list2)
[1, 2, 3, [´a´, ´w´, ´c´, ´d´], 4, 5, 6]
```

通过上面的代码可以看到,当嵌套列表 list1 被复制并赋值给 list2 后,list1 和 list2 都具有了相同的内容。将 list1 中索引为 1 的元素修改为 100,索引为 3 的元素(也就是被嵌套列表[´a´, ´b´, ´c´, ´d´])中索引为 1 的元素修改为'w'后,再次查看 list1 和 list2 两个列表就会发现复制方法执行后 list1 列表第一层的修改并不会影响到 list2 列表的第一层,但 list1 列表第二层的修改就会影响到 list2 列表的第二层。也就是说,在复制的过程中 copy( )方法只是复制了列表元素的内存 id 值,而不是列表元素中存储的具体内容!

1.4　count

该方法的作用是统计指定元素在列表中出现的次数。以 sample _list 为例,当执行 sample _list . count('d')之后会返回'd'在 sample _list 列表中出现的次数,该方法并不会修改 sample _list 列表的内容。此方法多用于对列表元素的统计。

```
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, ´w´]
>>>sample _list .count(´d´)1
```

通过以上代码的执行可以看到,返回结果为 1。这就意味着对 sample _list 统计其中元素'd'的个数为 1。

1.5　extend

该方法的作用是通过传入指定元素或另一个可迭代对象来扩展现有列表中的元素。以 sample _list 为例,当执行 sample _list .extend('t')之后在末尾添加元素't'。

```
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>sample _list .extend(´t´)
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, ´t´]
```

通过以上的代码可以看到列表的 extend 方法似乎和 append 方法的功能相同。两者之间的不同可通过下面的例子说明。

```
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´]
>>>sample _list .append([1, 2, 3])
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, [1, 2, 3]]
>>>sample _list .extend([1, 2, 3])
>>>print(sample _list)
[´c´, ´a´, ´f´, ´b´, ´d´, ´g´, ´e´, ´q´, [1, 2, 3], 1,
2, 3]
```

通过以上的代码可以清楚地看出,append 方法是将[1, 2, 3]这个列表当做一个整体的对象添加到 sample _list 列表之中,而 extend 方法是将[1, 2, 3]这个列表中的元素逐一追加到 sample _list 列表之中。这就是两者之间的区别!

1.6　index

返回指定元素的索引值。以 sample _list 为例,

当执行 sample _list .index('f')之后会返回'f'元素在该列表中的索引值,如果列表中没有要查找的元素则会抛出 Value Error 异常。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .index('f')
2
>>>sample _list .index('t')
Trace back (most recent call last):
  File "<input>", line 1, in <module>
Value Error: 't' is not in list
```

通过以上的代码可以看出'f'在 sample _list 中的索引值为 2。't'元素不在列表中,无法返回索引值抛出 Value Error 异常提示。

1.7 insert

在指定的索引值前插入指定的元素。以 sample _list 为例,当执行 sample _list .insert (3, 'w')之后,sample _list 列表就会在索引值为 3 的元素前插入元素'w'。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .insert (3, 'w')
>>>print(sample _list)
['c', 'a', 'f', 'w', 'b', 'd', 'g', 'e', 'q']
```

1.8 pop

通过索引值将列表元素移除并返回,若无索引值则操作列表最后一个元素。以 sample _list 为例,当执行 sample _list .pop(2)之后,会将列表中索引值为 2 的元素('f')从列表中移除并返回。如果不指定索引值则会将列表最后一个元素('q')从列表中移除并返回。该方法改变了 sample _list 中的内容,属于删除元素的一种形式。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .pop(2)
'f'
>>>print(sample _list)
['c', 'a', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .pop()
'q'
>>>print(sample _list)
['c', 'a', 'b', 'd', 'g', 'e']
```

通过以上的代码可以看出,sample _list 列表中的内容发生了变化,将原索引值为 2 和最后的一个元素('f'和'q')从列表中删除并返回。

1.9 remove

移除列表中指定的元素。以 sample _list 为例,当执行 sample _list .remove('a')时,列表中的'a'元素将会被移除且被移除的元素不会被返回。如果被移除的元素不在列表中时,则会抛出 Value Error 异常提示。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .remove('a')
>>>print(sample _list)
['c', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .remove('x')
Trace back (most recent call last):
  File "<input>", line 1, in <module>
Value Error: list .remove(x): x not in list
```

remove 和 pop 都是列表中元素的移除方法。二者之间的区别是:remove 方法接收的参数是元素本身,而 pop 方法接收的参数是元素的索引值。pop 方法在没有传入参数时默认移除列表中最后一个元素;remove 方法在没有传入参数时会抛出 Type Error:异常提示。

1.10 reverse

将列表反转。当执行 sample _list .reverse()时,列表中的元素会从原来的顺序反转为逆序。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .reverse()
>>>print(sample _list)
['q', 'e', 'g', 'd', 'b', 'f', 'a', 'c']
```

1.11 sort

将列表排序。当执行 sample _list .sort()方法之后,列表会按一定的规则进行排序且还可在排序时进行反转。

```
>>>print(sample _list)
['c', 'a', 'f', 'b', 'd', 'g', 'e', 'q']
>>>sample _list .sort()
>>>print(sample _list)
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'q']
>>>sample _list .sort(reverse=True)
>>>print(sample _list)
['q', 'g', 'f','e', 'd', 'c', 'b', 'a']
```

通过以上的代码,可以看到列表已经正序和逆序排序。

2 结束语

在 Python 语言大行其道的今天,作为其重要基石之一的列表给用户展现了一种更为高级、灵活的数据结构。使用时可以最小化考虑语法的复杂而更专注于特定业务逻辑的实现。用户可以通过列表提供的方法完成绝大多数类数组这种数据结构的功能需求,让开发过程更为便捷、流畅。

参考文献

[1] 郭海等. 基于 Python 的模式识别综合设计性实验[J]. 实验技术与管理,2019,36(8):178-181.

[2] 彭双和. 基于特征矩阵的 Python 克隆代码漏洞检测方法[J/OL]. 武汉大学学报(理学版),2019(5):472-478.

[3] 牛作东,李捍东. 基于 Python 与 flask 工具搭建可高效开发的实用型 MVC 框架[J]. 计算机应用与软件,2019,36(7):21-25.

[4] 姚新丽,周凌,郝新宇,等. Python 在防涝模型后处理中的应用[J]. 给水排水,2019,55(7):137-141.

[5] 陈宇昊. 基于 python 的弓网仿真研究[J]. 铁道机车车辆,2019,39(3):38-42.

[6] 陈家浩,王轶骏,吕诚. 一种基于 Python 符号执行的自动化网络攻击流量获取方法[J]. 计算机应用与软件,2019,36(2):294-307.

[7] 居政,王端宜,厉淡宁. 基于 Python 语言的弯沉盆数据库构建[J]. 公路,2019,64(1):26-30.

(上接第 155 页)

[2] BACHER U, MAYER H. Automatic road extraction from multispectral high resolution satellite images. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 3/W24), 2005. 2934.

[3] BARSI A, HEIPKE C. Artificial neural networks for the detection of road junctions in aerial images. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34 (Part 3/W8), 2003. 17-19.

[4] ZIEMS M, GERKE M, HEIPKE C. Automatic road extraction from remote sensing imagery incorporating prior information and colour segmentation. In:Photogrammetric Image Analysis. 141.

[5] LEBACQUE J P. Intersection modeling, application to macroscopic network traffic flow models and traffic management [M]//Traffic and Granular Flow ' 03. Springer, Berlin, Heidelberg, 2005: 261-278.

[6]李少毅, 董敏周, 陈康, 等. 基于空间多路重叠成像的多目标跟踪方法研究[J]. 西北工业大学学报, 2014(2):227-234.

[7]陈琦,曾妮红,王留召. 基于车载 LiDAR 点云的道路建模研究[J]. 北京测绘,2018,32(9):1015-1019.

[8]李俊超,李楼. AutoCAD Civil 3D 和 3ds Max Design 在道路建模中的应用[J]. 测绘通报,2013(2):91-94.

[9]王伟,占伟伟,王超,等. 一种利用模板的三维道路动态建模方法[J]. 武汉大学学报(信息科学版),2013,38(9):1092-1096.

[10]向宸薇,王拓,于舰. 应用色彩空间聚类方法实现道路建模[J]. 中国图象图形学报,2013,18(8):976-981.

[11]吴伟,刘洋,刘威,等. 自动驾驶环境下交叉口车辆路径规划与最优控制模型[J/OL]. 自动化学报:1-15[2019-09-10].