Python3 列表list

序列是Python中最基本的数据结构。序列中的每个元素都分配一个数字 - 它的位置，或索引，第一个索引是0，第二个索引是1，依此类推。

Python有6个序列的内置类型，但最常见的是列表和元组。

序列都可以进行的操作包括索引，切片，加，乘，检查成员。

此外，Python已经内置确定序列的长度以及确定最大和最小的元素的方法。

列表是最常用的Python数据类型，它可以作为一个方括号内的逗号分隔值出现。

列表的数据项不需要具有相同的类型

创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。如下所示：

```
list1 = ['Google', 'Unigress', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"];
```

与字符串的索引一样，列表索引从0开始。列表可以进行截取、组合等。

访问列表中的值

使用下标索引来访问列表中的值，同样你也可以使用方括号的形式截取字符，如下所示：

实例(Python 3.0+)

```
#!/usr/bin/python3

list1 = ['Google', 'Unigress', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

运行实例 »

以上实例输出结果：

```
list1[0]:  Google
list2[1:5]:  [2, 3, 4, 5]
```

更新列表

你可以对列表的数据项进行修改或更新，你也可以使用append()方法来添加列表项，如下所示：

实例(Python 3.0+)

```
#!/usr/bin/python3

list = ['Google', 'Unigress', 1997, 2000]

print ("第三个元素为 : ", list[2])
list[2] = 2001
print ("更新后的第三个元素为 : ", list[2])
```

注意：我们会在接下来的章节讨论append()方法的使用

以上实例输出结果：

```
第三个元素为 :  1997
更新后的第三个元素为 :  2001
```

删除列表元素

可以使用 del 语句来删除列表的的元素，如下实例：

实例(Python 3.0+)

```
#!/usr/bin/python3

list = ['Google', 'Unigress', 1997, 2000]

print ("原始列表 : ", list)
del list[2]
print ("删除第三个元素 : ", list)
```

以上实例输出结果：

```
原始列表 :  ['Google', 'Unigress', 1997, 2000]
删除第三个元素 :  ['Google', 'Unigress', 2000]
```

注意：我们会在接下来的章节讨论 remove() 方法的使用

Python列表脚本操作符

列表对 + 和 * 的操作符与字符串相似。+ 号用于组合列表，* 号用于重复列表。

如下所示：

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print(x, end=" ")	1 2 3	迭代

Python列表截取与拼接

Python的列表截取与字符串操作类型，如下所示：

```
L=['Google', 'Unigress', 'Taobao']
```

操作：

Python 表达式	结果	描述
L[2]	'Taobao'	读取第三个元素
L[-2]	'Unigress'	从右侧开始读取倒数第二个元素: count from the right
L[1:]	['Unigress', 'Taobao']	输出从第二个元素开始后的所有元素

```
>>>L=['Google', 'Unigress', 'Taobao']
>>> L[2]
'Taobao'
>>> L[-2]
'Unigress'
>>> L[1:]
['Unigress', 'Taobao']
>>>
```

列表还支持拼接操作：

```
>>>squares = [1, 4, 9, 16, 25]
>>> squares += [36, 49, 64, 81, 100]
>>> squares
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>>
```

嵌套列表

使用嵌套列表即在列表里创建其它列表，例如：

```
>>>a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]
'b'
```

Python列表函数&方法

Python包含以下函数:

序号	函数
1	len(list)列表元素个数
2	max(list) 返回列表元素最大值
3	min(list) 返回列表元素最小值
4	list(seq) 将元组转换为列表

```
#!/usr/bin/python3

squares = [1, 4, 9, 16, 25]
print(len(squares))
print(max(squares))
print(min(squares))
print(list(squares))
```

Python包含以下方法:

序号	方法
1	list.append(obj)在列表末尾添加新的对象
2	list.count(obj)统计某个元素在列表中出现的次数
3	list.extend(seq)在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
4	list.index(obj)从列表中找出某个值第一个匹配项的索引位置
5	list.insert(index, obj)将对象插入列表
6	list.pop([index=-1])移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
7	list.remove(obj)移除列表中某个值的第一个匹配项
8	list.reverse() 反向列表中元素
9	list.sort(key=None, reverse=False) 对原列表进行排序
10	list.clear() 清空列表
11	list.copy()复制列表

```python
#!/usr/bin/python3

aList = ["TMall", 'Google', 'Unigress', 'Taobao', 'Facebook']

aList.sort()
print("List : ", aList)
aList.sort(reverse = True)

# 获取列表的第二个元素
def takeSecond(elem):
    return elem[1]

# 列表
random = [(2, 2), (3, 4), (4, 1), (1, 3)]

# 指定第二个元素排序
random.sort(key=takeSecond)

print(random)
```

- ## Python2 List sort()方法

  描述

  sort() 函数用于对原列表进行排序，如果指定参数，则使用比较函数指定的比较函数。

  语法

  sort()方法语法：

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

参数

- cmp -- 可选参数, 如果指定了该参数会使用该参数的方法进行排序。
- key -- 主要是用来进行比较的元素，只有一个参数，具体的函数的参数就是取自于可迭代对象中，指定可迭代对象中的一个元素来进行排序。
- reverse -- 排序规则，reverse = True 降序， reverse = False 升序（默认）。

返回值

该方法没有返回值，但是会对列表的对象进行排序。

解决方法

为了照顾到广大`cmp`用户的心情，Python3还是留了一条活路的，同样也在官网文档里有介绍：

> The functools.cmp_to_key() utility is available to convert a 2.x style cmp function to a key function.

就是用`functools.cmp_to_key()`来曲线救国啦，废话不多说，直接上代码：

Python

```
from functools import cmp_to_key

nums = [1, 3, 2, 4]
nums.sort(key=cmp_to_key(lambda a, b: a - b))
print(nums)  # [1, 2, 3, 4]
```

还是熟悉的味道有木有！千万别忘了`import`哦～