5 玩转Python中的List

> AI领域中的Python开发 --- by 丁宁

@(SIGAI课程录制)

- 前两节课：Python作为一门编程语言的基础部分
- 本节课：Python中最重要的动态数据模型**List**

切片：充分发挥List的有序特性

有如下列表

```
>>> L = ['tensorflow', 'torch', 'caffe', 'mxnet', 'keras']
```

假如要取出前三个字符串组成新的列表，此时便可以使用**List**的切片功能了

```
>>> newL = L[:3]
>>> newL
['tensorflow', 'torch', 'caffe']
```

切片功能的三个参数：`[起始位置:终止位置:步长]`（只有第一个`:`是必须有的）

先来讲一下`range()`函数

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1,11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1,11,2))
[1, 3, 5, 7, 9]
>>> list(range(1,11,3))
[1, 4, 7, 10]
```

接下来生成1-10的数组

```
>>> L = list(range(1,11))
>>> L
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

当`:`之前没有数字时，默认补0，也就是从头开始

```
>>> L[:5]
[1, 2, 3, 4, 5]
>>> L[:12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> L[:-1]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L[:-3]
[1, 2, 3, 4, 5, 6, 7]
>>> L[:-15]
[]
>>> L[:0]
[]
```

> **0**是第一个；**-1**是最后一个 从起始位置到终止位置，包含起始位置，**不包含终止位置** 超出范围不报错

步长也可以是负数，轻松实现**List**倒序

```
>>> L[2:5]
[3, 4, 5]
>>> L[2:-1]
[3, 4, 5, 6, 7, 8, 9]
>>> L[1:5:3]
[2, 5]
>>> L[-1:0:-1]
[10, 9, 8, 7, 6, 5, 4, 3, 2]
>>> L[-1::-1]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> L[::-1]
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> L[::]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> L[:]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

> 还记得引用与拷贝的内容吗?

多重List的浅拷贝与深拷贝

```
import copy

a = [[1, 2, 3], [4, 5, 6]]
b = a
c = copy.copy(a)
d = copy.deepcopy(a)

a.append(7)
a[1][2] = 10

print('原数组: ', a)
print('引用赋值: ', b)
print('浅拷贝: ', c)
print('深拷贝: ', d)
```

```
sigai@8a5f47e78164:~/workspace$ python test.py
原数组:  [[1, 2, 3], [4, 5, 10], 7]
引用赋值:  [[1, 2, 3], [4, 5, 10], 7]
浅拷贝:  [[1, 2, 3], [4, 5, 10]]
深拷贝:  [[1, 2, 3], [4, 5, 6]]
```

序列的加法，乘法，初始化

序列的加法

本质上讲，字符串也是序列，同类型的序列是可以相加的：

```
>>> L
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> L + L[-2::-1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> s = 'sigai'
>>> S = 'SIGAI'
>>> s + S
'sigaiSIGAI'
>>> s + L
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'list' object to str implicitly
```

序列的乘法

序列与整数相乘，也可以快速创建包含重复元素的序列

```
>>> L * 2
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> s * 5
'sigaisigaisigaisigaisigai'
```

常用的序列初始化方案：

```
>>> L = []
>>> L = [] * 10
>>> L
[]
>>> L = [0] * 10
>>> L
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> L = [None] * 10
>>> L
[None, None, None, None, None, None, None, None, None, None]
```

序列的常见内置方法

序列的包含关系： `in`

```
>>> s
'sigai'
>>> 'i' in s
True
>>> 'I' in s
False
>>> L = list(range(11))
>>> 5 in L
True
>>> 20 in L
False
```

序列的长度，最大值，最小值：`len` `max` `min`

```
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> len(L)
11
>>> s
'sigai'
>>> len(s)
5
>>> max(L)
10
>>> min(L)
0
>>> max(s)
's'
>>> min(s)
'a'
```

List与str的相互转换

```
>>> s
'sigai'
>>> l = list(s)
>>> l
['s', 'i', 'g', 'a', 'i']
>>> s_from_l = ''.join(l)
>>> s_from_l
'sigai'
```

List的元素或切片的赋值与删除

对列表内的元素可以随意赋值和删除

```
>>> L = list(range(5))
>>> L
[0, 1, 2, 3, 4]
>>> L[2], L[4] = L[4], L[2]
>>> L
[0, 1, 4, 3, 2]
>>> del L[2]
>>> L
[0, 1, 3, 2]
```

列表内的切片也可以随意的赋值和删除，还可以用作插入

```
>>> L = [1,5]
>>> L[1:1] = list(range(2,5))
>>> L
[1, 2, 3, 4, 5]
>>> L[2:4] = []
>>> L
[1, 2, 5]
>>> del L[1:]
>>> L
[1]
```

计数函数：`count`

```
>>> from functools import reduce
>>> L = reduce(lambda x,y: x+y, [[i]*i for i in range(1,6)])
>>> import random
>>> random.shuffle(L)
>>> L
[3, 2, 5, 5, 4, 5, 3, 1, 2, 5, 5, 4, 4, 4, 3]
>>> L.count(4)
4
>>> L.count(5)
5
>>> L.count(6)
0
```

列表的排序

```
>>> L
[3, 2, 5, 5, 4, 5, 3, 1, 2, 5, 5, 4, 4, 4, 3]
>>> L_sorted = sorted(L)
>>> L_sorted
[1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
>>> L
[3, 2, 5, 5, 4, 5, 3, 1, 2, 5, 5, 4, 4, 4, 3]
>>> L.sort()
>>> L
[1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
>>> x = L.sort()
>>> print(x)
None
```

> 小坑： `sorted()`函数排序后返回，原列表不变 `L.sort()`就地排序，直接修改原列表
> `L.sort()`是对象的方法，不是函数，没有返回值

[List的内置方法及其时间空间复杂度]

初见 List comprehension

```
>>> sum([x for x in range(101) if x % 2 == 0])
2550

>>> sum([1 if x % 3 == 2 else -1 if x % 3 == 1 else 0 for x in range(101)])
-1
```

> **List Comprehension与lambda以及map/reduce还有filter连用功能更强大**
>
> 为避免重复： 关于**lambda**以及**map/reduce**还有**filter**等高阶函数，我们会在函数式编程章节中详细讲解 关于可迭代对象，生成器，迭代器等内容，我们会在迭代器章节中详细讲解