知识点

1. 切片

2. 迭代

3. 生成式

5.组合数据类型的高级特性

5.1切片

- L[a:b]表示从索引a开始取，直到索引b为止，但不包括索引b。

```
>>> L = ['Michael', 'Sarah', 'Tracy', 'Bob', 'Jack']
```

- 取前3个元素，用一行代码就可以完成切片。

```
>>> L[0:3]
['Michael', 'Sarah', 'Tracy']
```

- 从索引1开始，取出2个元素：

```
>>> L[1:3]
['Sarah', 'Tracy']
```

- **支持倒数切片**。L[-1]取倒数第一个元素

5.组合数据类型的高级特性

5.1切片——简单应用

切片操作十分有用。我们先创建一个0~99的数列。

```
>>> L = list(range(100))
>>> L
[0, 1, 2, 3, …, 99]
```

可以通过切片轻松取出某一段数列。比如前10个数：

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

后10个数：

```
>>> L[-10:]
[90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

5.组合数据类型的高级特性

5.1切片——简单应用

前11~20个数：

```
>>> L[10:20]
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

前10个数，每两个取一个：

```
>>> L[:10:2]
[0, 2, 4, 6, 8]
```

所有数，每5个取一个：

```
>>> L[::5]
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
80, 85, 90, 95]
```

5.组合数据类型的高级特性

5.1切片——tuple切片、字符串切片

tuple也是一种list，唯一区别是tuple不可变。因此，tuple也可以用切片操作，只是操作的结果仍是tuple。

```
>>> (0, 1, 2, 3, 4, 5)[:3]
(0, 1, 2)
```

字符串'xxx'也可以看成是一种list，每个元素就是一个字符。因此，字符串也可以用切片操作，只是操作结果仍是字符串。

```
>>> 'ABCDEFG'[:3]
'ABC'
>>> 'ABCDEFG'[::2]
'ACEG'
```

5.组合数据类型的高级特性

5.2迭代——可迭代对象的迭代

列表这种数据类型有下标，但很多其他数据类型是没有下标的，但是，只要是可迭代对象，无论有无下标，都可以迭代。

5.组合数据类型的高级特性

5.2迭代——可迭代对象的迭代

字典的迭代

```
>>> d = {'a': 1, 'b': 2, 'c': 3}
>>> for key in d:
    print(key)


a
c
b
```

由于字典是无序的，因此迭代出的结果顺序可能与上述结果不一样。

默认情况下，dict迭代的是key。如果要迭代value，可以用for value in d.values()，如果要同时迭代key和value，可以用for k, v in d.items()。

5.组合数据类型的高级特性

5.2迭代——可迭代对象的迭代

集合的迭代

```
>>> s=set([1,2,3,4])
>>> s
{1, 2, 3, 4}
>>> for x in s:
                    print(x)

1
2
3
4
```

5.组合数据类型的高级特性

5.2迭代——可迭代对象的迭代

字符串的迭代

```
>>> for ch in 'ABC':
        print(ch)

A
B
C
```

所以，当我们使用for循环时，只要作用于一个可迭代对象，for循环就可以正常运行，而我们不用太关心该对象究竟是list还是其他数据类型。

5.组合数据类型的高级特性

5.2迭代——Iterable类型

对于一个对象，通常是通过collections模块的Iterable类型判断该对象是否是一个可迭代的对象，如：

```
>>> from collections import Iterable
>>> isinstance('abc', Iterable              # str是否可迭代
True
>>> isinstance([1,2,3], Iterable)           # list是否可迭代
True
>>> isinstance(123, Iterable)               # 整数是否可迭代
False
```

5.组合数据类型的高级特性

5.2迭代——列表实现下标循环

如果要对list实现类似Java那样的下标循环，要怎么办呢？Python内置的enumerate函数可以把一个list变成索引—元素对，这样就可以在for循环中同时迭代索引和元素本身。

```
>>> for i, value in enumerate(['A', 'B', 'C']):
        print(i, value)

0 A
1 B
2 C
```

5.组合数据类型的高级特性

5.3列表生成式——一层循环

要生成list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]，可以用list(range(1, 11))：

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

但如果要生成[1×1, 2×2, 3×3,…, 10×10]，可使用下列语句。

```
>>> L = []
>>> for x in range(1, 11):
    L.append(x * x)

>>> L
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

5.组合数据类型的高级特性

5.3列表生成式——一层循环

利用列表生成式可以用一行语句代替循环生成上面的list。

```
>>> [x * x for x in range(1, 11)]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

for循环后面还可以加上if判断，这样我们就可以筛选出仅偶数的平方。

```
>>> [x * x for x in range(1, 11) if x % 2 == 0]
[4, 16, 36, 64, 100]
```

5.组合数据类型的高级特性

5.3列表生成式——一层循环

另外，也可把一个list中所有的字符串变成小写：

```
>>> L = ['Hello', 'World', 'IBM', 'Apple']
>>> [s.lower() for s in L]
['hello', 'world', 'ibm', 'apple']
```

5.组合数据类型的高级特性

5.3列表生成式——两层循环

下面看一个两层循环的例子。

```
>>> [m + n for m in 'ABC' for n in 'XYZ']
['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']
```

上例同时执行for m in 'ABC'循环和for n in 'XYZ'循环，然后再执行*m*+*n*。

代码练习题

请创建列表aList，值为3，4，5，6，7，9，11，13，15，17，并取出其中6,7,9,11值。

答案：

```
aList=[3, 4, 5, 6, 7, 9, 11, 13, 15, 17]
aList[3:7]
```

代码练习题

使用生成式创建列表[1,8,27,64,125]。

答案：

```
[x**3 for x in range(1, 6)]
```