# 第三章 python列表、元组、字典、集合

本节所讲内容：

3.1 python 列表

3.2 python 元组的概述

3.3 python 字典

3.4 Python 集合

## 3.1 Python列表

### 3.1.1 python列表的概述

想一想：

前面学习的字符串可以用来存储一串信息，那么想一想，怎样存储咱们班所有同学的名字呢？

定义100个变量，每个变量存放一个学生的姓名可行吗？有更好的办法吗？

答：【列表】

如果存储多个数据的用列表；例如存储100个人的名字 或者存储100个人的年龄。

很多个信息描述一个人需要用字典。字典用来描述一个信息的时候。

列表的定义：

列表是一个有序的，可修改的，元素以逗号分割，以中括号包围的序列。

列表的索引和字符串的索引类似，但不完全相同，因为列表可以修改，所以我们可以通过列表的索引来修改列表。

定义列表

name = [] name等于中括号就是定义一个空列表

列表用例存储数据的

### 3.1.2 python列表的定义

#定义一个空列表

>>> name = []

>>> type(name)

<class 'list'>

#注意：列表里面的数据类型是可变的，甚至可以嵌套一个列表

>>> name = ['12',10]

>>> name

['12', 10]

>>> type(name)

<class 'list'>

3.1.3 python列表操作：

|  |  |  |
| --- | --- | --- |
| 列表的添加 | append | 追加，在列表的尾部加入指定的元素 |
| insert | 将指定的元素插入到对应的索引位上，注意负索引 |
| extend | 将指定序列的元素依次追加到列表的尾部(合并) |
| 列表的删除 | pop | 弹出，返回并删除指定索引位上的数据，默认删除索引为-1的数据 |
| remove | 从左往右删除一个指定的元素 |
| del | 删除整个列表或列表的数据，del是python内置功能，不是列表独有的 |
| 列表的查找  注:列表没有find方法 | count | 计数，返回要计数的元素在列表当中的个数 |
| index | 查找，从左往右返回查找到的第一个指定元素的索引，如果没有找到，报错 |
| in | 查找元素是否在列表中，在输出true，否则false |
| 列表的排序 | reverse | 顺序倒序 |
| sort | 按照ascii码表顺序进行排序 |

1. 列表的添加

>>> list1 = ['诺手', 'ez', 'uzi']

追加，在列表的尾部加入指定的元素

>>> list1.append('雪人')

>>> list1

['诺手', 'ez', 'uzi', '雪人']

>>> list1.append('雪人')

>>> list1

['诺手', 'ez', 'uzi', '雪人', '雪人']

将指定的元素插入到对应的索引位上，注意负索引

>>> list1.insert(0,'寒冰')

将指定序列的元素依次追加到列表的尾部(合并)

>>> list1

['寒冰', '诺手', 'ez', 'uzi', '雪人', '雪人']

>>> list2 = [1,2,3]

>>> list1.extend(list2)

>>> list1

['寒冰', '诺手', 'ez', 'uzi', '雪人', '雪人', 1, 2, 3]

>>> list1 + list2

['寒冰', '诺手', 'ez', 'uzi', '雪人', '雪人', 1, 2, 3, 1, 2, 3]

1. 列表的删除

弹出，返回并删除指定索引位上的数据，默认删除索引为-1的数据，可指定对应的下标

>>> list1

['寒冰', '诺手', 'ez', 'uzi', '雪人', '雪人', 1, 2, 3]

>>> list1.pop()

3

>>> list1.pop()

2

>>> list1

['寒冰', '诺手', 'ez', 'uzi', '雪人', '雪人', 1]

>>> list1.pop(1)

'诺手'

>>> list1

['寒冰', 'ez', 'uzi', '雪人', '雪人', 1]

从左往右删除一个指定的元素

>>> list1.remove('ez')

>>> list1

['寒冰', 'uzi', '雪人', '雪人', 1]

删除整个列表或列表的数据，del是python内置功能，不是列表独有的

>>> del list1[0]

>>> list1

['uzi', '雪人', '雪人', 1]

>>>

>>> list1

['uzi', '雪人', '雪人', 1]

1. 列表的修改和查找

>>> list1

['uzi', '雪人', '雪人', 1, 1, 2, 3]

>>> list1[-1] = '诺手'

>>> list1

['uzi', '雪人', '雪人', 1, 1, 2, '诺手']

计数，返回要计数的元素在列表当中的个数

>>> list1

['uzi', '雪人', '雪人', 1]

>>> list1.count(1)

1

>>> list1.count('雪人')

2

>>> list1.index('雪人')

1

查找，从左往右返回查找到的第一个指定元素的索引，如果没有找到，报错

列表的排序

>>> list1.index('雪人')

1

>>> list1.index(10)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: 10 is not in list

in从左到右进行查找，找到后返回true 找不到返回flase

>>> list2 = [1,2,3,4]

>>> 1 in list2

True

>>> 10 in list2

False

1. 列表的排序

顺序倒序

>>> list1

['uzi', '雪人', '雪人', 1, 1, 2, '诺手']

>>> list1.reverse()

>>> list1

['诺手', 2, 1, 1, '雪人', '雪人', 'uzi']

按照ascii码表顺序进行排序

>>> list.sort()

>>> list1

['诺手', 2, 1, 1, '雪人', '雪人', 'uzi']

>>> list1.sort()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unorderable types: int() < str()

>>> list

[1, 2, 3]

>>> list2 = [3,2,1]

>>> list2.sort()

>>> list2

[1, 2, 3]

>>>

注意：列表内不同数据类型之间，不能相互比较。否则会出错。

1. list 函数

list（），以一个序列作为参数并把它转换为列表，如果参数是list，那么该参数就会被原样返回。

>>>a = ‘abc’

>>>list(a)

[‘a’,’b’,’c’]

如何将列表中的变成字符串？

>>> a = ['a','b','c']

>>> ''.join(a)

'abc'

>>>a = [1,2,3]

>>>''.join([str(i) for i in a])

‘123’

注意：如果你定义了list这个变量，再用list（）这个函数时会报错。

1. 列表的切片操作

上节课咱们学习了，字符串的切片，既然列表有序，可得出它也有下标值，也可以进行切片处理。

>>> lsit1 = [1,2,3,4,5,6,7,8]

>>> lsit1[1]

2

>>> lsit1[-1]

8

>>> lsit1[1:4]

[2, 3, 4]

>>> lsit1[1:6:2]

[2, 4, 6]

>>> lsit1[1:6:-1]

[]

>>> lsit1[6:1:-1]

[7, 6, 5, 4, 3]

>>> lsit1[6:-1:-1]

[]

>>> lsit1[6:1:-1]

[7, 6, 5, 4, 3]

>>> lsit1[6:1:-2]

[7, 5, 3]

## 3.2 Python元组的概述

元组的定义

元组是有序的，不可修改的，以逗号分割的，以小括号包围的序列，可以通过()和tuple函数定义。

不能修改，只有在定义的一瞬间可以修改。

元组的优点

由于元组不可变，所以遍历元组比列表要快（较小的性能提升）。

由于元组不可变，所以元组中的数据被‘写保护’,这也可以联想到，一些软件比较重要的数据都是用元组存储的。

创建元祖：

>>> num = (11,22,33,'aa')

>>> num

(11, 22, 33, 'aa')

>>> num = 1,2,3,'a'

>>> num

(1, 2, 3, 'a')

>>> 1,2,3

(1, 2, 3)

>>> type(num)

<class 'tuple'>

元祖一个逗号的秘密

>>> num = (2)

>>> num

2

>>> type(num)

<class 'int'>

>>> num = (1,)

>>> num

(1,)

>>> (1+1)\*2

4

为什么会出现这种情况呢？，Python中的（）同样也表示数学运算的一个基本符号，比如（1+1）\*2，所以这就和元组的这个小括号产生了歧义，当元组中只有一个元素，而又没有逗号的时候，它会把这元素运算出来，这个小括号会被Python解释器识别为一个运算符号，所以得到的是那个元素本身的数据类型。

访问元组

>>> num

(1, 2, 3, 'a')

>>> num[1]

2

>>> num[2]

3

修改元组

说明：Python中不允许修改元组的数据，包括不能删除其中的元素。

元组是不可变的，也就是说，元组中的元素在被赋值后不能改变。但是，如果元素本身是一个可变数据类型的列表，那么其嵌套项可以被改变。

>>> num[2].append(3)

>>> num

(1, 'a', [1, 2, 3])

>>>

元祖不能修改：

>>> num[0] = 2

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

>>>

>>> num = (1,'a',[1,2])

tuple函数

tuple函数的功能与list函数基本上一样的，以一个序列作为参数并把它转换为元组，如果参数是元组，那么该参数就会被原样返回。

列表转元祖

>>> num = ['诺手','ez','雪人','寒冰']

>>> num2 = tuple(num)

>>> num2

('诺手', 'ez', '雪人', '寒冰')

字符串转元组

>>> str1 = 'hello'

>>> str1 = tuple(str1)

>>> str1

('h', 'e', 'l', 'l', 'o')

>>> num2

('诺手', 'ez', '雪人', '寒冰')

元组转元组

>>> num3 = tuple(num2)

>>> num3

('诺手', 'ez', '雪人', '寒冰')

str1 = 'hello'

>>>a = tuple(str1)

>>>a

('h', 'e', 'l', 'l', 'o')

>>>''.join(a)

'hello'

>>>a = (1,2,3)

>>>''.join([str(i) for i in a])

嵌套元祖访问的示例

>>> tup1 = (('诺手','寒冰'),('雪人','猴子'),('孙尚香','吃鸡'))

>>> tup1[0]

('诺手', '寒冰')

>>> tup1[1]

('雪人', '猴子')

>>> tup1[2]

('孙尚香', '吃鸡')

>>> tup1[0][1]

'寒冰'

>>> tup1[1][1]

'猴子'

>>>

元组的索引和字符串以及列表完全一致。

>>> tup2 = ('a','b','c','d','e','xuegod')

>>> tup2

('a', 'b', 'c', 'd', 'e', 'xuegod')

>>> tup2[0]

'a'

>>> tup2[-1]

'xuegod'

>>> tup2[0:4]

('a', 'b', 'c', 'd')

>>> tup2[0:]

('a', 'b', 'c', 'd', 'e', 'xuegod')

>>> tup2[0:4:2]

('a', 'c')

>>> tup2[::2]

('a', 'c', 'e')

>>> tup2[::-1]

('xuegod', 'e', 'd', 'c', 'b', 'a')

>>> tup2[1:4:-1]

()

>>> tup2[4:1:-1]

('e', 'd', 'c')

### 3.1.1 python元组的方法

|  |  |  |
| --- | --- | --- |
| 元组的查找 | index | 从左往右返回第一个遇到的指定元素的索引，如果没有，报错 |
| count | 返回元组当中指定元素的个数 |

>>> tup2 = ('a', 'b', 'c', 'd', 'e', 'xuegod')

>>> tup2.index('a')

0

>>> tup2.index('b')

1

>>> tup2.index('f')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: tuple.index(x): x not in tuple

>>> tup2.count('b')

1

>>> tup2.count('d')

1

### 3.2.2 python元组与列表的区别

1、元组和列表都是有序的，元组不可修改，列表可修改。

2、元组和列表的元素可以是任何类型

3、元组的元素长度可以任意。

## 3.3 Python字典

### 3.3.1 python字典的概述

字典的定义

字典是一个无序的，可以修改的，元素呈键值对的形式，以逗号分割的，以大括号包围的序列；

字典是python基础数据类型当中唯一一个映射关系的数据类型；

由于字典无序，所以我们定义字典和字典生成之后的元素的顺序是不同的；

生活中的字典特点：前几页相当于目录结构，通过偏旁部首可以查找内容。



### 3.3.2 python字典

字典和列表一样，也能够存储多个数据，字典中的值并没有特殊顺序。

列表中找某个元素时，是根据下标进行的，如果下标很多，查找数据不方便。

当索引不好用时，使用字典，字典中找某个元素时字典的每个元素由2部分组成，键:值。例如 'name':'班长' ,'name'为键，'班长'为值，字典由键值组成。

格式：

info = {'name':'for','addr':'beijing','age':18}

1. 获取字典中的元素

# 利用键名

>>> info = {'name':'for','addr':'beijing','age':18}

>>> type(info)

<class 'dict'>

>>> info['name']

'for'

>>> info['age']

18

#get方法

>>> info.get('name')

'for'

注意：get方法如果没有找到相应的值，就会输出None，后面也可以带默认的值，工作中常用。

1. 添加和修改

>>> info = {'name':'for','addr':'beijing','age':18}

>>> info['like'] = '玛利'

>>> info

{'name': 'for', 'like': '玛利', 'age': 18, 'addr': 'beijing'}

#更改

>>> info['like'] = '武藤兰'

>>> info

{'name': 'for', 'like': '武藤兰', 'age': 18, 'addr': 'beijing'}

>>>

1. 删除

>>> info

{'name': 'for', 'like': '武藤兰', 'age': 18, 'addr': 'beijing'}

>>> del info['like']

>>> info

{'name': 'for', 'age': 18, 'addr': 'beijing'}

>>>

### 3.3.3 python字典的特点

因为字典是无序的，所以字典没有索引值；

因为字典没有索引值，所以字典以键取值，(字典的键相当于列表的索引)；

因为字典以键取值，所以字典的键唯一且不可修改；

因为字典的键不可修改，所以列表和字典等可变类型的数据不可以给字典做键。

### 3.3.4 python字典的常见操作

|  |  |  |
| --- | --- | --- |
| 字典的取值 | keys | 返回一个包含字典所有key的列表 |
| values | 返回一个包含字典所有vaule的列表 |
| get | 以键取值，如果指定键不存在，默认返回None,可以指定返回内容 |
| setdefault | 设置默认，如果键存在，返回值，如果键不存在，创造键，值默认为None，值也可以自定义setdefault(key,default=None) |
| update | 以字典格式更新指定键的内容，如果键不存在，创建键和值 |
| items | 返回字典键值呈元组形式的格式 |
| 内置方法 | len | 测量字典，键值对的个数（整体） |

>>> info = {'name': 'for', 'age': 18, 'addr': 'beijing'}

返回一个包含字典所有key的列表

>>> info.keys()

dict\_keys(['name', 'age', 'addr'])

>>> info['like'] = '吉泽'

>>> info

{'name': 'for', 'like': '吉泽', 'age': 18, 'addr': 'beijing'}

>>> info.keys()

dict\_keys(['name', 'like', 'age', 'addr'])

返回一个包含字典所有vaule的列表

>>> info.values()

dict\_values(['for', '吉泽', 18, 'beijing'])

以键取值，如果指定键不存在，默认返回None,可以指定返回内容

>>> info.get('like')

'吉泽'

设置默认，如果键存在，返回值，如果键不存在，创造键，值默认为None，值也可以自定义setdefault(key,default=None)

>>> info.setdefault('play')

>>> info

{'play': None, 'name': 'for', 'like': '吉泽', 'age': 18, 'addr': 'beijing'}

>>> info['play'] = '捆绑'

>>> info

{'play': '捆绑', 'name': 'for', 'like': '吉泽', 'age': 18, 'addr': 'beijing'}

以字典格式更新指定键的内容，如果键不存在，创建键和值

>>> info.update({'cosplay':'ez'})

>>> info

{'cosplay': 'ez', 'name': 'for', 'age': 18, 'addr': 'beijing', 'like': '吉泽', 'play': '捆绑'}

返回字典键值呈元组形式的格式

>>> info.items()

dict\_items([('cosplay', 'ez'), ('name', 'for'), ('age', 18), ('addr', 'beijing'), ('like', '吉泽'),

('play', '捆绑')])

测量字典，键值对的个数（整体）

>>> len(info)

6

|  |  |  |
| --- | --- | --- |
| 字典的删除 | pop | 弹出，返回并删除指定键对应的值 |
| popitem | 随机弹出一个键值元组，这里随机的原因是因为字典无序 |
| clear | 清空字典 |
| 字典的判断 | in | 判断指定的键是否在字典当中，即将被废除，之后我们python3用in |
| has\_key | 判断键是否在字典里返回true，否则返回false（python3去除） |
| 字典的遍历 | 通过for ...in ... | 通过for ... in ...:的语法结构，我们可以遍历字符串、列表、元组、字典等数据结构。 |

弹出，返回并删除指定键对应的值

>>> info = {'cosplay': 'ez', 'name': 'for', 'age': 18, 'addr': 'beijing', 'like': '吉泽', 'play': '捆绑'}

>>> info.pop('age')

18

随机弹出一个键值元组，这里随机的原因是因为字典无序

>>> info = {'cosplay': 'ez', 'name': 'for', 'addr': 'beijing', 'like': '吉泽', 'play': '捆绑'}

>>> info.popitem()

('cosplay', 'ez')

清空字典

>>> info1 = {'name':'for','age':10}

>>> info1

{'name': 'for', 'age': 10}

>>> info1.clear()

>>> info1

{}

判断指定的键是否在字典当中。

>>> 'name' in info

True

>>> 'namea' in info

False

通过for ... in ...:的语法结构，我们可以遍历字符串、列表、元组、字典等数据结构。

>>> for i in info:

... print(i)

...

name

addr

like

play

>>> for i in info.values():

... print(i)

...

for

beijing

吉泽

捆绑

这个了解下：

|  |  |  |
| --- | --- | --- |
| 视图模式 | viewitems | 视图模式的主要特征是与字典保持同步，在我们python基础部分不做深入研究 |
| viewkeys |
| viewvalues |
| 迭代模式 | Iteritems | 在python2.4版本之后，为了提高python运行效率python提出了迭代器、生成器、装饰器的定义，而字典的迭代模式正式去用了迭代器原理，  当我们直接调用的时候，我们只能得到一个内存地址，但我们具体想要取值的时候，需要借助next方法（python3中没有） |
| Iterkeys |
| itervalues |

## 3.4 Python集合

### 3.4.1 集合定义

集合是一组无序不重复的元素集合。

集合与之前列表、元组类似，可以存储多个数据，但是这些数据是不重复的。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

创建一个集合

>>> s1 = {1,2,3,4}

>>> type(s)

<class 'set'>

元组里面没有相同的元素（去重）

>>> num = {1,2,3,4,5,6,6}

>>> num

{1, 2, 3, 4, 5, 6}

>>>

集合对象还支持交集(intersection),差集(difference)、并集和对称差集(sysmmetric\_difference)

交集(&)：两个集合的公共部分

并集(|): 两者集合合并，没有重复元素

差集(-)： 只有前项有的元素，不存在后项中的元素。

对称差集(^)：只在a或b中，但是不会同时出现在二者中

>>> a = set('1234')

>>> b = set('3456')

交集：&两个集合的公共部分

>>> a&b

{'4', '3'}

并集：| 两者集合合并，没有重复元素

>>> a|b

{'1', '6', '5', '2', '4', '3'}

差集：-只有前项有的元素，不存在后项中的元素。

>>> a-b

{'1', '2'}

对称差集(^)：只存在a或者b中，但是不会同时出现在二者中

>>> a^b

{'1', '6', '5', '2'}

>>>

set、list、tuple之间可以相互转换

>>> a = 'hello'

字符串转列表

>>> b = list(a)

>>> b

['h', 'e', 'l', 'l', 'o']

字符串转元祖

>>> c = tuple(a)

>>> c

('h', 'e', 'l', 'l', 'o')

字符串转集合

>>> d = set(a)

>>> d

{'l', 'e', 'o', 'h'}

>>> a

'hello'

>>> b

['h', 'e', 'l', 'l', 'o']

>>> c

('h', 'e', 'l', 'l', 'o')

>>> d

{'l', 'e', 'o', 'h'}

>>>

### 3.4.2 集合操作

|  |  |  |
| --- | --- | --- |
| 集合增添 | add | 为集合无规律添加元素 |
| update | 也可以添加元素，且参数可以是不同类型，并用逗号分隔开 |
| 删除 | remove | 删除集合中的指定元素，当指定的元素不存在的时候会报错 |
| discard | 是删除集合中的指定元素，且如果元素不存在的时候，不会报错 |

s.add(x)

>>> thisset = set(("Google", "Runoob", "Taobao"))

>>> thisset.add("Facebook")

>>> print(thisset)

{'Google', 'Taobao', 'Runoob', 'Facebook'}

还有一个方法，也可以添加元素，且参数可以是列表，元组，字典等，语法格式如下：

s.update(x)

注：x可以是多个值，用逗号分隔开

>>> thisset = set(("Google", "Runoob", "Taobao"))

>>> thisset.update({1,3})

>>> print(thisset)

{3, 'Google', 'Taobao', 'Runoob', 1}

>>>

>>> thisset.update([1,4],[5,6])

>>> print(thisset)

{'Taobao', 1, 3, 4, 5, 6, 'Google', 'Runoob'}

移除元素

s.remove(x) 删除集合中的指定元素，当指定的元素不存在的时候会报错

>>> thisset = set(("Google", "Runoob", "Taobao"))

>>> thisset.remove("Taobao")

>>> print(thisset)

{'Google', 'Runoob'}

>>>

>>> thisset.remove("Facebook")

Traceback (most recent call last):

File "<pyshell#92>", line 1, in <module>

thisset.remove("Facebook")

KeyError: 'Facebook'

>>>

s.discard(x) 也是删除集合中的指定元素，且如果元素不存在的时候，不会发生错误

>>> thisset = set(("Google", "Runoob", "Taobao"))

>>> thisset.discard("Facebook")

>>> print(thisset)

{'Google', 'Taobao', 'Runoob'}

>>> thisset.discard("Google")

>>> print(thisset)

{'Taobao', 'Runoob'}

s.pop() 随机删除集合中的一个元素

>>> thisset = set(("Google", "Runoob", "Taobao", "Facebook"))

>>> thisset.pop()

'Google'

>>> print(thisset)

{'Taobao', 'Runoob', 'Facebook'}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据类型比较 | 字符串 | 列表 | 元祖 | 字典 | 集合 |
| 是否有序 | 是 | 是 | 是 | 否 | 否 |
| 是否可修改 | 否 | 是 | 不 | 是 | 是 |
| 方法多少 | 很多 | 一般 | 很少 | 较多 | 一般 |

可变类型与不可变类型

可变类型，值可以改变：

* 列表 list
* 字典 dict
* 集合 set

不可变类型，值不可以改变：

* 数值类型 int, long（python3去除）, bool, float
* 字符串 str
* 元组 tuple

通用方法整合

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 常见操作 | Python 表达式 | 结果 | 描述 | 支持的数据类型 |
| + | [1, 2] + [3, 4] | [1, 2, 3, 4] | 合并 | 字符串、列表、元组 |
| \* | 'Hi!' \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | 复制 | 字符串、列表、元组 |
| in | 3 in (1, 2, 3) | True | 元素是否存在 | 字符串、列表、元组、字典、集合 |
| not in | 4 not in (1, 2, 3) | True | 元素是否不存在 | 字符串、列表、元组、字典、集合 |

注意，in在对字典操作时，判断的是字典的键

公共方法

python内置函数

Python包含了以下内置函数

|  |  |  |
| --- | --- | --- |
| 序号 | 方法 | 描述 |
| 1 | len(item) | 计算容器中元素个数 |
| 2 | max(item) | 返回容器中元素最大值 |
| 3 | min(item) | 返回容器中元素最小值 |
| 4 | del(item) | 删除变量 |

总结：

3.1 Python 列表

3.2 Python 元组的概述

3.3 Python 字典

3.4 Python 集合

练习作业：

1、有如下变量（tu是个元祖），请实现要求的功能：

tu = ("alex", [11, 22, {"k1": 'v1', "k2": ["age", "name"], "k3": (11,22,33)}, 44])

a. 讲述元祖的特性:

1）元祖tuple是一个只读列表，只可以查看，不可以修改增加和删除；  
2）在定义只有一个元素的元祖时，要加上逗号以示和小括号的区别，如 age = (18, )

b. 请问tu变量中的第一个元素 “alex” 是否可被修改？

c. 请问tu变量中的"k2"对应的值是什么类型？是否可以被修改？如果可以，请在其中添加一个元素 “Seven”

d. 请问tu变量中的"k3"对应的值是什么类型？是否可以被修改？如果可以，请在其中添加一个元素 “Seven”