# 实验3.Python数据结构

实验目的：

掌握Python中的列表、元组、字典、集合等常见数据结构的使用方法。

实验平台：

Windows系列+Python3.X

主要实验内容：

#### 一、列表

在介绍列表之前，我们先了解一下Python中“序列”的概念。在Python中，序列是最基本的数据结构。序列中的每个元素都被分配一个数字，以表明它的位置，并称之为索引。其中，第一个索引值为0，第二个索引值为1，依此类推。

Python中的序列都可以进行索引、切片、加、乘、检查成员等操作。为了使用方便，Python还内建了确定序列长度以及确定最大和最小元素的方法。

Python中最常用的序列是列表和元组，本节介绍列表，元组放在下一节介绍。

1．列表的概念

列表（list）是Python中使用最频繁的数据类型，它是放在方括号([])内、用逗号分隔的一系列元素。

列表中元素的类型可以不同，它支持数字、字符串甚至可以包含列表。换言之，列表允许嵌套。

2．列表的创建

创建一个列表，只要把逗号分隔的不同的数据项用方括号括起来即可。例如：

list1 = ['Google', 'Runoob', 1997, 2017]

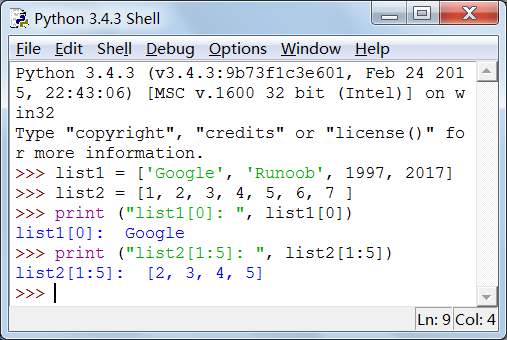
list2 = [1, 2, 3, 4, 5 ]

list3 = ["a", "b", "c", "d"]

与字符串一样，列表同样可以被索引、截取和组合。列表被截取后返回一个包含所需元素的新列表。与字符串的索引一样，列表索引也是从0开始的。

3．列表的访问与截取

可以使用下标索引来访问列表中的元素，同样，也可以使用类似于字符串切片运算的形式截取列表中的元素，例如：



令

L=['Google', 'Runoob', 'Taobao']

则截取操作如表3-1所示。

表3-1 Python列表的截取操作

|  |  |  |
| --- | --- | --- |
| Python 表达式 | 结果 | 描述 |
| L[2] | 'Taobao' | 读取第三个元素。 |
| L[-2] | 'Runoob' | 从右侧开始读取倒数第二个元素。 |
| L[1:] | ['Runoob', 'Taobao'] | 输出从第二个元素开始的所有元素。 |

下面将上述的列表截取操作放入一个程序中。

#!/usr/bin/python3

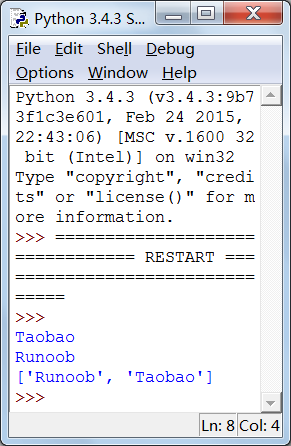
L=['Google', 'Runoob', 'Taobao']

print(L[2])

print(L[-2])

print(L[1:])

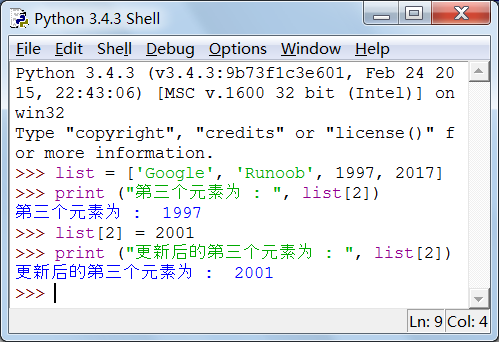
以上程序的输出结果如下：



注意：字符串、列表和元组（下一节介绍）三者都是序列，都支持切片运算，操作方法也极为相似，只是操作结果的类型有所不同。此外，不能通过切片运算对字符串和元组进行更新，列表则可以。

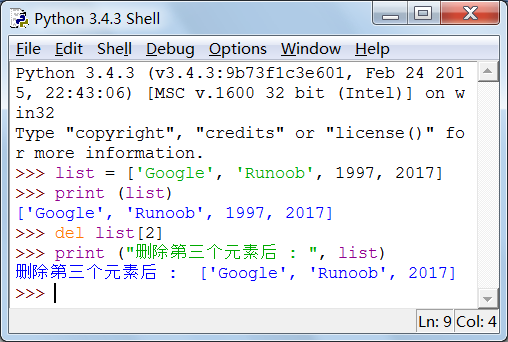
4．列表的更新

可以对列表的数据项进行修改或更新，也可以使用append()方法（稍后介绍）添加一些列表项，例如：



5．列表元素的删除

可以使用 del 语句来删除列表中的元素，例如：



使用remove()方法也可以删除列表的元素，我们稍后再讨论。

6．列表操作符

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。+ 和 \* 的用法如表3-2所示。

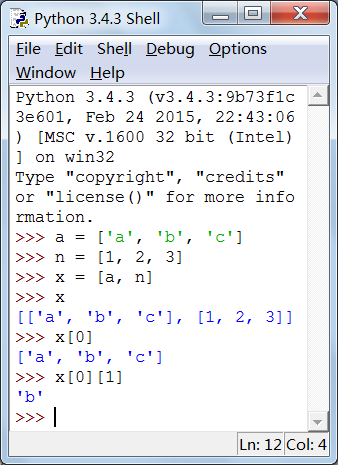
表3-2列表中+和\*的用法

|  |  |  |
| --- | --- | --- |
| Python 表达式 | 结果 | 描述 |
| len([1, 2, 3]) | 3 | 长度（即列表中元素的个数）。 |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | 组合（即拼接）。 |
| ['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 | 遍历并输出列表的各个元素。 |

注意：为了让列表的各个元素打印在一行上，而且相邻两个元素之间隔一个空格，表3-2的最后一行使用了“print (x,end=' ')”这种打印格式，后面遍历元组和集合时采用了同样的方法。

7．列表嵌套

列表嵌套指的是在列表里创建其它列表，例如：



8．Python列表中的内建函数与方法

Python列表中的内建函数如表3-3所示。

表3-3 Python列表中的内建函数

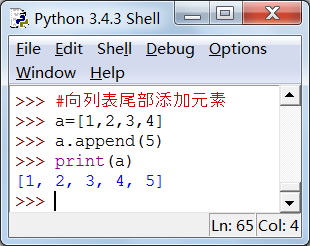
|  |  |
| --- | --- |
| 序号 | 函数 |
| 1 | len(list) 返回列表元素的个数。 |
| 2 | max(list) 返回列表元素的最大值。 |
| 3 | min(list) 返回列表元素的最小值。 |
| 4 | list(seq) 将元组转换为列表。 |

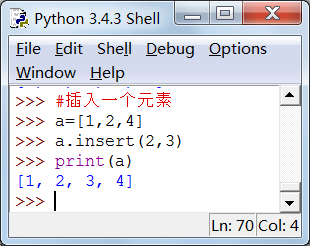
Python列表中的方法如表3-4所示。

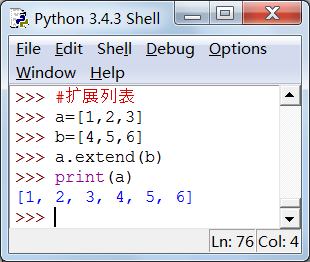
表3-4 Python列表中的方法

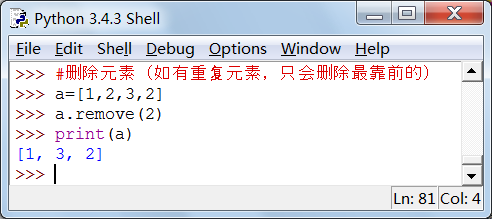
|  |  |
| --- | --- |
| 序号 | 方法 |
| 1 | list.append(obj) 在列表末尾添加新的对象。 |
| 2 | list.count(obj) 统计某个元素在列表中出现的次数。 |
| 3 | list.extend(seq) 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）。 |
| 4 | list.index(obj) 从列表中找出某个值第一个匹配项的索引位置。 |
| 5 | list.insert(index, obj) 将对象插入列表。 |
| 6 | list.pop(obj=list[-1]) 移除列表中的一个元素（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() 将列表中的元素反向。 |
| 9 | list.sort([func]) 对原列表进行排序。 |
| 10 | list.clear() 清空列表。 |
| 11 | list.copy() 复制列表。 |

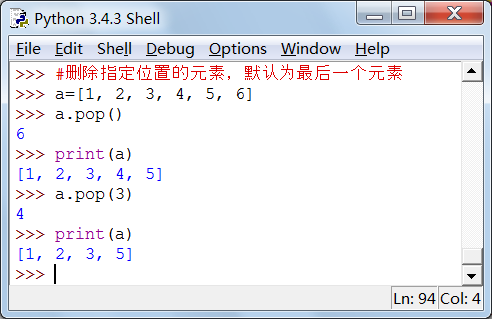
下面看几个列表操作的例子。

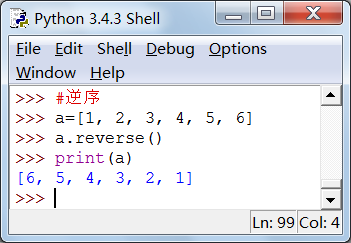


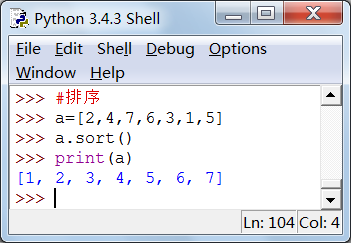












#### 二、元组

Python 的元组（tuple）与列表相似，不同之处在于元组的元素是不能修改的。另外，列表使用方括号（[]），而元组使用圆括号（()）。

1．元组的创建

元组的创建很简单，只需要在括号中添加元素，并使用逗号分隔各元素即可。例如：

tup1 = ('Google', 'Runoob', 1997, 2017)

tup2 = (1, 2, 3, 4, 5 )

tup3 = ("a", "b", "c", "d")

创建空元组使用语句

tup1 = ()

注意：

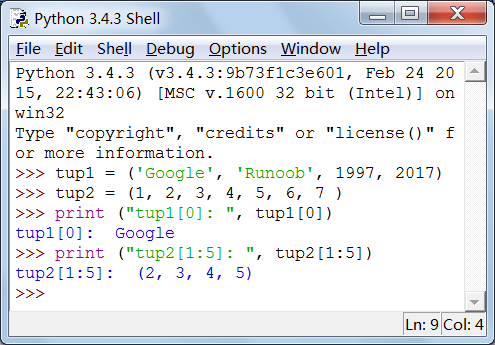
* 当元组中只包含一个元素时，需要在元素后面添加逗号，例如：

tup1 = (50,)

* 与字符串类似，元组的下标索引也从0开始，而且也可以进行截取、组合等操作。

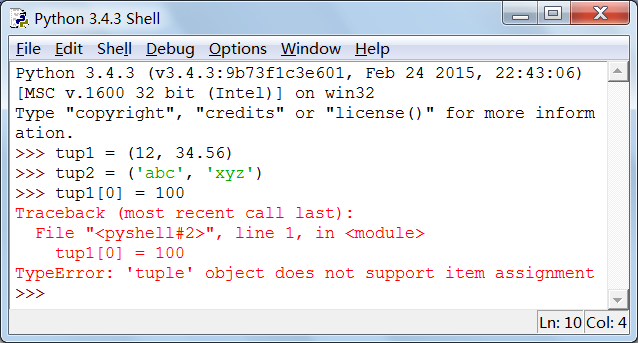
2．元组的访问

可以使用下标索引来访问元组中的元素，例如：

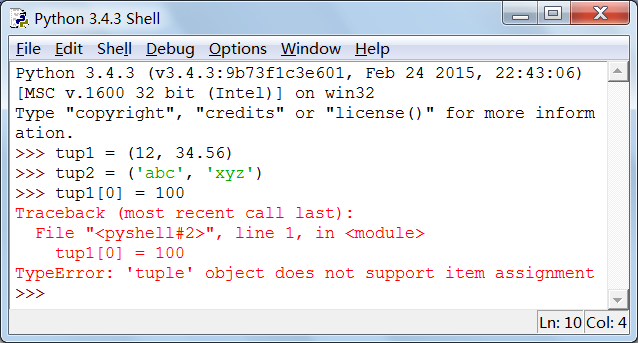


3．元组的修改

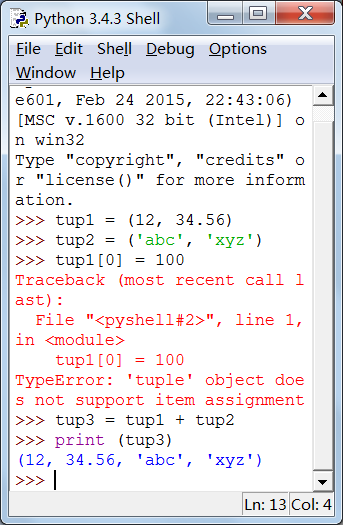
如前所述，元组中的元素值是不允许修改的，但我们可以对元组进行连接组合。



以下修改元组元素的操作是非法的：

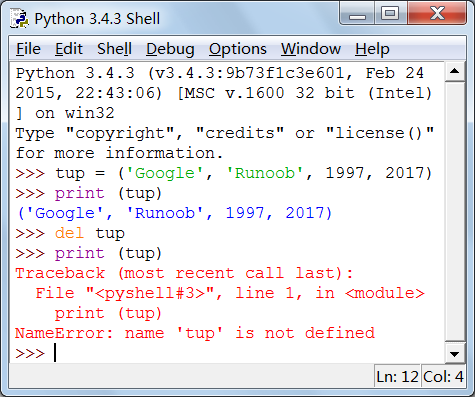


但可以通过连接运算符“+”创建一个新的元组：



4．元组的删除

既然元组不能修改，那么其中的元素值也就不允许删除，但我们可以使用del语句删除整个元组，例如：



可见，元组被删除后，再输出该元组时会显示异常信息。

5．元组运算符

与字符串类似，元组之间可以使用 + 号和 \* 号进行运算。这就意味着可以将它们组合和复制，运算后将生成一个新的元组。表3-5给出了几个例子。

表3-5 Python元组的运算符

|  |  |  |
| --- | --- | --- |
| Python 表达式 | 结果 | 描述 |
| len((1, 2, 3)) | 3 | 计算元素个数。 |
| (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | 连接。 |
| ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | 重复。 |
| 3 in (1, 2, 3) | True | 元素是否存在于元组中。 |
| for x in (1, 2, 3): print (x,end=' ') | 1 2 3 | 遍历并输出元组的各个元素。 |

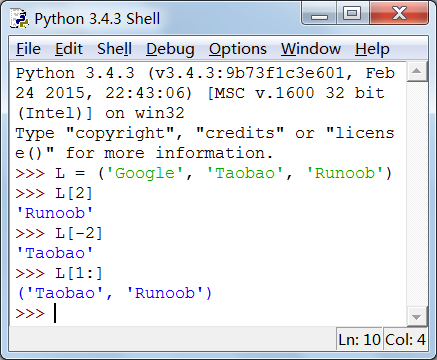
6．元组的索引、截取

前已提及，元组是一个序列，所以可以访问元组中指定位置的元素，也可以通过索引截取元组中的一段元素。假设元组为L = ('Google', 'Taobao', 'Runoob')，表3-6说明了元组的索引和截取。

表3-6 Python元组的索引和截取

|  |  |  |
| --- | --- | --- |
| Python 表达式 | 结果 | 描述 |
| L[2] | 'Runoob!' | 读取第三个元素。 |
| L[-2] | 'Taobao' | 反向读取；读取倒数第二个元素。 |
| L[1:] | ('Taobao', 'Runoob!') | 截取元素，从第二个开始后的所有元素。 |

运行结果如下：



7．元组的内建函数

Python元组包含了一些内建函数，如表3-7所示。

表3-7 Python元组的内建函数

|  |  |  |
| --- | --- | --- |
| 序号 | 方法及描述 | 举例 |
| 1 | len(tuple)  计算元组中元素的个数。 |  |
| 2 | max(tuple)  返回元组中元素的最大值。 |  |
| 3 | min(tuple)  返回元组中元素的最小值。 |  |
| 4 | tuple(seq)  将列表转换为元组。 |  |

#### 三、字典

1．字典的概念

Python的字典（dictionary）是一种可变容器模型，且可以存储任意类型对象。

字典中的每个项都是“键/值对”（有时写做“键-值对”或“键值对”），“键”与“值”之间用冒号(:)分割，而每个“对”之间用逗号(,)分割，整个字典放在花括号({})中，格式如下：

d = {key1 : value1, key2 : value2 }

对每个键/值对而言，键必须是唯一的，但值可以改变。值可以取任何数据类型，但键必须是不可变的。例如，字符串、数字或元组均可作为键，但列表不可以。

2．字典的创建

Python中创建字典的方法很简单，只要将键值对放入花括号内，并用逗号隔开即可。一个简单的字典例子如下：

dict = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}

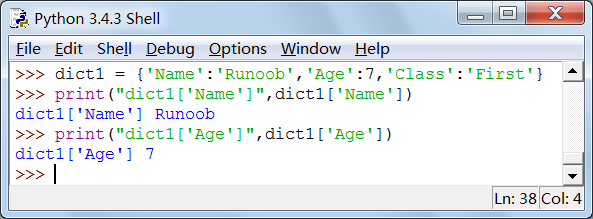
也可这样创建字典：

dict1 = { 'abc': 456 }

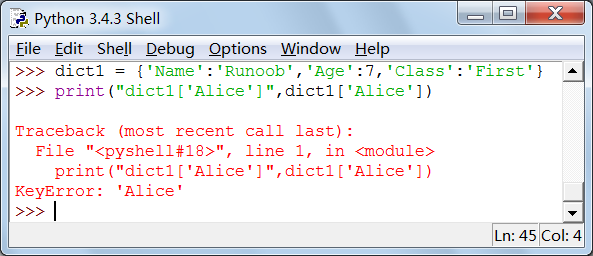
dict2 = { 'abc': 123, 98.6: 37 }

3．字典的访问

把相应的键放入方括号内，即可得到相应的值。例如：

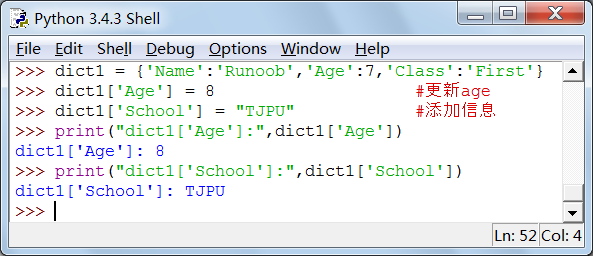


如果所用的键在字典中不存在，则数据无法访问，会输出错误信息，例如：



4．字典的添加与修改

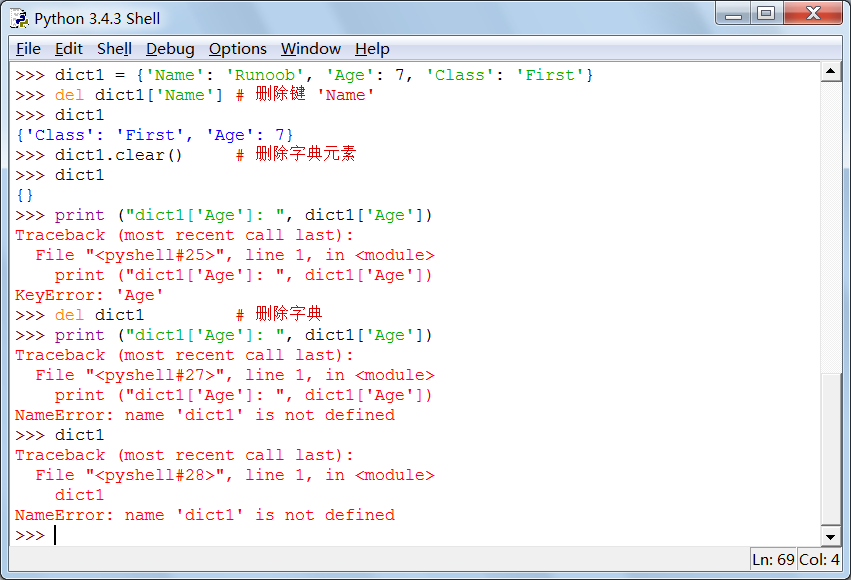
向字典添加新的键/值对，便向字典中添加了新的内容。修改或添加已有键/值对的例子如下:



5．字典元素的删除

既能删除字典中的单一元素，也能将整个字典清空，而且清空只需一项操作。

删除一个字典用del命令，例如：



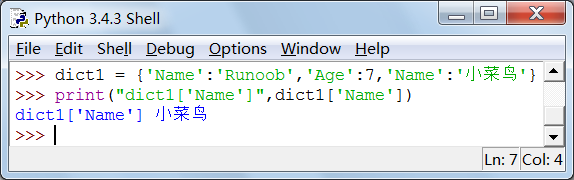
第一次引发异常是因为字典中已无元素（空字典），试图访问键‘age’会出错；第二次引发异常是因为前面已执行了del操作，字典已不复存在，当然不能访问键‘age’；第三次引发异常也是因为字典不存在，所以两次引发异常的名称均为“NameError”。

6．字典键的特性

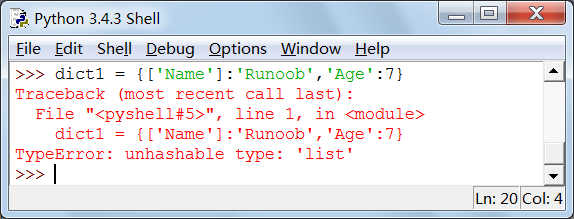
字典的值可以取任何Python对象，没有任何限制，它既可以是标准对象，也可以是用户自己定义的对象，但键不行。

关于字典的键，有两点必须牢记：

（1）同一个键不得出现两次。创建字典时，如果同一个键被赋值两次，则后一个值被记住，例如：



（2）键必须不可变。可以用数字、字符串或元组做键，用列表则不行，例如：



7．字典的内建函数与方法

Python字典包含的内建函数如表3-8所示。

表3-5 Python字典的内建函数

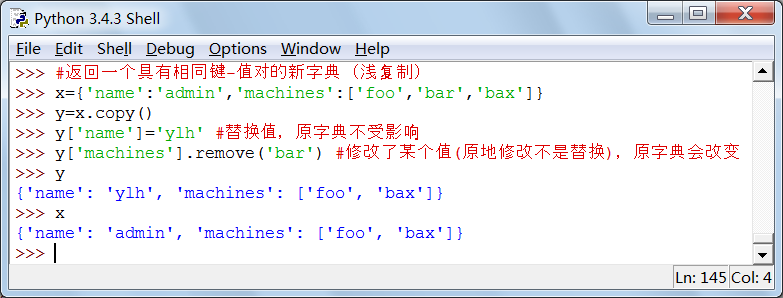
|  |  |  |
| --- | --- | --- |
| 序号 | 函数及描述 | 举例 |
| 1 | len(dict)  计算字典元素个数，即键的总数。 |  |
| 2 | str(dict)  输出字典，以可打印的字符串表示。 |  |
| 3 | type(variable)  返回输入的变量类型，如果变量是字典就返回字典类型。 |  |

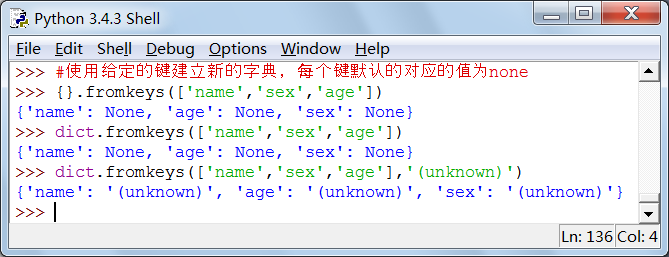
Python字典包含的内建方法如表3-9所示。

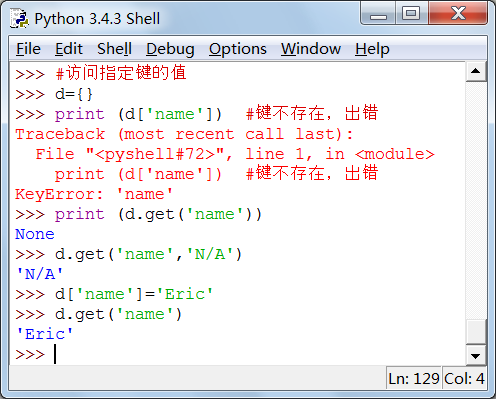
表3-9 Python字典的内建方法

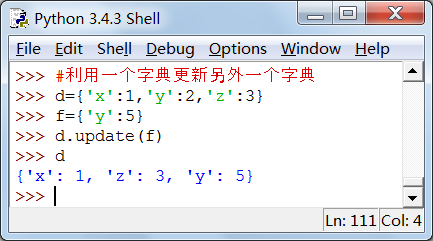
|  |  |
| --- | --- |
| 序号 | 函数及描述 |
| 1 | radiansdict.clear()  删除字典内所有元素 。 |
| 2 | radiansdict.copy()  返回一个字典的浅复制。 |
| 3 | radiansdict.fromkeys(seq[, val])  创建一个新字典，以序列seq中的元素做字典的键，val为字典所有键对应的初始值（缺省时为None）。 |
| 4 | radiansdict.get(key, default=None)  返回指定键的值，如果值不在字典中则返回default值。 |
| 5 | key in dict  如果键在字典dict里，则返回True，否则返回False。 |
| 6 | radiansdict.items()  以列表返回可遍历的(键, 值) 元组数组。 |
| 7 | radiansdict.keys()  以列表返回一个字典所有的键。 |
| 8 | radiansdict.setdefault(key, default=None)  和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default。 |
| 9 | radiansdict.update(dict2)  把字典dict2的键/值对更新到字典里。 |
| 10 | radiansdict.values()  以列表返回字典中的所有值。 |

下面看几个典型的例子。





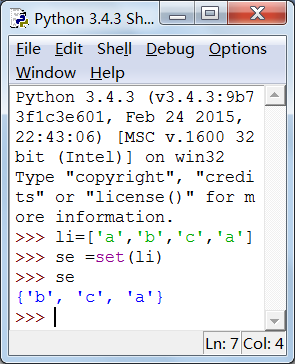




#### 四、集合

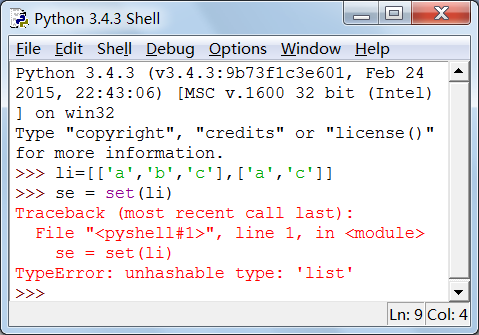
1．集合概述

集合(set)是Python的基本数据类型，把不同的元素组合在一起便形成了集合。组成一个集合的成员称作该集合的元素(element)。在一个集合中不能有相同的元素。下面是集合的例子：



可以看到，相同的元素被自动删除了。

集合对象是一组无序排列的可哈希的值，集合成员可以作为字典的键。相比之下，列表对象是不可哈希的，所以下面的程序会出错。



集合可以分为两类：可变集合（set）与不可变集合（frozenset）。可变集合可添加和删除元素，是非可哈希的，不能用作字典的键，也不能做其他集合的元素。不可变集合与之相反。

2．集合操作符和关系符号

集合有各种操作，各种操作符和关系符号如表3-10所示。

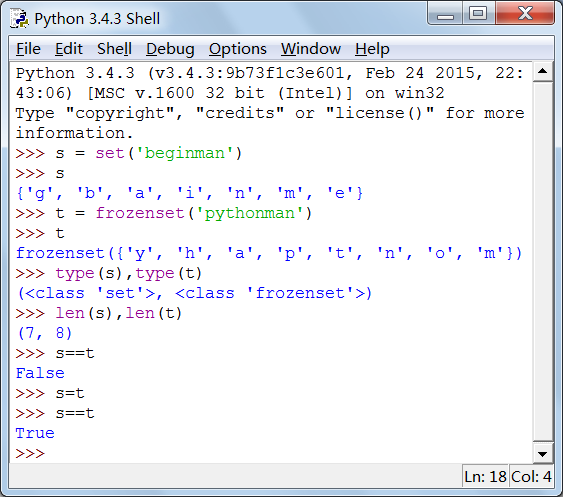
表3-10集合操作符和关系符号

|  |  |  |
| --- | --- | --- |
| 数学符号 | Python符号 | 说明 |
| ∈ | in | 是……的成员 |
|  | not in | 不是……的成员 |
| = | == | 等于 |
| ≠ | != | 不等于 |
|  | < | 是……的真子集 |
|  | <= | 是……的子集 |
|  | > | 是……的真超集 |
|  | >= | 是……的超集 |
| ∩ | & | 交集 |
| ∪ | | | 并集 |
| -或\ | - | 差集或相对补集 |
| Δ | ^ | 对称差分 |

3．集合的相关操作

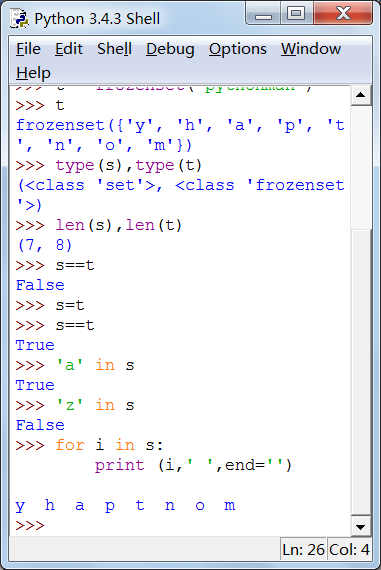
（1）集合的创建

由于集合没有自己的语法格式，只能通过集合的工厂方法set()和frozenset()来创建。



（2）集合的访问

由于集合本身是无序的，所以不能像列表和元组那样，为集合创建索引或进行切片操作，只能循环遍历或使用in、not in来访问或判断集合元素。有关循环的内容将在下一章中作具体介绍。



（3）集合的更新

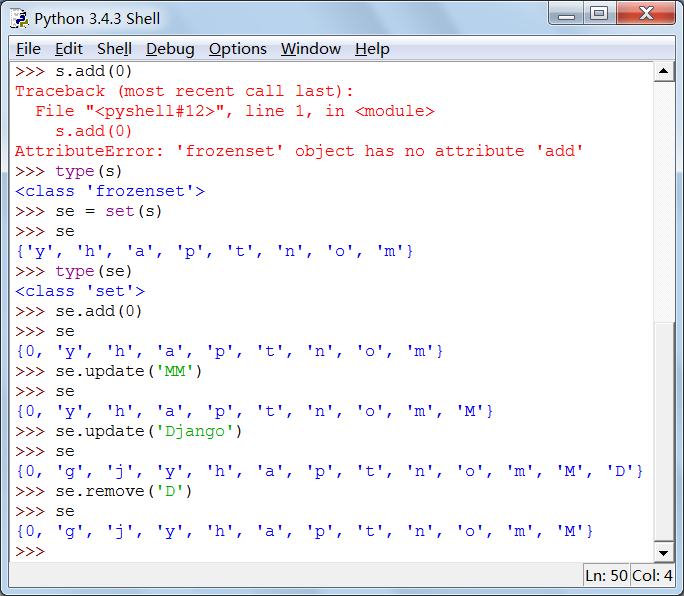
Python内建了以下方法，可以实现集合的更新：

s.add()

s.update()

s.remove()

当然，只有可变集合才能更新，试图更新不可变集合将会出错。例如：



内建的del命令可以删除集合本身。

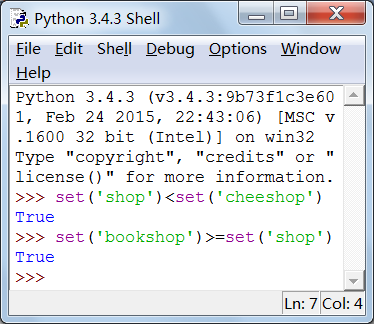
4．集合类型操作符

集合类型操作符有7类：

（1）in ,not in（是否是集合的元素）

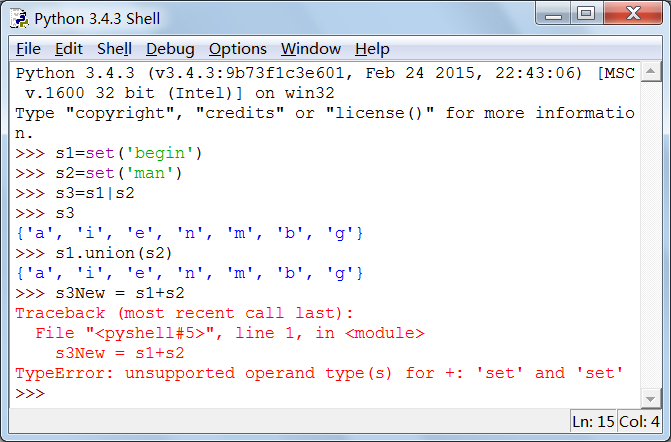
（2）==,!=（集合等价与不等价）

（3）子集、超集(见表2-17)

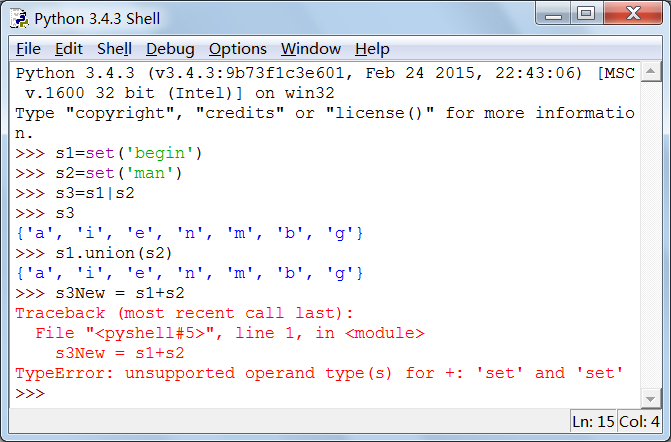


（4）联合(|)

联合(union)操作与集合的or操作其实是等价的，联合操作符还有一个与之等价的方法union()。

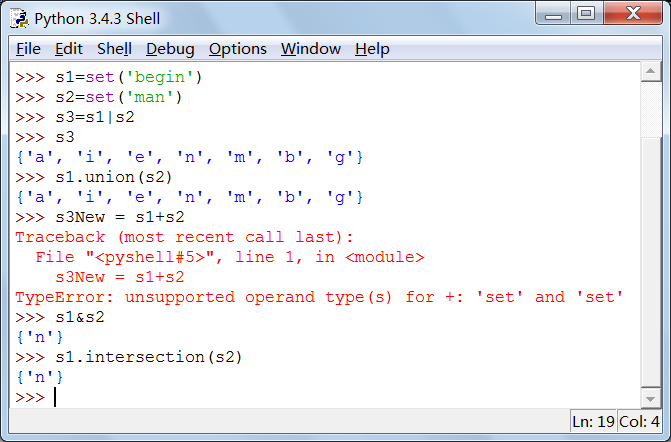


但 + 运算则不可用于集合：



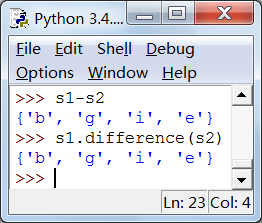
（5）交集(&)

与集合的and操作等价，交集符号&的等价方法是intersection()。



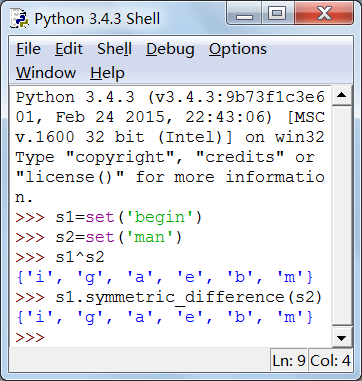
（6）差补(-)

与之等价的方法是difference()。

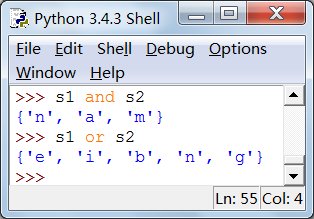


（7）对称差分(^)

对称差分是集合的xor（异或），取得的元素属于s1和s2，但不同时属于s1和s2，其等价的方法是symmetric\_difference()。

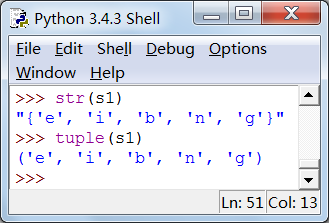


注意集合之间的and,or运算：



可见，s1 and s2运算取的是s2，而s1 or s2运算取的是s1。

5．集合转换为字符串、元组



6．关于集合的内建函数、内建方法

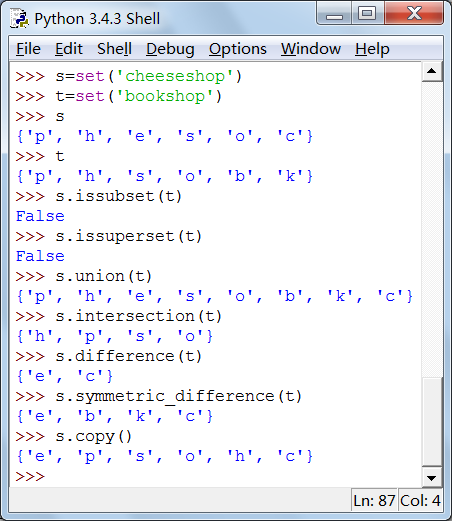
（1）len()：返回集合元素的个数

（2）set()、frozenset()：创建集合（属工厂函数）

（3）适合所有集合的方法（见表3-11）

表3-11 适合所有集合的方法

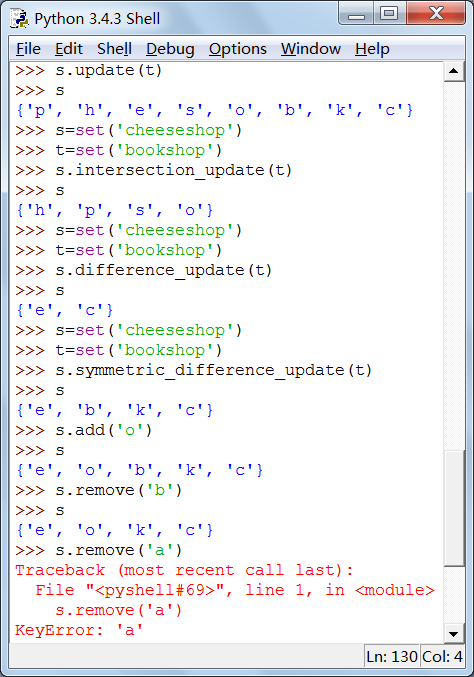
|  |  |
| --- | --- |
| 方法名称 | 操作 |
| s.issubset(t) | 如果s是t的子集，则返回True，否则返回False。 |
| s.issuperset(t) | 如果s是t的超集，则返回True，否则返回False。 |
| s.union(t) | 返回一个新集合，该集合是s和t的并集。 |
| s.intersection(t) | 返回一个新集合，该集合是s和t的交集。 |
| s.difference(t) | 返回一个新集合，该集合是s的成员，但不是t的成员。 |
| s.symmetric\_difference(t) | 返回一个新集合，该集合是s或t的成员，但不是s和t共有的成员。 |
| s.copy() | 返回一个新集合，它是集合s的浅复制。 |



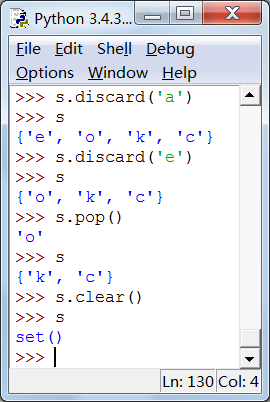
（4）仅适合可变集合的方法（见表3-12）

表3-12仅适合可变集合的方法

|  |  |
| --- | --- |
| 方法名称 | 操作 |
| s.update(t) | 用t中的元素修改s，即s现在包含s或t的成员。 |
| s.intersection\_update(t) | s中的成员是共同属于s和t中的元素。 |
| s.difference\_update(t) | s中的成员是属于s但不包含在t中的元素。 |
| s.symmetric\_difference\_update(t) | s中的成员更新为那些包含在s或t中，但不是s和t共有的元素。 |
| s.add(obj) | 在集合s中添加对象obj。 |
| s.remove(obj) | 从集合s中删除对象obj，如果obj不是集合s中的元素（obj not in s），将引发KeyError错误。 |
| s.discard(obj) | 如果obj是集合s中的元素，从集合s中删除对象obj。 |
| s.pop() | 删除集合s中的任意一个对象，并返回它。 |
| s.clear() | 删除集合s中的所有元素。 |



‘a’元素已不在集合中，试图删除将引发异常。



#### 五、练习

1．列表练习：写出各条命令的执行结果。

names = [ "Dave", "Mark", "Ann", "Phil" ]

a = names[2]

print (a)

names[0] = "Jeff"

print (names)

print (len(names))

names.append("Kate")

print (names)

names.insert(2, "Sydney")

print (names)

print(names[0:2])

print(names[2:])

names[1] = 'Jeff'

print (names)

names[0:2] = ['Dave','Mark','Jeff']

print (names)

a = [1,2,3] + [4,5]

print (a)

a = [1,"Dave",3.14, ["Mark", 7, 9, [100,101]], 10]

print(a[1])

print(a[3][2])

print(a[3][3][1])

2．元组练习：写出各条命令的执行结果。哪条命令会出错，为什么？

tup1 = ('red', 'green', 1997, 2000)

tup2 = (1, 2, 3, 4, 5, 6, 7 )

print ("tup1[0]: ", tup1[0])

print ("tup2[1:5]: ", tup2[1:5])

print ("tup2[3:-1]: ", tup2[3:-1])

tup1 = (12, 34.56)

tup2 = ('abc', 'xyz')

tup3 = tup1 + tup2

print (tup3)

tup = ('red', 'green', 1997, 2000)

print (tup)

del tup

print ("After deleting tup : ")

print (tup)

3．字典练习：写出下列各条命令的执行结果。

info = {"stu01":"张三","stu02":"李四","stu03":"王五"}

print(info)

print(info["stu01"])

print(info.get("stu04"))

print("stu03" in info)

info["stu02"] = "李四四"

print(info)

info["stu04"] = "赵六"

print(info)

del info["stu04"]

info.pop("stu03")

print(info)

4．集合练习：写出下列各条命令的执行结果。

(1)第一组

list\_1 = [1,3,4,7,3,6,7,9]

list\_1 = set(list\_1)

list\_2 = set([2,6,0,66,22,8,4])

list\_3 = set([1,3,7])

print(list\_1)

print(list\_2)

print(list\_3)

print(list\_1.intersection(list\_2))

print(list\_1.union(list\_2))

print(list\_1.difference(list\_2))

print(list\_1.symmetric\_difference(list\_2))

print(list\_3.issubset(list\_1))

print(list\_1.issuperset(list\_3))

list\_4 = set([5,6,8])

print(list\_3.isdisjoint(list\_4))

(2)第二组

s = set([3,5,9,10]);print("s=",s)

t = set("Hello");print("t=",t)

a = t | s ;print("a=",a)

b = t & s ;print("b=",b)

c = t - s ;print("c=",c)

d = t ^ s ;print("d=",d)

t.add('x');print("t=",t)

s.update([10,37,42]);print("s=",s)

t.remove('H') ;print("t=",t)

t.pop() ;print("t=",t)

t.discard('H') ;print("t=",t)

print(len(s))

print('x' in s)

print('x' not in s)

s.issubset(t);print(s)

print(s <= t)

print(s.issuperset(t))

print(s >= t)

print(s.union(t))

print(s | t)

print(s.intersection(t))

print(s & t)

print(s.difference(t))

print(s - t)

print(s.symmetric\_difference(t))

print(s ^ t)

print(s.copy())