第七章：映射和集合类型

前言：本章讲解映射和集合类型。分别从操作符，工厂函数、内建函数(BIF)和方法以及相关模块来介绍。

目录

[7.1映射类型：字典 1](#_Toc350504653)

[1、字典方面的操作 1](#_Toc350504654)

[2·映射类型的操作符 3](#_Toc350504655)

[3、映射类型相关的函数 4](#_Toc350504656)

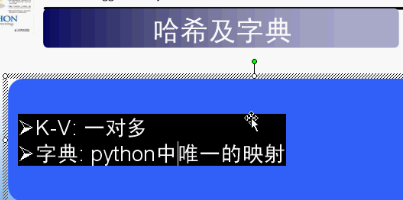
[7.2集合类型 6](#_Toc350504657)

[1、集合类型的操作 6](#_Toc350504658)

[2、集合类型操作符 8](#_Toc350504659)

[3、内建函数 9](#_Toc350504660)

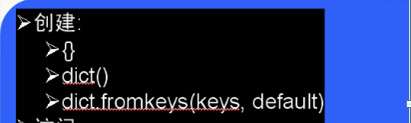
## 7.1映射类型：字典



Python中的字典是一对1的。Python中字典使用keys()获得键的列表, values()获得值的列表, items()获得包含key->value对的元组的列表.

### 1、字典方面的操作

1）如何创建字典和给字典赋值



dict = {key: value}

dict = dict((key, value), (key, value))

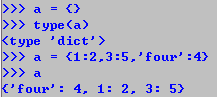
dict.fromkeys(sequence\_keys, default\_value) 创建一个key是sequence\_keys中元素的字典, 所有的value都是default\_value, 如果不指定default\_value, 默认是None

使用dict.has\_key()可以判断一个字典中是否有这个键, 该方法在后期python可能弃用, 推荐使用in和not in

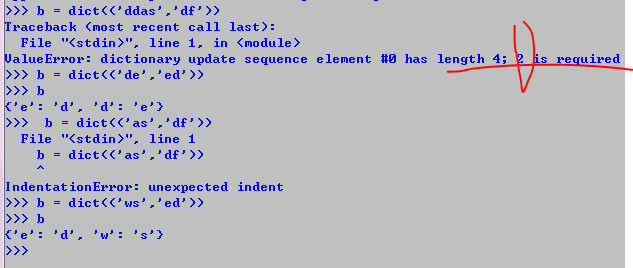
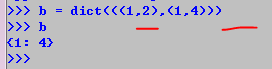
字典的键必须是可哈希的

print中使用到字典的时候, 使用字符串格式化方式是非常优雅的。

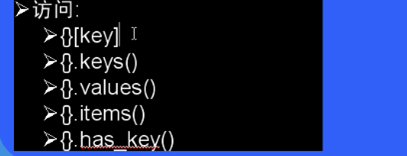
二元序列的序列就行。



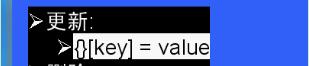




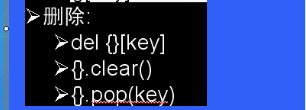
2）怎么访问字典



3）如何更新字典



4）如何删除字典

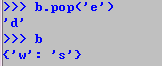


元素的删除

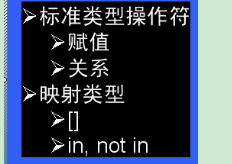
del dict[“key”] 删除键是key的条目

dict.clear() 清空字典内的内容

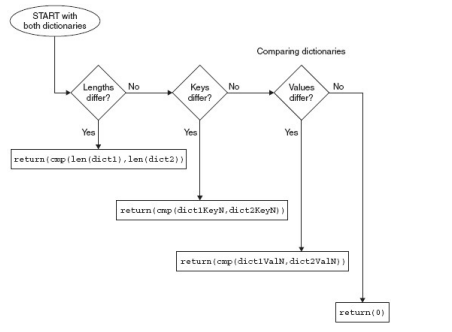
dict.pop(“name”) 删除键是key的条目并返回他的value



### 2·映射类型的操作符

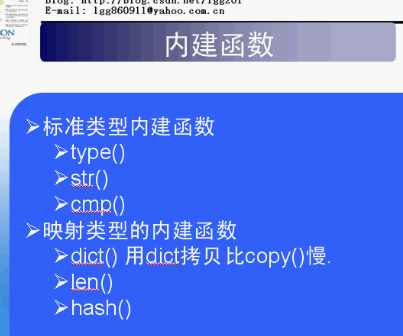


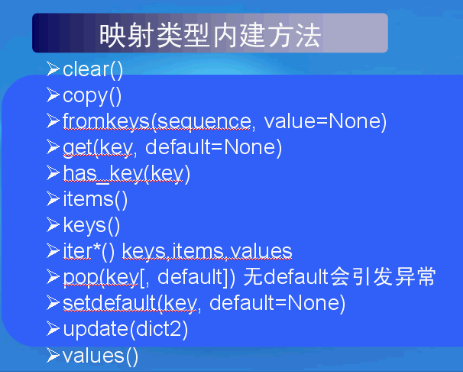
1）标准类型操作符: <, >等比较操作符可以使用, 在比较过程中, 还是调用了字典的cmp方法, 但是, 字典的cmp方法中指示, 首先比较字典的长度, 然后比较键的大小, 最后比较值的大小。



1. 字典查找操作: [], 成员关系操作: in, not in

### 3、映射类型相关的函数





1）dict工厂函数

接受不定关键字参数: dict(a = 1, b = 2, c = 3)

接受字典或关键字参数: 将原有的字典拷贝出来成为一个新的字典(这里使用的浅拷贝, 这里的浅拷贝得到的结果和使用copy函数得到的结果是一样的,但是, copy函数的效率更高)

dict\_instance.copy() 使用已有的字典拷贝一个字典(这里使用的也是浅拷贝)

1. 系统内建函数

len(): 返回字典的key-value对的数目

hash(): 这个函数不是为字典设计的, 但是它可以判断某个对象是否可以做一个字典的键, 将一个对象作为参数传递给hash(), 会返回这个对象的哈希值. 如果对象是不可哈希的, 会返回TypeError, 提示该对象是unhashable的

sorted(dict): 返回dict的所有key排序后的列表

1. dict类型的内建方法

keys(): 返回字典的所有key的列表

values(): 返回字典的所有value的列表

items(): 返回字典的所有key-value的元组的列表

get(key, default = None): 获取字典内key对应的值, 如果没有该key, 返回default指定的默认值

setdefault(key, default=None): 如果字典中不存在key, 由dict[key]=default为其赋值

iterkeys(), itervalues(), iteritems() 对应没有iter命名的方法, 这里使用了惰性赋值的迭代器, 节省内存

1. 数字作为字典的键的时候, 只要值相同, 就代表相同的键, 比如, 1, 和1.0代表的就是相同的键

5）键必须是可哈希的, 所有的不可变对象都是可哈希的, 可变对象如果在定义中定义了\_\_hash\_\_()方法, 那么就可以作为键

1. 如果元组中的值都是不可变类型的, 那么元组也可以作为字典的键

## 7.2集合类型

集合sets有两种不同的类型, 可变集合set和不可变集合frozenset…可变集合也是不可哈希的…

集合中不能有重复的元素, 如果有元素和已有元素重复, 就不会被插入.集合是无序的, 但是, 可以使用排序函数为它排序

### 1、集合类型的操作

1）如何创建集合类型和给集合赋值

集合与列表( [ ] )和字典( { } ) 不同，没有特别的语法格式。列表和字典可以分别用他们自

己的工厂方法 list() 和 dict() 创建，这也是集合被创建的唯一方法 - 用集合的工厂方法 set() 和 frozenset():

2）如何访问集合中的值

你可以遍历查看集合成员或检查某项元素是否是一个集合中的成员:

>>> 'k' in s

False

>>> 'k' in t

True

>>> 'c' not in t

True

>>> for i in s:

... print i

...

c

e

h

o

p

s

3）如何更新集合

用各种集合内建的方法和操作符添加和删除集合的成员:

>>> s.add('z')

>>> s

set(['c', 'e', 'h', 'o', 'p', 's', 'z'])

>>> s.update('pypi')

>>> s

set(['c', 'e', 'i', 'h', 'o', 'p', 's', 'y', 'z'])

>>> s.remove('z')

>>> s

set(['c', 'e', 'i', 'h', 'o', 'p', 's', 'y'])

>>> s -= set('pypi')

>>> s

注意：我们之前提到过，只有可变集合能被修改。试图修改不可变集合会引发异常。

4）如何删除集合中的成员和集合

前面我们看到如何删除集合成员。如果如何删除集合本身，可以像删除任何 Python 对象一样，令集合超出它的作用范围，或调用 del 将他们直接清除出当前的名字空间。如果它的引用计数为零，也会被标记以便被垃圾回收。

>>> del s

>>>

### 2、集合类型操作符

集合中可以使用的数学符号



1. 集合可用的标准类型操作符
   1. 成员关系: in, not in
   2. 集合等价/不等价: ==, !=
   3. 子集/超集: <, <=, >, >=
2. 集合类型操作符(所有的集合类型) 集合互相操作的时候, 最后产生的集合是可变集合还是不可变集合取决于第一个参与操作的集合的类型
   1. 联合 |
   2. 交集 &
   3. 差补/相对补集 - A-B就返回属于A 但不属于B的元素集合
   4. 对称差分 ^ A^B = A-B + B-A
3. 可变集合特有的操作符
   1. |= 相当于update方法, 并集赋值
   2. &= 相当于intersection\_update()方法, 交集赋值
   3. -= 相当于difference\_update()方法, 差集赋值
   4. ^= 相当于symmetric\_difference\_update()方法, 对称差分更新

### 3、内建函数

1）标准类型函数

Len()

2）集合类型工厂函数

set() and frozenset()

3)集合类型的内部方法(所有集合的方法)

* 1. s.issubset(t) 判断s是不是t的子集
  2. s.issuperset(t) 判断s是不是t的超集
  3. s.union(t) 返回一个新集合, 该集合是s和t的并集
  4. s.intersection(t) 返回一个新集合, 该集合是s和t的交集
  5. 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, 即返回s不同于t的元素
  6. s.symmetric\_defference(t) 返回所有s和t独有的(非共同拥有)元素集合
  7. s.copy() 返回一个s的浅拷贝, 效率比工厂要好

4)可变集合特有的方法: add, remove, discard, pop, clear, 这些接受对象的方法, 参数必须是可哈希的

5）操作符和内建方法比较

像你看到的， 很多内建的方法几乎和操作符等价。我们说"几乎等价"，意思是它们间是有一个重要区别： 当用操作符时，操作符两边的操作数必须是集合。 在使用内建方法时，对象也可以是迭代类型的。