


Chap08– Dictionary and Set

College of Computer Science
Nankai University



Outline

- ▶ Dictionary
 - Methods
 - Create
 - Access
 - Add or Update
 - Delete
 - Exist
 - ▶ Set
 - Create
 - Access
 - Operation
 - Methods
 - Application
- 

Dictionary

- ▶ 字典是python中唯一内置映射数据类型，可以通过指定的键从字典访问值。
- ▶ 字典是一个由键和值组成的键值对构成的集合，每一个字典元素分为两部份：键(key)和值（value）。
- ▶ 字典中每个元素键的取值是无序的，但是必须唯一（即集合中不能包含键相同的元素）且必须是可哈希类型的数据，但对于每个元素值的取值则没有任何限制。

Dictionary-Methods

Method	Description
d.keys()	返回字典d中所有键的列表，类型为dict_keys。
d.values()	返回字典d中值的列表，类型为dict_values。
d.items()	返回字典d中由键和相应值组成的元组的列表，类型为dict_items。
d.clear()	删除字典d的所有条目。
d.copy()	返回字典d的浅复制拷贝，不复制嵌入结构。
d.update(x)	将字典x中的键值加入到字典d。
d.pop(k)	删除键值为k的键值对，返回k所对应的值。
d.get(k[,y])	返回键k对应的值，若未找到该键返回None。如果指定参数y的值，则未找到k时返回y的值。

Dictionary-Create

- ▶ 可以使用一对大括号{}或dict函数创建字典，如果要创建空字典可以使用{}或dict()。

```
>>> d1 = {0:'SUN', 1:'MON', 2:'TUE', 3:'WED',  
4:'THU', 5:'FRI', 6:'SAT'}
```

```
>>> d2 = dict(age=18)
```

```
>>> d3 = {}
```

```
>>> d4 = dict()
```

Dictionary-Create

- ▶ dict函数对键值参数名的要求比{}的要求更严格，参数名必须是一个标识符，而不能是表达式。

```
>>> d5 = dict(1="Mon",2="Tue")
```

```
SyntaxError: keyword can't be an expression
```

```
>>> d5 = {1:"Mon",2:"Tue"}
```

```
>>> d5
```

```
{1: 'Mon', 2: 'Tue'}
```

Dictionary-Access

- 字典元素的访问方式是通过键访问相关联的值，访问形式为：<字典>[<键>]。

```
>>> d5 = {1:"Mon",2:"Tue"}
```

```
>>> d5[1]
```

```
'Mon'
```

```
>>> d5[3]
```

```
Traceback (most recent call last):
```

```
File "<pyshell#20>", line 1, in <module>
```

```
d5[3]
```

```
KeyError: 3
```

Dictionary-Access

- ▶ 访问字典元素之前，需要判断是否在字典里

```
>>> i = 3
```

```
>>> if i in d5:  
    d5[i]
```


Dictionary-Add or Update

- ▶ 对指定键的元素赋值时，如果该键在字典中已存在，则会将该键对应的元素值做修改；如果该键在字典中不存在，则会在字典中插入一个新元素。

```
>>> d1 = {1:'MON', 2:'TUE', 3:'WED', 4:'THU',  
5:'FRI', 6:'SAT'}
```

```
>>> d1[0] = 'SUN'
```

```
>>> d1
```

```
{1: 'MON', 2: 'TUE', 3: 'WED', 4: 'THU', 5: 'FRI', 6:  
'SAT', 0: 'SUN'}
```

```
>>> d1[0] = 'Sunday'
```

Dictionary-Add or Update

- 字典中的update方法一次修改或插入多个元素

```
>>> d1 = {1: 'MON', 2: 'TUE', 3: 'WED', 4: 'THU', 5: 'FRI', 6: 'SAT', 0: 'SUN'}
```

```
>>> d1.update({0: 'Sun', 1: 'Mon'})
```

```
>>> d1
```

```
{1: 'Mon', 2: 'TUE', 3: 'WED', 4: 'THU', 5: 'FRI', 6: 'SAT', 0: 'Sun'}
```

Dictionary-Delete

- ▶ 使用del可以删除字典元素
- ▶ 使用字典中的pop方法删除指定键的元素。从字典中删除键为key的元素并返回该元素的值；>>>
d1 = {1: 'MON', 2: 'TUE', 3: 'WED', 4: 'THU', 5: 'FRI', 6: 'SAT', 0: 'SUN'}
>>> d1.pop(0)
'Sun'
>>> d1
{1: 'Mon', 2: 'TUE', 3: 'WED', 4: 'THU', 5: 'FRI', 6: 'SAT'}

Dictionary-Delete

- ▶ 如果字典中不存在键为key的元素，报KeyError的错误。

```
>>> d1.pop(8)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#35>", line 1, in <module>
```

```
    d1.pop(8)
```

```
KeyError: 8
```

Dictionary-Exist

- ▶ 两种方法判断字典中是否存在某个键，一种方法是使用字典中的get方法，一种方法是使用成员运算符in。

Dictionary-Exist

▶ 成员运算符in

```
>>> d = {0:'SUN', 1:'MON', 2:'TUE', 3:'WED',  
4:'THU', 5:'FRI', 6:'SAT'}
```

```
>>> 0 in d
```

```
True
```

```
>>> 7 in d
```

```
False
```



Dictionary-Exist

- 字典的get(key)方法从字典中获取键为key的元素的值并返回。如果在字典中不存在键为key的元素，则返回default参数指定的值。

```
>>> d = {0:'SUN', 1:'MON', 2:'TUE', 3:'WED',  
4:'THU', 5:'FRI', 6:'SAT'}
```

```
>>> d.get(0)
```

```
'SUN'
```

```
>>> d.get(7)
```

```
>>> type(d.get(7))
```

```
<class 'NoneType'>
```

Set

- ▶ 集合的特点
 - 无序的，不能通过数字进行索引
 - 非重复的，不能出现重复的元素
- ▶ 集合中不能包含有重复值的元素。如果创建集合或向集合中插入元素时，指定的元素具有重复值，则集合会自动过滤掉重复值的元素、使得每种取值的元素只保留一个。

Set-Create

- ▶ Python的集合可分为可变集合（set）和不可变集合（frozenset）
- ▶ 可变集合可以添加和删除元素，
- ▶ 不可变集合不允许添加和删除元素
- ▶ 可以使用一对大括号{}或set函数创建可变集合，如果要创建空集合则只能使用set函数。

Set-Create

- ▶ 使用大括号{}创建可变集合

```
>>> s = {1,2,3,4}
```

```
>>> s
```

```
{1, 2, 3, 4}
```

```
>>> type(s)
```

```
<class 'set'>
```

```
>>> s = {}
```

```
>>> type(s)
```

```
<class 'dict'>
```

Set-Create

- ▶ set函数的参数是容器对象，可以是字符串，列表和元组，它可以将序列的数据元素作为集合set的元素。

```
<class 'set'>
```

```
>>> s1 = set('hello')
```

```
>>> s1
```

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

```
>>> s2 = set([1,2,3,4])
```

```
>>> s2
```

```
{1, 2, 3, 4}
```

Set-Access

- ▶ 由于集合是无序的，所以不能为集合创建索引或切片操作，只能循环遍历或使用in、not in来访问或判断集合元素。

```
>>> s1 = {'FRI', 'SUN', 'WED', 'SAT', 'THU', 'TUE',  
'MON'}
```

```
>>> 'SUN' in s1
```

```
True
```

```
>>> 'SON' in s1
```

```
False
```

```
>>> for i in s1:  
    print(i, end=" ")
```

FRI SUN WED SAT THU TUE MON

Set-Operation

运算	描述	运算	描述
$x \in s$	检测 x 是否在集合 s 中	$x \notin s$	检测 x 是否不在集合 s 中
$s1 \mid s2$	并集	$s1 == s2$	判断集合是否相等
$s1 \& s2$	交集	$s1 \leq s2$	判断 $s1$ 是否是 $s2$ 的子集
$s1 - s2$	差集	$s1 < s2$	判断 $s1$ 是否是 $s2$ 的真子集
$s1 \wedge s2$	异或集，求 $s1$ 与 $s2$ 中相异元素	$s1 \geq s2$	判断 $s1$ 是否是 $s2$ 的超集
$s1 \mid= s2$	将 $s2$ 的元素并入 $s1$	$s1 > s2$	判断 $s1$ 是否是 $s2$ 的真超集

Set-Operation

```
>>> s2 = {'hello'}
```

```
>>> s3 = {'here', 'hello', 'he', 'her'}
```

```
>>> s2 <= s3
```

True

```
>>> s3 > s2
```

True

```
>>> s2 >= s3
```

False

```
>>> s2 == s3
```

False



Set-Operation

```
>>> s4 = {'hen','height','her'}
```

```
>>> s4 |= s2
```

```
>>> s4
```

```
{'hen', 'her', 'height', 'hello'}
```

```
>>> s4 & s3
```

```
{'her', 'hello'}
```

```
>>> s4 | s3
```

```
{'her', 'hello', 'hen', 'he', 'here', 'height'}
```

```
>>> s4 - s3
```

```
{'hen', 'height'}
```

```
>>> s4 ^ s3
```

```
{'hen', 'he', 'here', 'height'}
```

Set-Method

	方法	描述
	<code>s1.union(s2)</code>	<code>s1 s2</code> ，返回一个新的集合对象
	<code>s1.difference(s2)</code>	<code>s1-s2</code> ，返回一个新的集合对象
	<code>s1.intersection(s2)</code>	<code>s1&s2</code> ，返回一个新的集合对象
	<code>s1.issubset(s2)</code>	<code>s1<=s2</code>
	<code>s1.issuperset(s2)</code>	<code>s1>=s2</code>
*	<code>s1.update(s2)</code>	将s2的元素并入s1
*	<code>s1.add (x)</code>	增加元素x到s1
*	<code>s1.remove(x)</code>	从s1移除x，x不存在报错
*	<code>s1.clear ()</code>	清空s1
	<code>s1.copy()</code>	复制s1，返回一个新的集合对象
	<code>s1.union(s2)</code>	<code>s1 s2</code> ，返回一个新的集合对象
	<code>s1.difference(s2)</code>	<code>s1-s2</code> ，返回一个新的集合对象

Set-Application

- ▶ 通过set函数建立列表的去重复集合元素

```
>>> L1 = [1,2,3,4,1,2,3,4]
```

```
>>> s4=set(L1)
```

```
>>> s4
```

```
{1, 2, 3, 4}
```

```
>>> L2=list(set(L1))
```

```
>>> print L2
```

```
[1, 2, 3, 4]
```

