



Chap4 Dictionary and Set

第4章 字典与集合

Department of Computer Science and Technology
Department of University Basic Computer Teaching



字典

Dictionary

集合

Set

4.1

字典

为什么要使用字典？



某公司人事部门让技术部门用Python构建一个简易的员工信息表，包含员工的姓名和工资信息。根据信息表查询员工牛云的工资。

Source

```
>>> names = ['Mayue', 'Lilin', 'Wuyun']
```

```
>>> salaries = [3000, 4500, 8000]
```

```
>>> print(salaries[names.index('Lilin')])
```

```
4500
```

→ salaries['Lilin']

- 什么是字典?——一种映射类型

- 键 (key)
- 值 (value)
- key-value对

键是**唯一**的：

数字
字符串
元组

- `aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}`

| key | value |
|---------|-------|
| 'Mayue' | 3000 |
| 'Lilin' | 4500 |
| 'Wuyun' | 8000 |



```
>>> sorted(aInfo)
['Lilin', 'Mayue', 'Wuyun']
```

4.1.1 创建字典

创建字典

8

直接创建



```
>>> alInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```


用dict()函数创建

Source

```
>>> info = [('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)]
>>> blnfo = dict(info)
>>> print(blnfo)
{'Lilin': 4500, 'Wuyun': 8000, 'Mayue': 3000}
>>> clnfo = dict(['Mayue', 3000], ['Lilin', 4500], ['Wuyun', 8000]))
>>> dlnfo = dict(Mayue = 3000, Lilin = 4500, Wuyun = 8000)
>>> elnfo = dict(((Mayue, 3000), (Lilin, 4500), (Wuyun, 8000)))
```

用方法fromkeys(seq[, value])创建



```
>>> glInfo = {}.fromkeys(('Mayue', 'Lilin', 'Wuyun'), 3000)
>>> print(glInfo)
{'Lilin': 3000, 'Mayue': 3000, 'Wuyun': 3000}
```



创建员工信息表时将所有员工的工资默认值设置为3000

生成字典



已知有姓名列表和工资列表，如何生成字典类型的员工信息表？



```
>>> names = ['Mayue', 'Lilin', 'Wuyun']  
>>> salaries = [3000, 4500, 8000]  
>>> dict(zip(names, salaries))  
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```

生成字典



对于几个公司的财经数据，如何构造公司代码和股票价格的字典？

```
pList = [('AXP', 'American Express Company', '78.51'),  
         ('BA', 'The Boeing Company', '184.76'),  
         ('CAT', 'Caterpillar Inc.', '96.39'),  
         ('CSCO', 'Cisco Systems, Inc.', '33.71'),  
         ('CVX', 'Chevron Corporation', '106.09')]
```

生成字典



对于几个公司的财经数据，如何构造公司代码和股票价格的字典？

```
aDict = {'AXP': '78.51', 'BA': '184.76', 'CAT ': '96.39',  
'CSCO': '33.71', 'CVX': '106.09'}
```

算法分析：可用循环将公司代码和股票价格分别append到一个新列表中，再利用zip()和dict()函数将这两个列表转化为字典。

生成字典

F

Filename: createdict.py

```
aList = []
```

```
bList = []
```

```
for i in range(5):
```

```
    aStr = pList[i][0]
```

```
    bStr = pList[i][2]
```

```
    aList.append(aStr)
```

```
    bList.append(bStr)
```

```
aDict = dict(zip(aList, bList))
```

```
print(aDict)
```

其中：

```
pList = [('AXP', 'American Express Company', '78.51'),  
         ('BA', 'The Boeing Company', '184.76'),  
         ('CAT', 'Caterpillar Inc.', '96.39'),  
         ('CSCO', 'Cisco Systems,Inc.', '33.71'),  
         ('CVX', 'Chevron Corporation', '106.09')]
```

4.1.2 字典的基本操作

字典的基本操作



键值
查找



字典
更新



添加
元素



成员
判断



删除
元素

1. 键值查找

S
ource

```
>>> alnfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```

```
>>> alnfo['Lilin']
```

```
4500
```

2. 字典更新



```
>>> aInfo['Lilin'] = 9999
```

```
>>> aInfo
```

```
{'Wuyun': 8000, 'Mayue': 3000, 'Lilin': 9999}
```

3. 添加元素

S

ource

```
>>> alnfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```

```
>>> alnfo['Liuxi'] = 6000
```

```
>>> alnfo
```

```
{'Wuyun': 8000, 'Liuxi': 6000, 'Mayue': 3000, 'Lilin': 9999}
```

4. 成员判断



```
>>> aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}  
>>> 'Liuyun' in aInfo  
False
```

5. 删除元素



```
>>> alInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}  
>>> del alInfo['Lilin']  
>>> alInfo  
{ 'Mayue': 3000, 'Wuyun': 8000 }
```

字典的内建函数



| |
|--------|
| dict() |
| len() |
| hash() |

```
>>> alnfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```

```
>>> len(alnfo)
```

```
3
```

```
>>> hash('Mayue')
```

```
7716305958664889313
```

```
>>> testList = [1, 2, 3]
```

```
>>> hash(testList)
```

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

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

```
    hash(testList)
```

```
TypeError: unhashable type: 'list'
```

字典的内建函数

S
ource

```
>>> hash('Wangdachui')  
7716305958664889313
```

```
>>> testList = [1, 2, 3]
```

```
>>> hash(testList)
```

Traceback (most recent call last):

File "<pyshell#2>", line 1, in <module>

hash(testList)

TypeError: unhashable type: 'list'

hash() 函数判断对象是否可哈希，因为 'Mayue' 是不可变的字符串，所以可哈希，执行函数后返回它的哈希值，而 testList 是可变的列表，因此它是不可哈希的。

字典方法

| | | | | |
|---------|--------|--------------|----------|----------|
| clear() | copy() | fromkeys() | get() | items() |
| keys() | pop() | setdefault() | update() | values() |

字典方法

keys()

values()

items()

S_{source}

```
>>> alnfo = {'Mayue': 3000, 'Linling': 4500, 'Wuyun': 8000}
```

```
>>> alnfo.keys()
```

```
dict_keys(['Mayue', 'Lilin', 'Wuyun'])
```

```
>>> alnfo.values()
```

```
dict_values([3000, 4500, 8000])
```

```
>>> alnfo.items()
```

```
dict_items([('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)])
```

字典方法

get()

get()方法与用“[]”索引键值有所不同：例如执行 `alInfo['Qiqi']`，键 'Qiqi' 在字典 `alInfo` 中不存在，执行后会直接报错，无法像 `get()` 方法那样返回 `None`。



```
>>> alInfo = {'Mayue': 3000, 'Linling': 4500, 'Wuyun': 8000}
>>> print(alInfo.get('Qiqi'))
None
>>> print(alInfo.get('Lilin'))
4500
>>> alInfo['Qiqi']
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    alInfo['Qiqi']
KeyError: 'Qiqi'
```

字典方法

copy()



```
>>> alnfo = {'Mayue': 3000, 'Linling': 4500, 'Wuyun': 8000}  
>>> alnfoBackup = alnfo.copy()  
>>> alnfoBackup  
{ 'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000 }
```

字典方法

pop()



```
>>> alnfo = {'Mayue': 3000, 'Linling': 4500, 'Wuyun': 8000}  
>>> alnfo.pop('Lilin')  
4500  
>>> alnfo  
{ 'Mayue': 3000, 'Wuyun': 8000 }
```

字典方法

clear()



```
>>> alInfo = {'Mayue': 3000, 'Linling': 4500, 'Wuyun': 8000}  
>>> alInfo.clear()  
>>> alInfo  
{}
```

字典方法

update()

添加字典的键值对，也可以更新已有键的值。



```
>>> anfo={}
>>> blnfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> alnfo.update(blnfo)
>>> alnfo
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> clnfo = {'Mayue': 4000, 'Wanqi': 6000, 'Lilin': 9999}
>>> alnfo
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> alnfo.update(clnfo)
>>> alnfo
{'Mayue': 4000, 'Lilin': 9999, 'Wanqi': 6000, 'Wuyun': 8000}
```

字典方法

setdefault()



```
>>> alInfo.setdefault('Lilin', None)
# 与alInfo.setdefault('Lilin')效果一样
```

```
9999
```

```
>>> alInfo.setdefault('Jinhe', None)
# 与alInfo.setdefault('Jinhe')效果一样
```

```
>>> alInfo.setdefault('Qiqi', 8000)
```

```
8000
```

```
>>> alInfo
```

```
{'Mayue': 4000, 'Lilin': 9999, 'Wanqi': 6000, 'Wuyun': 8000, 'Jinhe': None, 'Qiqi': 8000}
```

字典方法



已知有员工姓名和工资信息表{'Wangdachui':3000, 'Niuyun':2000, 'Linling':4500, 'Tianqi':8000}，如何单独输出员工姓名和工资金额？



```
>>> aInfo = {'Wangdachui': 3000, 'Niuyun': 2000, 'Linling': 4500, 'Tianqi': 8000}
>>> aInfo.keys()
dict_keys(['Wangdachui', 'Niuyun', 'Linling', 'Tianqi'])
>>> aInfo.values()
dict_values([3000, 2000, 4500, 8000])
```


字典方法



人事部门有两份人员和工资信息表，第一份是原有信息，第二份是公司中有工资更改人员和新进人员的信息，如何处理可以较快地获得完整的信息表？

Source

```
>>> aInfo = {'Wangdachui': 3000, 'Niuyun': 2000, 'Linling': 4500}
>>> bInfo = {'Wangdachui': 4000, 'Niuyun': 9999, 'Wangzi': 6000}
>>> aInfo.update(bInfo)
>>> aInfo
{'Wangzi': 6000, 'Linling': 4500, 'Wangdachui': 4000, 'Niuyun': 9999}
```

字典方法



下面两个程序都通过键查找值，区别在哪里？你更喜欢哪一个？

S
ource

```
>>> stock = {'AXP': 78.51, 'BA': 184.76}
>>> stock['AAA']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'AAA'
```

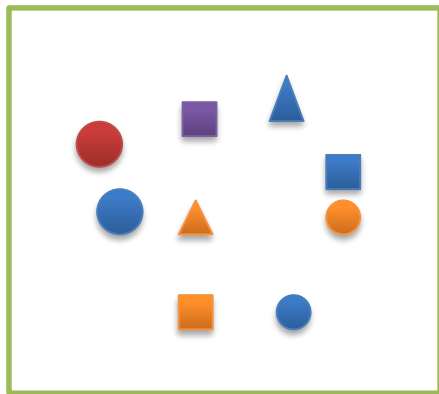
S
ource

```
>>> stock = {'AXP': 78.51, 'BA': 184.76}
>>> print(stock.get('AAA'))
None
```

4.2

用Python玩转数据

集合



去重

- 什么是集合？

一个无序不重复的元素组合

- 可变集合 (set)
- 不可变集合 (frozenset)


A small orange speech bubble icon containing the word "Source" in a stylized font.

```
>>> names = ['Mayue', 'Lilin', 'Wanqi', 'Mayue', 'Lilin']
>>> names
['Mayue', 'Lilin', 'Wanqi', 'Mayue', 'Lilin']
>>> nameset = set(names)
>>> nameset
{'Mayue', 'Wanqi', 'Lilin'}
>>> type(nameset)
<class 'set'>
```

集合的创建

大括号

{ }



```
>>> aSet = set('hello')
>>> aSet
{'h', 'e', 'l', 'o'}
>>> fSet = frozenset('hello')
>>> fSet
frozenset({'h', 'e', 'l', 'o'})
>>> type(aSet)
<class 'set'>
>>> type(fSet)
<class 'frozenset'>
```

集合的基本操作



```
>>> aSet = set('sunrise')
>>> bSet = set('sunset')
>>> 'u' in aSet
True
>>> aSet == bSet
False
>>> aSet < bSet
False
>>> set('sun') < aSet
True
```

| 数学符号 | Python符号 |
|-------------|----------|
| \in | in |
| \notin | not in |
| $=$ | == |
| \neq | != |
| \subset | < |
| \subseteq | <= |
| \supset | > |
| \supseteq | >= |

标准类型运算符

| 数学符号 | Python 符号 | 功能 |
|-------------|---------------------|--------------------|
| \in | <code>in</code> | 是否是集合的成员 |
| \notin | <code>not in</code> | 是否不是集合的成员 |
| $=$ | <code>==</code> | 判断集合是否相等 |
| \neq | <code>!=</code> | 判断集合是否不相等 |
| \subset | <code><</code> | 判断是否是集合的真子集 |
| \subseteq | <code><=</code> | 判断是否是集合的子集（包括非真子集） |
| \supset | <code>></code> | 判断是否是集合的真超集 |
| \supseteq | <code>>=</code> | 判断是否是集合的超集（包括非真超集） |

集合的基本操作

Source

```
>>> aSet = set('sunrise')
>>> bSet = set('sunset')
>>> aSet & bSet
{'u', 's', 'e', 'n'}
>>> aSet | bSet
{'e', 'i', 'n', 's', 'r', 'u', 't'}
>>> aSet - bSet
{'i', 'r'}
```

Source

```
>>> aSet = set('sunrise')
>>> bSet = set('sunset')
>>> aSet ^ bSet
{'i', 'r', 't'}
>>> aSet -= set('sun')
>>> aSet
{'e', 'i', 'r'}
```

| 数学符号 | Python符号 |
|-------------------|--------------------|
| \cap | <code>&</code> |
| \cup | <code> </code> |
| $-$ 或 \setminus | <code>-</code> |
| Δ | <code>^</code> |

集合类型运算符

运算符可复合

`&=` `|=` `-=` `^=`

| 数 学 符号 | Python 符号 | 功能 |
|-----------|--------------|---------|
| \cap | & | 交集 |
| \cup | | 合集 |
| - 或 \ | - | 差补或相对补集 |
| Δ | ^ | 对称差分 |

集合的基本操作

不可变集合 的运算



```
>>> fSet = frozenset('hello')
>>> gSet = set('here')
>>> fSet &= gSet
>>> fSet
frozenset({'h', 'e'})
```

运算结果的类型要与运算符左边的集合类型保持一致

集合内建函数

面向
所有集合

issubset(t)

issuperset(t)

union(t)

intersection(t)

difference(t)

symmetric_difference(t)

copy()



```
>>> aSet = set('sunrise')
```

```
>>> bSet = set('sunset')
```



```
>>> aSet.issubset(bSet)
```

```
False
```

```
>>> aSet.intersection(bSet)
```

```
{'u', 's', 'e', 'n'}
```

```
>>> aSet.difference(bSet)
```

```
{'i', 'r'}
```

```
>>> aSet.symmetric_difference(bSet)
```

```
{'i', 't', 'r'}
```

```
>>> aSet.difference(bSet)
```

```
{'i', 'r'}
```

```
>>> cSet = aSet.copy()
```

```
>>> cSet
```

```
{'s', 'r', 'e', 'i', 'u', 'n'}
```

| 方法名 | 功能 |
|----------------------------------------|---------------------------------------|
| <code>s.issubset(t)</code> | 判断s是否是t的子集 |
| <code>s.issuperset(t)</code> | 判断s是否是t的超集 |
| <code>s.union(t)</code> | 返回新集合，是s和t的并集 |
| <code>s.intersection(t)</code> | 返回新集合，是s和t的交集 |
| <code>s.difference(t)</code> | 返回新集合，是属于s但不属于t的成员组成的集合 |
| <code>s.symmetric_difference(t)</code> | 返回新集合，是只属于其中一个集合的成员（不同时属于s和t），即对称差分操作 |
| <code>s.copy()</code> | 返回集合s的副本 |

集合内建函数

面向
可变集合

`update(t)`

`intersection_update(t)`

`difference_update(t)`

`symmetric_difference_update(t)`

`add(obj)`

`remove(obj)`

`discard(obj)`


`pop()`

`clear()`


| 方法名 | 功能 |
|-----------------------------------------------|----------------------------------|
| <code>s.update(t)</code> | 修改s集合，使s中包含s和t并集的成员 |
| <code>s.intersection_update(t)</code> | 修改s集合，使s中包含s和t交集的成员 |
| <code>s.difference_update(t)</code> | 修改s集合，使s中包含只属于集合s但不属于集合t的成员 |
| <code>s.symmetric_difference_update(t)</code> | 修改s集合，使s中包含只属于s或只属于t但不同时属于s和t的成员 |
| <code>s.add(obj)</code> | 将对象obj添加到集合s中去 |
| <code>s.remove(obj)</code> | 从s中删除对象obj，如果obj不属于s，则报错 |
| <code>s.discard(obj)</code> | 从s中删除对象obj，如果不存在则没有任何操作 |
| <code>s.pop()</code> | 从s中删除任意一个成员，并返回这个成员 |
| <code>s.clear()</code> | 将s中的成员清空 |

集合内建函数

面向 可变集合

 Source

```
>>> aSet = set('sunrise')
>>> aSet.add('!')
>>> aSet
{'!', 'e', 'i', 'n', 's', 'r', 'u'}
>>> aSet.remove('!')
>>> aSet
{'e', 'i', 'n', 's', 'r', 'u'}
>>> aSet.discard('a')
>>> aSet
{'s', 'u', 'e', 'i', 'n'}
```

 Source

```
>>> aSet.remove('a')
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in
<module>
    aSet.remove('a')
KeyError: 'a'
>>> aSet.update('Yeah')
>>> aSet
{'a', 'e', 'i', 'h', 'n', 's', 'r', 'u', 'Y'}
>>> aSet.clear()
>>> aSet
set()
```




人事部门的一份工资信息表登记时由于工作人员的疏忽有部分姓名重复登记了，如何快速解决这个问题？



```
>>> names = ['Wangdachui', 'Niuyun', 'Wangzi', 'Wangdachui', 'Linling', 'Niuyun']
>>> namesSet = set(names)
>>> namesSet
{'Wangzi', 'Wangdachui', 'Niuyun', 'Linling'}
```

- 字典
- 集合

