Chap4 Dictionary and Set

第4章 字典与集合

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

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Linlin的工资。

Source

```
>>> names = ['Mayue', 'Lilin', 'Wuyun']
>>> salaries = [3000, 4500, 8000]
>>> print(salaries[names.index('Lilin')])
4500
```

→ salaries['Lilin']

字典

- 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
不可变对象

字典

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

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

4.1.1 创建字典

创建字典

直接创建

Source

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

创建字典

用dict()函数创建

Source

```
>>> info = [('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)]
>>> bInfo = dict(info)
>>> print(bInfo)
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> cInfo = dict([['Mayue', 3000], ['Lilin', 4500], ['Wuyun', 8000]])
>>> dInfo = dict(Mayue = 3000, Lilin = 4500, Wuyun = 8000)
>>> eInfo = dict(('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)))
```

创建字典

用方法fromkeys(seq[, value])创建

Source

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

Source

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

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

Source

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

2. 字典更新

Source

```
>>> aInfo['Lilin'] = 9999
>>> aInfo
{'Mayue': 3000, 'Lilin': 9999, 'Wuyun': 8000}
```

3. 添加元素

Source

```
>>> aInfo = {'Mayue': 3000, 'Lilin': 9999, 'Wuyun': 8000}
>>> aInfo['Liuxi'] = 6000
>>> aInfo
{'Mayue': 3000, 'Lilin': 9999, 'Wuyun': 8000, 'Liuxi': 6000}
```

3. 添加元素

Source

```
>>> d = {}
>>> d["Liuyue"] = [65,88,90]
>>> d
{'Liuyue': [65, 88, 90]}
>>> d['Majin'] = [89]
>>> d['Majin'] += [94]
>>> d["Majin"] += [85]
>>> d
{'Liuyue': [65, 88, 90], 'Majin': [89, 94, 85]}
```

4. 成员判断

Source

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

5. 删除元素

Source

```
>>> del aInfo
>>> aInfo
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    aInfo
NameError: name 'aInfo' is not defined
>>> aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> del aInfo['Lilin']
>>> aInfo
{'Mayue': 3000, 'Wuyun': 8000}
```

字典的内建函数

dict()
len()
hash()

Source

```
>>> aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> len(aInfo)
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'
```

字典方法

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

字典方法

keys()
values()
items()

Source

```
>>> aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> aInfo.keys()
dict_keys(['Mayue', 'Lilin', 'Wuyun'])
>>> aInfo.values()
dict_values([3000, 4500, 8000])
>>> aInfo.items()
dict_items([('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)])
```

字典方法

get()

Source

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

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？

Source

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

Source

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

字典方法

setdefault()

Source

```
>>> aInfo.setdefault('Lilin', None)
# 与aInfo.get('Lilin')和aInfo.setdefault('Lilin')效果一样
9999
>>> aInfo.setdefault('Jinhe', None)
# 与aInfo.setdefault('Jinhe')效果一样
>>> aInfo.setdefault('Qiqi', 8000)
8000
>>> aInfo
{'Mayue': 4000, 'Lilin': 9999, 'Wanqi': 6000, 'Wuyun': 8000, 'Jinhe': None, 'Qiqi': 8000}
```

字典方法

copy()

Source

```
>>> aInfo = {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> aInfoBackup = aInfo.copy()
>>> aInfoBackup
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
```

字典方法

pop()

Source

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

字典方法

clear()

Source

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

字典方法

update()

Source

```
>>> anfo={'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> bInfo = {}
>>> bInfo.update(aInfo)
>>> bInfo
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> cInfo = {'Mayue': 4000, 'Wanqi':6000, 'Lilin': 9999}
>>> aInfo
{'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
>>> aInfo.update(cInfo)
>>> aInfo
{'Mayue': 4000, 'Lilin': 9999, 'Wuyun': 8000, 'Wanqi': 6000}
```

字典方法简单应用

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
{'Wangdachui': 4000, 'Niuyun': 9999, 'Lilin': 4500, 'Wangzi': 6000}
```

4.2

集合

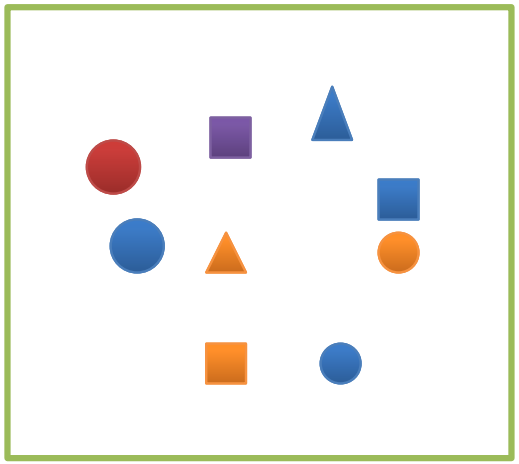
集合

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?

Source

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

集合



去重

- **什么是集合?**

  一个无序不重复的元素的组合

  – 可变集合（set）

  – 不可变集合（frozenset）

集合

Source

```
>>> aSet = {1, 2, 3}
>>> 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

{}

Source

```
>>> 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

Source

```
>>> 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
∉	not in
=	==
≠	!=
⊂	<
⊆	<=
⊃	>
⊇	>=

标准类型运算符

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
∩	&
∪	|
- 或 \	-
Δ	^

集合类型运算符

运算符可复合

&=　|=　-=　^=

集合内建函数

len()

Source

```
>>> aSet = {1,2,3}
>>> type(aSet)
<class 'set'>
>>> len(aSet)
3
```

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()

Source

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

Source

```
>>> aSet.issubset(bSet)
False
>>> aSet.intersection(bSet)
{'u', 's', 'e', 'n'}
>>> aSet.difference(bSet)
{'i', 'r'}
>>> aSet.symmetric_difference(bSet)
{'i', 't', 'r'}
>>> cSet = aSet.copy()
>>> cSet
{'s', 'r', 'e', 'i', 'u', 'n'}
```

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()

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', 'r', '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()
```

小结

- 字典
- 集合