Python程序设计

字典

刘安
苏州大学，计算机科学与技术学院

http://web.suda.edu.cn/anliu/

字典的基本概念

- 字典是对象的无序集合，每个对象是一个键值对
- 通过对象的键来访问该对象相应的值
- 键必须是不可变类型（数值、字符串、元组），值可以是任意类型

- 优点：实现快速的查找（比列表快的多）

```
>>> my_lib = {'Java': 10, 'C++': 5, 'Python': 1}
>>> len(my_lib)
3
>>>
>>> 'Java' in my_lib
True
>>> 'c++' in my_lib
False
```

创建字典的常见方法

- 通过{}创建空字典
- 通过在{}中直接指定键值对创建非空字典
- 通过dict.fromkeys(iterable[, value])方法
  - 以iterable中的元素为键，value为值，如果没有提供value，使用默认值None

```
>>> my_lib = {}
>>> my_lib = {'Java': 10, 'C++': 5, 'Python': 1}
>>> freq = dict.fromkeys('0123456789', 0)
>>> freq
{'0': 0, '1': 0, '2': 0, '3': 0, '4': 0, '5': 0,
'6': 0, '7': 0, '8': 0, '9': 0}
```

修改字典

- 通过赋值操作修改字典，如果键存在，修改成相应的值，否则增加一个新的键值对

```
>>> my_lib['Java'] = 8
>>> my_lib['Python'] += 4
>>> my_lib['Algorithm'] = 30
>>> my_lib
{'Java': 8, 'C++': 5, 'Python': 5, 'Algorithm': 30}
```

- 通过del语句删除指定的键值对，如果键不存在，抛出异常

```
>>> del my_lib['C++']
>>> my_lib
{'Java': 8, 'Python': 5, 'Algorithm': 30}
>>> del my_lib['Calculus']
Traceback (most recent call last):
  File "<pyshell#250>", line 1, in <module>
    del my_lib['Calculus']
KeyError: 'Calculus'
```

获取字典所有的键、值、键值对

- keys()、values()、items()方法分别返回字典所有的键、值、键值对，注意返回结果都是可迭代对象

```
>>> for k in my_lib.keys():
        print(k, end = ' ')
Java Python Algorithm
>>> for pair in my_lib.items():
        print(pair, end = ' ')


('Java', 8) ('Python', 5) ('Algorithm', 30)
>>> my_lib.values()
dict_values([8, 5, 30])
>>> list(my_lib.values()) # create list
[8, 5, 30]
```

获取字典所有的键、值、键值对

- keys()、values()、items()方法分别返回字典所有的键、值、键值对，注意返回结果都是可迭代对象

- 另外也可以通过字典自己的迭代器获取所有的键

```
>>> I = iter(my_lib)
>>> next(I)
'Java'
>>> next(I)
'Python'
>>> next(I)
'Algorithm'
>>>
>>> for k in my_lib:
        print(k, end = ' ')


Java Python Algorithm
```

通过推导来创建字典

- 类似于列表，字典也可以通过推导来创建

```
>>> D = {x: x ** 2 for x in range(5)}
>>> D
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
>>>
>>> D = {c: 0 for c in 'aeiou'}
>>> D
{'a': 0, 'e': 0, 'i': 0, 'o': 0, 'u': 0}
>>>
>>> D = {k: v for k, v in zip('abc', [1, 2, 3])}
>>> D
{'a': 1, 'b': 2, 'c': 3}
>>>
>>> D = {k: v for k, v in D.items() if v % 2 == 1}
>>> D
{'a': 1, 'c': 3}
```

和字典相关的排序

- 字典是无序的，没有sort方法（注意：列表有sort方法）

- 元素的显示顺序取决于元素放入字典的顺序（从3.7开始）

```
>>> stu = {'1011': 80, '1016': 90, '1003': 85, '1008': 95}
>>> stu
{'1011': 80, '1016': 90, '1003': 85, '1008': 95}
>>> del stu['1003']
>>> stu['1003'] = 85
>>> stu
{'1011': 80, '1016': 90, '1008': 95, '1003': 85}
```

- 先通过keys()获取字典的所有键（可迭代对象），然后通过sorted函数对其进行排序，就可以按照一定的顺序输出字典的所有元素，注意这里是键的顺序

和字典相关的排序 - 按照键

- 先通过keys()获取字典的所有键（可迭代对象），然后通过sorted函数对其进行排序，就可以按照一定的顺序输出字典的所有元素，注意这里是键的顺序

```
>>> stu
{'1011': 80, '1016': 90, '1008': 95, '1003': 85}
>>> for k in sorted(stu.keys()): #stu is also fine
        print(k, stu[k])


1003 85
1008 95
1011 80
1016 90
```

和字典相关的排序 - 按照值

- 如果要按照值对字典元素排序，上页的思路不可行：虽然值可以排序，但是无法根据值找到其对应的键

- 值得注意的细节：通过keys()可以拿到所有健，通过values()可以拿到所有值，并且值和键的对应关系恰好和字典里的顺序一样

```
>>> stu
{'1011': 80, '1016': 90, '1008': 95, '1003': 85}
>>> list(stu.keys())
['1011', '1016', '1008', '1003']
>>> list(stu.values())
[80, 90, 95, 85]
```

和字典相关的排序 - 按照值

- 解决方案：通过values()和keys()拿到所有值和所有键，然后通过zip并行遍历值和键，生成（值-键）列表后再排序

```
>>> for pair in sorted(zip(stu.values(), stu.keys())):
        print(pair)

(80, '1011')
(85, '1003')
(90, '1016')
(95, '1008')

>>> list(sorted(zip(stu.values(), stu.keys()), reverse=True))
[(95, '1008'), (90, '1016'), (85, '1003'), (80, '1011')]
```

学生查找

- 编写一个函数，接受一个记录学生成绩的字典，返回
  - 语文成绩最高的学生学号
  - 平均分最高的学生学号
  - 至少有一门课不及格的学生学号
  - 键：字符串，表示学生学号，值：列表，包含语数外三门课的成绩

```
>>> stu = {'1011':[80, 90, 90], '1003':[55, 85, 70],
           '1008':[95, 90, 80], '1005':[75, 70, 50]}
```

```python
def f(x):
    return x[0][0]

def g(x): # len(x[0]) > 0
    return sum(x[0]) / len(x[0])

def h(x):
    for t in x[0]:
        if t < 60:
            return True
    return False
```

```python
L = sorted(zip(stu.values(), stu.keys()), key=f, reverse = True)
print(L[0][1])

L = sorted(zip(stu.values(), stu.keys()), key=h, reverse = True)
print(L[0][1])

L = [t[1] for t in zip(stu.values(), stu.keys()) if h(t)]
print(L)
```

反向查找

- 编写一个函数，接受一个字典和一个值，返回一个列表，包含该字典中所有对应该值的键

```
>>> D = {'1':'a', '2':'b', '3':'a', '4':'c'}
```

- 给定字典D，对于值'a'，返回['1'和'3']，对于值'5'，返回[]

```
1 def reverse_lookup(D, t):
2     res = []
3     for k in D:
4         if D[k] == t:
5             res.append(k)
6     return res
```

```
1 def reverse_lookup_v1(D, t): # 列表推导
2     return [k for k in D if D[k] == t]
```

变位词（anagram）

- 变位词是通过重新排列一个单词的字母形成的新单词，比如ate、eat、tea就是一组变位词
- 编写一个函数，接受一个单词库（用字符串表示，每个单词用空格分隔），根据变位词规则分组，返回一个列表，每个元素是一组变位词（注意：该元素是一个列表）
- 如何判断两个词是变位词：把单词放入sorted函数
- 如何把变位词放在一组：使用字典，键是变位词的唯一表示，值是一个列表，存放相同的变位词
  - 键必须是不可变类型

变位词

```
1 def anagram(s):
2     D = {}
3     words = s.split()
4     for w in words:
5         ks = tuple(sorted(w))
6         if ks not in D: #1
7             D[ks] = []  #2
8         D[ks].append(w) #3
9     return [x for x in D.values()]
```

```
>>> s = 'ate but eat tub tea dealer ladder peat leader tape'
>>> anagram(s)
[['ate', 'eat', 'tea'], ['but', 'tub'], ['dealer', 'leader'],
['ladder'], ['peat', 'tape']]
```

setdefault方法

- setdefault(key[, default])：如果键key存在，返回其对应的值，否则将键值对(key, default)插入字典，并返回值default

```
    for w in words:
        ks = tuple(sorted(w))
        if ks not in D: #1
            D[ks] = []  #2
        D[ks].append(w) #3
```

```
def anagram(s):
    D = {}
    words = s.split()
    for w in words:
        ks = tuple(sorted(w))
        D.setdefault(ks, []).append(w) #4
    return [x for x in D.values()]
```