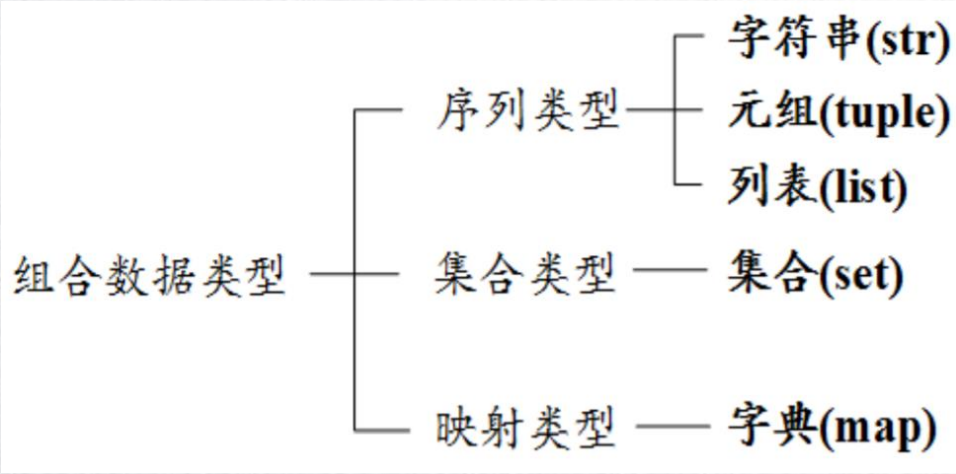
上次课回顾：组合数据类型

- 组合数据类型
  - 组合数据类型概述
    - 序列类型
    - 集合类型
    - 映射类型
  - 字典类型
    - 键和值
    - 创建方法
    - 操作



字典类型的概念

- Python语言中我们可以使用字典（dict）实现映射
  - 字典是包含0个或者多个键值对信息的关联数组
  - 关联数组是支持以下操作的抽象数据类型：
    - 向关联数组添加键值对
    - 从关联数组内删除键值对
    - 修改关联数组内的键值对
    - 根据已知的键寻找键值对
  - 注意：字典中的键是唯一的，无法保存一对多的映射
    - "RUC"=>"人民大学"
    - "RUC"=>"中国人民大学"
    - 解决方法"RUC"=>["人民大学"，"中国人民大学"]

简称（键）	全称（值）
RUC	中国人民大学
THU	清华大学
PKU	北京大学
BIT	北京理工大学
……	……

字典类型的创建

- 字典类型的创建方法有以下几种：
  - 使用{}创建

```
[2]: dict1 = {'RUC': '中国人民大学',
              'THU': '清华大学',
              'PKU': '北京大学',
              'BIT': '北京理工大学',
              'BUAA': '北京航空航天大学'} # 用{}创建
     empty_dict = {} # 可以用这个方式创建空字典

     print(dict1)
     print(empty_dict)
```

```
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空
航天大学'}
{}
```

字典类型的创建

- 字典类型的创建方法有以下几种：
  - 使用构造函数dict()创建

[3]:
```
empty_dict = dict() # 利用构造函数创建字典对象
dict2 = dict(dict1) # 利用另一个dict创建，注意这里dict1中的数据会被拷贝一份
dict3 = dict([('RUC', '中国人民大学'),
              ('THU', '清华大学'),
              ('PKU', '北京大学'),
              ('BIT', '北京理工大学'),
              ('BUAA', '北京航空航天大学')]) # 利用包含 键值对 的序列对象创建
dict4 = dict((['RUC', '中国人民大学'],
              ['THU', '清华大学'],
              ['PKU', '北京大学'],
              ['BIT', '北京理工大学'],
              ['BUAA', '北京航空航天大学'])) # 元组也是序列对象；键值对 可以是列表

print(empty_dict)
print(dict2)
print(dict3)
print(dict4)
```

```
{}
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空
航天大学'}
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空
航天大学'}
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空
航天大学'}
```

字典类型的创建

- 字典类型的创建方法有以下几种：
  - 使用dict()和基于键值对的参数来构建

[4]:
```
# 一种"特殊"的创建方式
dict5 = dict(RUC='中国人民大学',
             THU='清华大学',
             PKU='北京大学',
             BIT='北京理工大学',
             BUAA='北京航空航天大学')
dict6 = dict(dict1, FDU='复旦大学') # 还可以和其他创建方式一起使用

print(dict5)
print(dict6)
```

```
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学', 'FDU': '复旦大学'}
```

字典类型的创建

- Python语言提供了一种基于字典的灵活的给函数传递参数的方式
  - 不需在定义函数的时候指定参数的个数和名称
  - 参数会以字典的形式被传递到函数内

[6]:
```
def g(**kwargs):
    print(type(kwargs))
    print(kwargs)

g()
g(RUC='中国人民大学', THU='清华大学')
```

```
<class 'dict'>
{}
<class 'dict'>
{'RUC': '中国人民大学', 'THU': '清华大学'}
```

字典类型的操作

- 利用键索引字典中保存的值：
  - 可以使用[]操作符访问字典中保存的值
  - 若键不存在，会报错并抛出异常

```
[7]: print(dict1['RUC'])
     print(dict1['FDU']) # 若键信息不在字典里，会报错
```

```
中国人民大学
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-227324f668a8> in <module>
      1 print(dict1['RUC'])
----> 2 print(dict1['FDU']) # 若键信息不在字典里，会报错

KeyError: 'FDU'
```

字典类型的操作

- 利用键索引字典中保存的值：
  - 可以使用get()方法来访问字典中保存的值

```
[8]: print(dict1.get('RUC')) # 也可以使用get方法
     print(dict1.get('FDU', '名称未知')) # get方法的第二个参数是若键信息不在字典里时返回的默认值信息
```

```
中国人民大学
名称未知
```

  - 可以使用 <key> in <d>表达式判断键是否在字典中

```
[9]: print('RUC' in dict1)
     print('FDU' in dict1)
```

```
True
False
```

字典类型的操作

- 通过键增加、修改值信息和删除相应的键值对：
  - 可以使用[]来增加、修改和删除字典中保存的信息

[10]:
```
dict2 = dict(dict1)
print(dict2)
dict2['FDU'] = '复旦大学'
dict2['RUC'] = 'Renmin University of China'
print(dict2)
```

```
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'RUC': 'Renmin University of China', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学', 'FDU': '复旦大学'}
```

[11]:
```
dict2 = dict(dict1)
print(dict2)
del dict2['RUC']
print(dict2)
```

```
{'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
```

词频统计-字典（dict）方法

- 词典方法词频统计：

In [74]:
```
import jieba
txt = open("C:\RUC\课程\PythonAI\课程课件\高瓴人工智能学院简介.txt", "r", encoding='utf-8').read()
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1: #排除单个字符的分词结果
        continue
    else:
        counts[word] = counts.get(word,0) + 1
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(15):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

```
人工智能        13
学院            8
一流            5
中国人民大学        4
未来            3
高瓴            3
院长            3
打造            3
```

```
人工智能        13
学院            8
一流            5
中国人民大学        4
未来            3
高瓴            3
院长            3
打造            3
全球            3
研究            3
联合            3
时代            2
影响            2
技术            2
发展            2
```

《人工智能与Python程序设计》—组合数据类型（三）

人工智能与Python程序设计 教研组

教学目标

- 理解字典类型的**原理**
  - 键和值

- 组合数据类型

人工智能与Python程序设计04-2
组合数据类型（三）

提纲

- **字典类型的实现原理***
- 组合数据类型的高级操作
- 组合数据类型相关模块

思考：什么信息可以作为字典的值和键

- 基本数据类型、组合数据类型、甚至是函数，均可以作为字典的值
  - 所有的对象（object）均可以作为字典的值

[65]:
```
dict2 = dict(dict1)
dict2['RUC'] = 1 # 基本数据类型
print(dict2)

dict2['RUC'] = ('中国人民大学', 'Renmin University of China') # 组合数据类型，元组
print(dict2)
dict2['RUC'] = ['中国人民大学', 'Renmin University of China'] # 组合数据类型，列表
print(dict2)

# 字典可以嵌套
dict2['RUC'] = {'中文名': '中国人民大学', '英文名': 'Renmin University of China'}
print(dict2)
print(dict2['RUC']['中文名'])
```

```
{'RUC': 1, 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'RUC': ('中国人民大学', 'Renmin University of China'), 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'RUC': ['中国人民大学', 'Renmin University of China'], 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
{'RUC': {'中文名': '中国人民大学', '英文名': 'Renmin University of China'}, 'THU': '清华大学', 'PKU': '北京大学', 'BIT': '北京理工大学', 'BUAA': '北京航空航天大学'}
中国人民大学
```

思考：什么信息可以作为字典的值和键

- 基本数据类型、组合数据类型、甚至是函数，均可以作为字典的值
  - 所有的对象（object）均可以作为字典的值

[64]:
```
# 字典的值还可以是一个函数
def output_ruc():
    print('中文名: 中国人民大学\t英文名: Renmin University of China')
dict2['RUC'] = output_ruc
print(dict2)
dict2['RUC']() # 在一个函数后加上括号就可以调用这个函数!
```

```
{'RUC': <function output_ruc at 0x7f1249ff8ca0>, 'THU': '清华大学', 'PKU': '北京大学', 'BI
T': '北京理工大学', 'BUAA': '北京航空航天大学'}
中文名: 中国人民大学        英文名: Renmin University of China
```

利用字典实现一个简单的计算器

- 函数可以作为字典的值
  - 利用字典将数据映射到相应的操作（函数）

[22]:
```
import math
oper_dict = {
    '+': lambda x, y: x + y,
    '-': lambda x, y: x - y,
    '*': lambda x, y: x * y,
    '/': lambda x, y: x / y,
    'sin': lambda x: math.sin(x),
    'exp': lambda x: math.exp(x),
    'ln': lambda x: math.log(x)
}

while(True):
    s = input('请输入一个前缀表达式，运算符和数字间用空格分开(输入空字符串退出)：')
    if len(s) == 0:
        break
    tokens = s.split(' ')
    operation = oper_dict[tokens[0]]
    operands = []
    for token in tokens[1:]:
        operands.append(float(token))
    print('计算结果:{0:.4f}'.format(operation(*operands)))
```

```
请输入一个前缀表达式，运算符和数字间用空格分开(输入空字符串退出)： + 1 2
计算结果:3.0000
请输入一个前缀表达式，运算符和数字间用空格分开(输入空字符串退出)： sin 3.14159
计算结果:0.0000
请输入一个前缀表达式，运算符和数字间用空格分开(输入空字符串退出)： exp 1
计算结果:2.7183
请输入一个前缀表达式，运算符和数字间用空格分开(输入空字符串退出)：
```

思考：什么信息可以作为字典的值和键

- 然而，并不是所有的对象均可以作为字典的键

[23]:
```
new_dict = {}
new_dict[1] = 1 # 可以用基本数据类型作为字典的键
new_dict['RUC'] = '中国人民大学' # 可以用字符串作为字典的键
new_dict[('RUC', 'GSAI')] = '中国人民大学高瓴人工智能学院' # 可以用元组作为字典的键
# 甚至可以用嵌套的元组作为字典的键
new_dict[(('RUC', 'GSAI'), 'address')] = '北京市海淀区中关村大街59号中国人民大学'
print(new_dict)
```

```
{1: 1, 'RUC': '中国人民大学', ('RUC', 'GSAI'): '中国人民大学高瓴人工智能学院', (('RUC', 'GSA
I'), 'address'): '北京市海淀区中关村大街59号中国人民大学'}
```

[24]:
```
new_dict[['THU', 'DCST']] = '清华大学计算机科学与技术系' # 然而使用列表作为字典的键会报错
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-91b83864798a> in <module>
----> 1 new_dict[['THU', 'DCST']] = '清华大学计算机科学与技术系' # 然而使用列表作为字典的键会报错

TypeError: unhashable type: 'list'
```

[25]:
```
new_dict[set(['THU', 'DCST'])] = '清华大学计算机科学与技术系' # 集合也不行
```

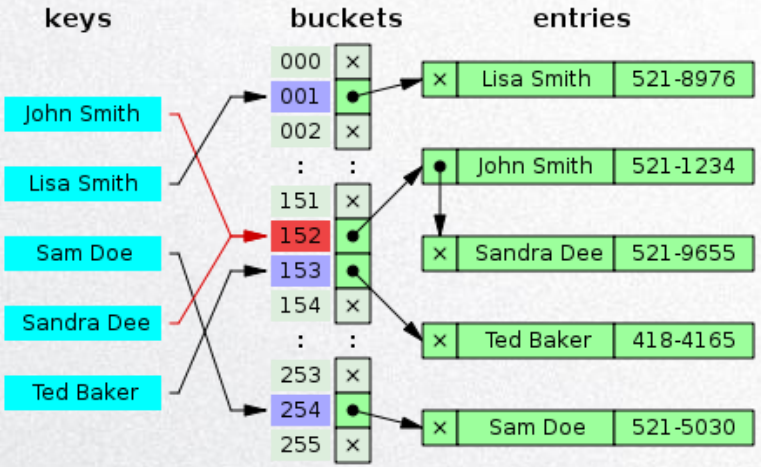
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-997e76ee367d> in <module>
----> 1 new_dict[set(['THU', 'DCST'])] = '清华大学计算机科学与技术系' # 集合也不行

TypeError: unhashable type: 'set'
```

思考：什么信息可以作为字典的值和键

- 那么，到底哪些对象可以作为字典的键呢？
  - 简单来说，所有一旦创建就不能被修改（immutable）的对象，都可以用作字典的键，例如：
    - 基本数据类型
    - 字符串（是的，字符串创建后就不能修改了）
    - 元组
  - 而创建后可以修改的（mutable）对象，都不能用作字典的键，例如：
    - 列表
    - 集合
    - 字典
  - 为什么？

思考：什么信息可以作为字典的值和键

- Python中的字典是基于哈希表（Hash table，又叫散列表）实现的
- 字典保存和索引元素（键值对）的原理：
  - 在将一个键值对添加到字典中时，会先调用键对象的哈希函数，计算哈希值
    - 哈希值通常为一个整数
  - 字典会将键的哈希值相同的元素保存在同一个“桶”（bucket）里
    - 一个桶里保存的元素的键的哈希值相等
  - 在索引时，会调用查询的键的哈希函数计算哈希值，找到相应的桶
    - 即找到所有与查询键哈希值相同的元素
    - 注意：不相等对象的哈希值可能相等（哈希冲突）
  - 再遍历桶中元素的键，调用__eq__()函数，判断其是否和查询相等
    - 相等：找到与查询相等的键，索引成功；
    - 均不相等：找不到查询对应的元素
- 思考：
  - 这样做是正确的的吗？
  - 这样做的目的是什么？



https://en.wikipedia.org/wiki/Hash_table

思考：什么信息可以作为字典的值和键

- 更准确的答案：
  - 所有可哈希（hash）的，即实现了__hash__()和__eq__()两个特殊方法的对象，可以作为字典的键
- Python语言要求：
  - 如果两个对象相等(a.__eq__(b) 或者 a == b返回True），那么他们的__hash__()函数返回值必须相等
  - 只有不可变对象才是可哈希的，才有__hash__()函数
    - 为什么?
- 哈希函数需要满足两个性质：
  - 哈希函数的计算应该比较高效
  - 如果两个对象不相等，那么他们的哈希值相等的可能性很低
    - 哈希值的分布尽可能是“均匀”的

人工智能与Python程序设计04-2
组合数据类型（三）

提纲

- 字典类型概念与操作
- **组合数据类型的高级操作**
- 组合数据类型相关模块

组合数据类型的高级操作

- Python语言自带的用于处理组合数据类型的函数
  - Filter
  - Map
  - Reduce
- 列表推导式
  - 方便的创建组合数据类型的方式

组合数据类型的高级操作

- filter函数：
  - 形式：filter(function, iterable)
  - 功能：将可遍历对象中不满足function定义条件的元素过滤掉

[67]:
```
# 保留所有偶数，过滤掉其他数
data = [1, 2, 3, 4, 5, 6]
print(list(filter(lambda x: x % 2 == 0, data)))
```

```
[2, 4, 6]
```

[69]:
```
# 分词后过滤掉所有单字词
import jieba
txt = open('./高瓴人工智能学院简介.txt', 'r', encoding='utf-8').read()
words = jieba.lcut(txt)[0:20]
print('过滤前:', words)
words = list(filter(lambda x: len(x) > 1, words))
print('过滤后:', words)
```

```
过滤前: ['“', '过去', '未', '去', '，', '未来', '已来', '”', '，', '在', '构建', '人工智能',
'时代', '的', '宏大', '世界观', '时', '，', '在', '影响']
过滤后: ['过去', '未来', '已来', '构建', '人工智能', '时代', '宏大', '世界观', '影响']
```

组合数据类型的高级操作

- map函数：
  - 形式：map(function, iterable, ...)
  - 功能： 将function函数应用于所有可遍历对象中的元素上

[80]:
```
# 求列表中每个数的平方
data = [1, 2, 3, 4, 5, 6]
results = list(map(lambda x: x * x, data))
print(results)
# 将每个数转化为字符串
results = map(str, results)
# 将字符串连接起来，用，分割
print(', '.join(results))
```

```
[1, 4, 9, 16, 25, 36]
1, 4, 9, 16, 25, 36
```

组合数据类型的高级操作

- map函数：
  - 形式：map(function, iterable, ...)
  - 功能： 将function函数应用于所有可遍历对象中的元素上
  - map还能同时作用于多个可遍历对象，例如：

[81]:
```
# map还能同时作用于多个可遍历对象，例如：
short_names = ['RUC', 'THU', 'PKU', 'BIT', 'BUAA']
full_names = ['中国人民大学', '清华大学', '北京大学', '北京理工大学', '北京航空航天大学']
results = map(lambda x, y: '{0:}=>{1:}'.format(x, y), short_names, full_names)
print(', '.join(results))
```

```
RUC=>中国人民大学, THU=>清华大学, PKU=>北京大学, BIT=>北京理工大学, BUAA=>北京航空航天大学
```

组合数据类型的高级操作

- reduce函数：
  - 形式：reduce(function, iterable[, initialzer])
  - 功能：利用function函数将可遍历对象中的元素合并起来

```
[85]: from functools import reduce # python 3 中需要从functools模块中import reduce函数

      # 将列表中的元素相乘
      data = [1, 2, 3, 4, 5, 6]
      print(reduce(lambda x, y: x * y, data))

      # 计算列表中的最大值
      data = [1, 2, 3, 4, 5, 6, 1, 10, -10]
      print(reduce(lambda x, y: x if x > y else y, data))

      720
      10
```

组合数据类型的高级操作

- 列表推导式
  - 方便的创建组合数据类型的方式
  - 其形式为：
    - <expression> for item in iterable <if optional_condition>
  - 上述语句会生成一个可遍历的对象，基于该对象，我们可以创建列表、元组、集合和字典，例如：

```
[93]: # 生成包含100以内所有3的倍数的列表和集合
      print([x for x in range(100) if x % 3 == 0])
      print({x for x in range(100) if x % 3 == 0})

      [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66,
      69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99]
      {0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66,
      69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99}

[95]: # 生成字典
      inverse_dict = {value: key for key, value in dict1.items()}
      print(inverse_dict)

      {'中国人民大学': 'RUC', '清华大学': 'THU', '北京大学': 'PKU', '北京理工大学': 'BIT', '北京航空航
      天大学': 'BUAA'}
```

人工智能与Python程序设计04-2
组合数据类型（三）

提纲

- 字典类型概念与操作
- 组合数据类型的高级操作
- **组合数据类型相关模块**

组合数据类型相关模块

- Python语言在collections模块中还提供了其他的组合数据类型：
  - deque
  - Counter
  - defaultdict

collections模块中提供的组合数据类型

- deque:
  - 可以在头部增删元素的列表

```
[43]: from collections import deque
      x = deque([1, 2, 3])
      x.append(4) # 在尾部加入新元素
      print(x)
      x.pop() # 返回并删除尾部元素
      print(x)
      x.appendleft(0) # 在头部加入新元素
      print(x)
      x.popleft() # 返回并删除头部元素
      print(x)

      deque([1, 2, 3, 4])
      deque([1, 2, 3])
      deque([0, 1, 2, 3])
      deque([1, 2, 3])
```

  - 思考：用下面这样的方法在列表（list）前增加元素会有什么问题？

```
[45]: # 思考：像这样在普通列表前增加元素会有什么问题?
      x = [1, 2, 3]
      x = [0] + x
      print(x)

      [0, 1, 2, 3]
```

collections模块中提供的组合数据类型

- deque:

```
[54]: %%time
      x = []
      for i in range(100000):
          x = [i * i] + x

      CPU times: user 9.58 s, sys: 0 ns, total: 9.58 s
      Wall time: 9.58 s

[55]: %%time
      x = deque()
      for i in range(100000):
          x.appendleft(i * i)

      CPU times: user 7.54 ms, sys: 11 µs, total: 7.55 ms
      Wall time: 7.42 ms
```

collections模块中提供的组合数据类型

- Counter
  - 专门用来计数的字典

```
[65]: from collections import Counter
      data = [1, 1, 1, 2, 2, 3]
      c = Counter(data) # 可以直接用需要计数的可遍历对象构造Counter
      print(c)
      print(c[1]) # 也可以像使用字典一样访问和修改每一个元素
      c[1] += 1
      print(c)
      print(c[0]) # 没出现过的元素的个数为0

      Counter({1: 3, 2: 2, 3: 1})
      3
      Counter({1: 4, 2: 2, 3: 1})
      0
```

  - 思考：可遍历对象中的元素需要满足什么要求？为什么？

collections模块中提供的组合数据类型

- defaultdict:
  - 有默认值的字典
  - 在构造时需提供一个函数对象
  - 在找不到某个查询键对应的元素时，调用上述函数对象
    - 返回函数的返回值
    - 并将查询键和函数的返回值构成一个新的键值对加入defaultdict

```
In [71]: from collections import defaultdict
d = defaultdict(lambda: '默认值') # 构造时需要一个生成默认值的函数
d.update(dict1)
print(d)
print(d['RUC']) # 若defaultdict包含该键，正常返回对应值
print(d['FDU']) # 若不包含该键，调用生成默认值函数，生成一个默认值，添加到defaultdict中
print(d)

defaultdict(<function <lambda> at 0x7f99a9368790>, {'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT':
'北京理工大学', 'BUAA': '北京航空航天大学'})
中国人民大学
默认值
defaultdict(<function <lambda> at 0x7f99a9368790>, {'RUC': '中国人民大学', 'THU': '清华大学', 'PKU': '北京大学', 'BIT':
'北京理工大学', 'BUAA': '北京航空航天大学', 'FDU': '默认值'})
```

collections模块中提供的组合数据类型

- defaultdict:
  - 例子：分词并记录词在文本中出现的位置

```
[81]: # 分词并记录每个词出现的位置
import jieba
txt = open('./高瓴人工智能学院简介.txt', 'r', encoding='utf-8').read()
positions = defaultdict(list) # 默认值是list函数返回的一个空列表
for word, start_pos, end_pos in jieba.tokenize(txt): # tokenize函数返回词，开始、结束位置
    if len(word) > 1:
        positions[word].append((start_pos, end_pos))
results = sorted(positions.items(), key=lambda x: -len(x[1]))
for word, positions in results[0:15]:
    print(word)
    print(positions)
```

```
人工智能
[(15, 19), (32, 36), (53, 57), (78, 82), (105, 109), (127, 131), (219, 223), (250, 254), (317, 321), (405, 409), (439, 443), (471, 475), (508, 512)]
学院
[(82, 84), (109, 111), (120, 122), (172, 174), (223, 225), (284, 286), (298, 300), (328, 330)]
一流
[(326, 328), (361, 363), (370, 372), (379, 381), (388, 390)]
中国人民大学
[(70, 76), (112, 118), (196, 202), (276, 282)]
未来
[(6, 8), (315, 317), (348, 350)]
高瓴
[(76, 78), (103, 105), (217, 219)]
院长
[(242, 244), (273, 275), (286, 288)]
打造
[(304, 306), (368, 370), (479, 481)]
全球
[(332, 334), (467, 469), (512, 514)]
研究
[(416, 418), (427, 429), (457, 459)]
联合
[(449, 451), (455, 457), (477, 479)]
时代
[(19, 21), (321, 323)]
影响
[(30, 32), (310, 312)]
技术
[(36, 38), (414, 416)]
发展
[(38, 40), (353, 355)]
```

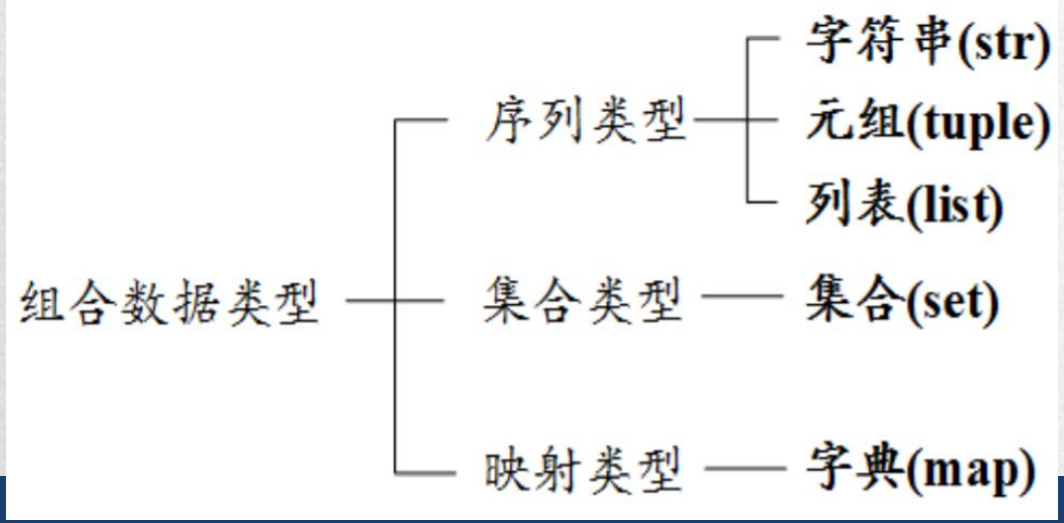
人工智能与Python程序设计04-2
组合数据类型（三）

提纲

- 字典类型概念与操作
- 组合数据类型的高级操作
- 组合数据类型相关模块

组合数据类型回顾和小结

- 计算机不仅对单个变量表示的数据进行处理，更多情况，计算机需要对一组数据进行批量处理。
- 组合数据类型能够将多个同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序更容易。
- 根据数据之间的关系，组合数据类型可以分为三类：序列类型、集合类型和映射类型。



组合数据类型回顾和小结

- 上述组合数据类型均为一个数据结构（data structure）：
  - 如何组织和存储数据
  - 对外提供了哪些接口
    - 创建、访问、修改、删除
    - 可变/不可变
- 编写程序时需要根据需求，选择合适的数据结构
  - 使用合适的组合数据类型
  - 简化程序设计过程
  - 提升程序运行效率
- Python语言自带了功能强大的组合数据类型
  - 学会使用这些组合数据类型
  - 《数据结构》课程会涉及及如何实现这些组合数据类型

小作业1：多个文本的词频统计

- 作业要求：
  - 按照章节读取《射雕英雄传》全文
    - 获取一个目录下所有文件，并依次打开、读取文件内容
  - 对文本进行分词
    - 需考虑使用自定义词典提升分词准确率
  - 对文本进行词频统计
    - 输出出现频率最高的词
    - 分析并思考：
      - 出现频率最高的词有什么特点?
      - 它们能否反映文本的内容?
      - 如何提取出能更好的体现文本内容的词?
  - 请在未来课堂提交

小作业2：统计人物出现次数

- 作业要求：
  - 按照章节读取《射雕英雄传》全文
  - 对文本进行分词
  - 读取射雕英雄传人物.txt中保存的人物名称
  - 将人物在章节中出现的次数输出到一个CSV文件中
    - 每行对应一个人物
    - 每列对应一个章节
    - 单元格中内容为人物名称在该章节文本中出现的次数
    - 注意：
      - 章节和人物应该按照一定顺序进行排序
      - 由于jieba库分词算法有一定的随机性，最后结果会有一定差异
  - 请在未来课堂提交

课后练习：《射雕英雄传》人物出场分析

- 作业要求：
  - 读取《射雕英雄传》全文，进行分词
    - 需使用用户自定义词典，提升分词准确率
  - 根据射雕英雄传人物.txt中保存的人物名称，找出每个人物出现的地方，并统计出现次数
    - 计算人物名称在文本出现次数
    - 只考虑该文件中的人物名称，不需考虑人物的别名、外号
  - 设计程序，使用字典等组合数据类型，统计两个人物一同出场的次数
    - 若人物B的名字出现在以人物A为中心，左右50个词的上下文环境内，则认为两个人物同时出现
  - 输出每个人物的出现次数，和与他共同出现次数最多的前十个人物及相应的共同出现次数
    - 例如，对于主角郭靖，输出：
    - 注意：jieba库得到的分词结果可能会有一定随机性，所以最终执行结果不一定会和该样例相同

```
郭靖：共出场3337次
        黄蓉：1202次
        洪七公：395次
        黄药师：298次
        周伯通：277次
        梅超风：190次
        拖雷：161次
        欧阳克：148次
        裘千仞：138次
        华筝：136次
        杨康：128次
```

课后练习：《射雕英雄传》人物出场分析

- 作业要求（选做）：
  - 实现一个能查找两个人物共同出现的文本片段的函数：
    - find_co_occurrence(p1, p2, k=0)
    - p2在以p1为中心的上下文中第k+1次出现的文本
    - 并用一定方式，在文本中标记出p1和p2
    - 例如：

```
In [49]: find_co_occurrence('郭靖', '黄蓉', 0)

Out[49]: '...情意，而他今日已不在人世，不能让我将这修订本的第一册书亲手送给他，再想到他那亲切的笑容和微带口吃的谈吐，心头甚感辛酸。 《射雕》中的人物个性单纯，*郭靖*诚朴厚重、[黄蓉]机智狡狯，读者容易印象深刻。这是中国传统小说和戏剧的特征，但不免缺乏人物内心世界的复杂性。大概由于人物性格单纯而情节热闹，所以《射雕》比较得到欢迎，曾拍过粤语电影，...'
```

```
In [50]: find_co_occurrence('郭靖', '黄蓉', 1)

Out[50]: '... 这次[黄蓉]领着他到了张家口最大的酒楼长庆楼，铺陈全是仿照大宋旧京汴梁大酒楼的格局。[黄蓉]不再大点酒菜，只要了四碟精致细点，一壶龙井，两人又天南地北的谈了起来。 [黄蓉]听*郭靖*说养了两头白雕，好生羡慕，说道：“我正不知到哪里去好，这么说，明儿我就上蒙古，也去捉两只小白雕玩玩。”*郭靖*道：“那可不容易碰上。”[黄蓉]道：“怎么...'
```

  - 提示：可以使用jieba.tokenize()函数得到每个词在原始字符串中的位置

谢谢！