《大数据分析导论》——组合数据类型

上次课回顾：函数与代码复用

- 函数
  - 函数定义、调用
  - 参数传递
  - 返回值
  - 函数对变量的作用
  - **递归**

```
2    def reverse(s):
3        if s == "" or len(s) == 1:
4            return s
5        else:
6            mStr = reverse(s[1:-1])
7            return s[-1] + mStr + s[0]
8
```

大数据分析导论
组合数据类型

提纲

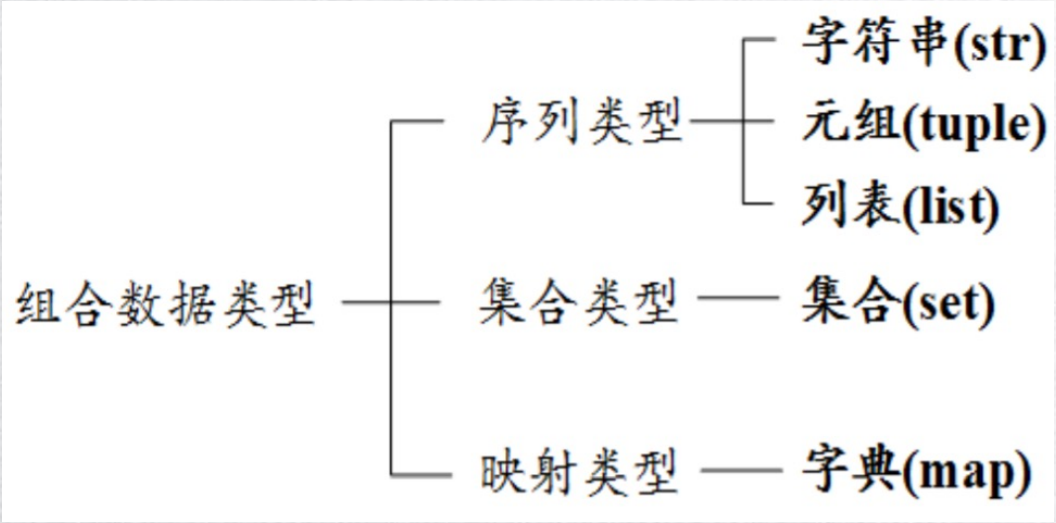
- 组合数据类型概述
- 列表
- 集合
- 字典

组合数据类型概述

- 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。一些例子包括：
  - 给定一组单词{python, data, function, list, loop}，计算并输出每个单词的长度；
  - 给定一个学院学生信息，统计一下男女生比例；
  - 一次实验产生了很多组数据，对这些大量数据进行分析；
- 基本数据类型仅能表示一个数据
- 实际计算中存在大量同时处理多个数据的情况，这需要将多个数据有效组织起来并统一表示，这种能够表示多个数据的类型称为组合数据类型

组合数据类型概述

- 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。

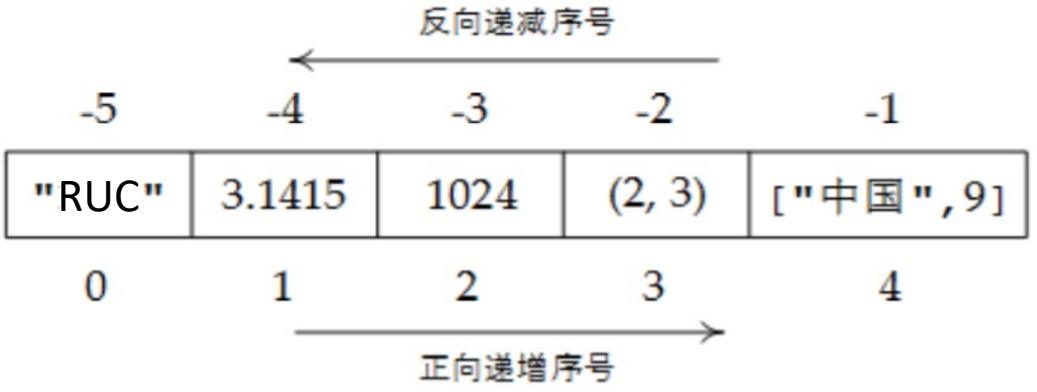


序列类型

- 序列类型是一维元素向量，元素之间存在先后关系，通过序号访问
- 当需要访问序列中某特定值时，只需要通过下标标出即可
- 由于元素之间存在顺序关系，所以序列中可以存在相同数值但位置不同的元素。序列类型支持成员关系操作符（in）、长度计算函数（len()）、分片（[]），元素本身也可以是序列类型。

序列类型

- Python语言中有很多数据类型都是序列类型，其中比较重要的是：str（字符串）、tuple（元组）、list（列表）
  - 元组是包含0个或多个数据项的不可变序列类型。元组生成后是固定的，其中任何数据项不能替换或删除。
  - 列表则是一个可以修改数据项的序列类型，使用也最灵活。
- 索引体系：

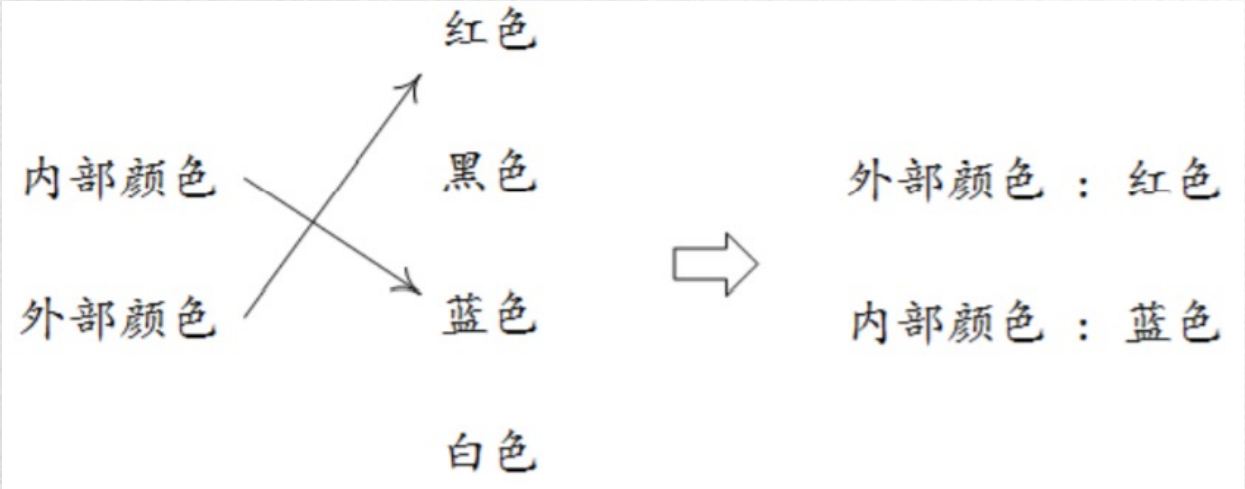


集合类型

- 集合类型与数学中集合的概念一致，即包含0个或多个数据项的无序组合。集合中元素不可重复，元素类型只能是固定数据类型，例如：整数、浮点数、字符串、元组等。列表、字典和集合类型本身都是可变数据类型，不能作为集合的元素出现。
- 由于集合是无序组合，它没有索引和位置的概念，不能分片，集合中元素可以动态增加或删除。集合用大括号（{}）表示，可以用赋值语句生成一个集合。

映射类型

- 映射类型是“键-值”数据项的组合，每个元素是一个键值对，即元素是(key, value)，元素之间是无序的。键值对(key, value)是一种二元关系。在Python中，映射类型主要以字典（dict）体现。



大数据分析导论
组合数据类型

提纲

- 组合数据类型概述
- **列表**
- 集合
- 字典

一个例子

- 读取三个数字，并计算平均数

```
num1 = float(raw_input())
num2 = float(raw_input())
num3 = float(raw_input())
avg = (num1 + num2 + num3) / 3
```

- 如果是30个数呢？

```
num1 = float(raw_input())
num2 = float(raw_input())
num3 = float(raw_input())
...

avg = (num1 + num2 + num3 + ...) / 30
```

列表（List）

- 内建（built-in）数据结构（data structure），用来存储一系列元素（items）
- 如：lst = [5.4, 'hello', 2]

lst	5.4	‘hello’	2
index	0	1	2
index	-3	-2	-1

  - lst[0] is 5.4
  - lst[3] ERROR!

  - lst[1:3] is ['hello', 2]

列表类型

- 列表（list）是包含0个或多个对象引用的有序序列，属于序列类型。与元组不同，列表的长度和内容都是可变的，可自由对列表中数据项进行增加、删除或替换。列表没有长度限制，元素类型可以不同，使用非常灵活。
- 由于列表属于序列类型，所以列表也支持成员关系操作符（in）、长度计算函数（len()）、分片（[]）。列表可以同时使用正向递增序号和反向递减序号，可以采用标准的比较操作符（<、<=、==、!=、>=、>）进行比较，列表的比较实际上是单个数据项的逐个比较。

回到第一个例子

- 读取5个数字，并计算平均数

```
nums = []
for i in range(5):
    nums.append(float(input()))
s = 0
for num in nums:
    s += num

avg = s / 30
print(avg)
```

```
1
2
5
4
6
0.6
```

- 内建 sum 函数
  - avg = sum(nums) / len(nums)
- 更多内建函数，如 max，min
  - http://docs.python.org/2/library/functions.html

列表类型

- 列表用中括号（[]）表示，也可以通过list()函数将元组或字符串转化成列表。直接使用list()函数会返回一个空列表。

```
In [35]: ls = ["信息楼", "RUC", [125, "AI"], 100872, "RUC"]
         ls
Out[35]: ['信息楼', 'RUC', [125, 'AI'], 100872, 'RUC']

In [36]: ls[2][-1][0]
Out[36]: 'A'

In [37]: list(("信息楼", "RUC", (12, "AI"), 100872, "RUC"))
Out[37]: ['信息楼', 'RUC', (12, 'AI'), 100872, 'RUC']

In [38]: list("中国人民大学高瓴人工智能学院")
Out[38]: ['中', '国', '人', '民', '大', '学', '高', '瓴', '人', '工', '智', '能', '学', '院']

In [39]: list()
Out[39]: []
```

列表类型

- 与整数和字符串不同，列表要处理一组数据，因此，列表必须通过显式的数据赋值才能生成，简单将一个列表赋值给另一个列表不会生成新的列表对象。

```
In  [40]: ls = ["信息楼", "RUC", [125, "AI"], 100872, "RUC"]
          lt = ls
          ls[2] = 0
          lt
Out[40]: ['信息楼', 'RUC', 0, 100872, 'RUC']
```

列表类型

- 列表类型的操作

函数或方法	描述
ls[i] = x	替换列表ls第i数据项为x
ls[i: j] = lt	用列表lt替换列表ls中第i到j项数据（不含第j项，下同）
ls[i: j: k] = lt	用列表lt替换列表ls中第i到j以k为步的数据
del ls[i: j]	删除列表ls第i到j项数据，等价于ls[i: j]=[]
del ls[i: j: k]	删除列表ls第i到j以k为步的数据
ls += lt或ls.extend(lt)	将列表lt元素增加到列表ls中
ls *= n	更新列表ls，其元素重复n次
ls.append(x)	在列表ls最后增加一个元素x
ls.clear()	删除ls中所有元素
ls.copy()	生成一个新列表，复制ls中所有元素
ls.insert(i, x)	在列表ls第i位置增加元素x
ls.pop(i)	将列表ls中第i项元素取出并删除该元素
ls.remove(x)	将列表中出现的第一个元素x删除
ls.reverse(x)	列表ls中元素反转

列表类型

- 当使用一个列表改变另一个列表值时，Python不要求两个列表长度一样，但遵循“多增少减”的原则，例子如下。

```
In  [43]: ls = list(range(6))
          ls
Out[43]: [0, 1, 2, 3, 4, 5]

In  [44]: len(ls[3:])
Out[44]: 3

In  [45]: 2 in ls
Out[45]: True

In  [46]: ls[3] = 'replace'
          ls
Out[46]: [0, 1, 2, 'replace', 4, 5]
```

```
In  [47]: ls[2:4] = ['machine','learning']
          ls
Out[47]: [0, 1, 'machine', 'learning', 4, 5]

In  [51]: ls[2:4] = ['machine','learning','AI']
          ls
Out[51]: [0, 1, 'machine', 'learning', 'AI', 5]

In  [52]: ls[2:4] = ['AI']
          ls
Out[52]: [0, 1, 'AI', 'AI', 5]
```

列表类型

- 与元组一样，列表可以通过for...in语句对其元素进行遍历，基本语法结构如下：

for <任意变量名> in <列表名>:
语句块

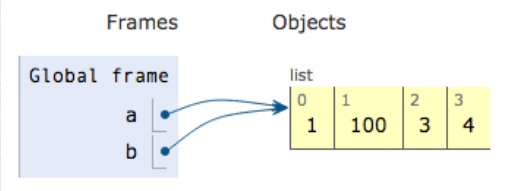
```
In [53]: for i in ls:
             print(i, end=" ")

0 1 AI AI 5
```

- 列表是一个十分灵活的数据结构，它具有处理任意长度、混合类型的能力，并提供了丰富的基础操作符和方法。当程序需要使用组合数据类型管理批量数据时，请尽量使用列表类型。

列表赋值

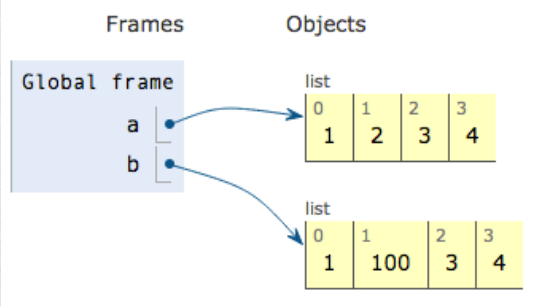
- 下列代码的执行结果是？

```
a = [1, 2, 3, 4]
b = a
b[1] = 100
print (a[1])
```



- 下面的呢？

```
a = [1, 2, 3, 4]
b = a[:]
b[1] = 100
print (a[1])
```



- 动态演示
  - http://www.pythontutor.com/

列表作函数参数

- 如交换列表中两个元素的函数

```
def swap(a, b):
    tmp = a
    a = b
    b = tmp

x = 10
y = 20
swap(x, y)
print(x, y)
```

```
10 20
```

VS.

```
def swap(lst, a, b):
    tmp = lst[a]
    lst[a] = lst[b]
    lst[b] = tmp

x = [10, 20, 30]
swap(x, 0, 1)
print(x)
```

```
[20, 10, 30]
```

单选题 1分

设置

列表ls=[[125, 12],[[7,2],2,7],[5,2],1]，len(ls)值是多少？

A 2

B 4

C 8

D 9

提交

单选题 1分

设置

列表ls1=[125,1]，ls2=ls1, ls1[0] = 2, ls2值是多少？

A [125, 1]

B [2,1]

提交

示例：基本统计值计算

- 以最简单的统计问题为例，求解一组不定长数据的基本统计值，即平均值、标准差、中位数。
- 一组数据$S=s_0, s_1, \ldots, s_{n-1}$，其算术平均值、标准差分别表示为：
- $$m=\frac{\sum_{i=0}^{n-1} s_i}{n}, \qquad d=\sqrt{\frac{\sum_{i=0}^{n-1}(s_i-m)^2}{n-1}}$$
- 由于平均数、标准差和中位数是三个不同的计算目标，使用函数方式编写计算程序。
  - getNum()函数从用户输入获得数据
  - mean()函数计算平均值
  - dev()函数计算标准差
  - median()函数计算中位数

示例：基本统计值计算

- 以最简单的统计问题为例，求解一组不定长数据的基本统

In [54]:

```
from math import sqrt

def getNum(): #获取用户输入
    nums = []
    iNumStr = input("请输入数字(直接输入回车退出): ")
    while iNumStr != "":
        nums.append(eval(iNumStr))
        iNumStr = input("请输入数字(直接输入回车退出): ")
    return nums

def mean(numbers): #计算平均值
    s = 0.0
    for num in numbers:
        s = s + num
    return s / len(numbers)

def dev(numbers, mean): #计算方差
    sdev = 0.0
    for num in numbers:
        sdev = sdev + (num - mean)**2
    return sqrt(sdev / (len(numbers)-1))
```

```
def median(numbers): #计算中位数
    sorted(numbers)
    size = len(numbers)
    if size % 2 == 0:
        med = (numbers[size//2-1] + numbers[size//2])/2
    else:
        med = numbers[size//2]
    return med

n = getNum() #主体函数
m = mean(n)
print("平均值:{},方差:{:.2},中位数:{}.".format(m,\
 dev(n,m),median(n)))
```

```
请输入数字(直接输入回车退出): 12.2
请输入数字(直接输入回车退出): 11.8
请输入数字(直接输入回车退出): 10.9
请输入数字(直接输入回车退出): 11.7
请输入数字(直接输入回车退出): 13.1
请输入数字(直接输入回车退出): 12.7
请输入数字(直接输入回车退出):
平均值:12.066666666666665,方差:0.78,中位数:11.3.
```

示例：查找

- 在列表中查找一个值，并返回该值第一次出现的位置；如果该值不存在，则返回 -1

```
def search(lst, x):
    for i in range(len(lst)):
        if lst[i] == x:
            return i
    return -1

search([1, 2, 2], 2)
```

```
1
```

- list.index( ) 方法
  - [1, 2, 2].index(2)
- 线性查找 O(n)

```
[1, 2, 2].index(2)
```

```
1
```

内建排序函数

- sorted() 函数

```
a = [5, 2, 3, 1, 4]
sorted(a)
```

```
[1, 2, 3, 4, 5]
```

- list.sort() 方法

```
a = [5, 2, 3, 1, 4]
a.sort()
a
```

```
[1, 2, 3, 4, 5]
```

- 算法：quicksort
  - 时间复杂度：$O(n\log n)$

列表类型

- 列表是一个动态长度的数据结构，可以根据需求增加或减少元素；
- 列表的一系列方法或操作符为计算提供了简单的元素运算手段；
- 列表提供了对每个元素的简单访问方式以及所有元素的遍历方式。

元组

- 元组（tuple）是序列类型中比较特殊的类型，因为它一旦创建就不能被修改。元组类型在表达固定数据项、函数多返回值、多变量同步赋值、循环遍历等情况下十分有用。
- Python中元组采用逗号和圆括号（可选）来表示。

```
In [1]: creature = "cat", "dog", "tiger", "human"
        creature
Out[1]: ('cat', 'dog', 'tiger', 'human')

In [2]: color = ("red", 0x001100, "blue", creature)
        color
Out[2]: ('red', 4352, 'blue', ('cat', 'dog', 'tiger', 'human'))

In [3]: color[2]
Out[3]: 'blue'

In [4]: color[-1][2]
Out[4]: 'tiger'
```

```
In [5]: def func(x): #函数多返回值
            return x, x**3
        a, b = 'dog', 'tiger' #多变量同步赋值
        a, b = (b, a) #多变量同步赋值，括号可省略
        print(a)

        tiger

In [6]: import math
        for x, y in ((1,0), (2,5), (3,8)): #循环遍历
            print(math.hypot(x,y)) #求多个坐标值到原点的距离

        1.0
        5.385164807134505
        8.54400374531753
```

序列类型

- 序列类型通用操作符和函数

操作符	描述
x in s	如果x是s的元素，返回True，否则返回False
x not in s	如果x不是s的元素，返回True，否则返回False
s + t	连接s和t
s * n 或 n * s	将序列s复制n次
s[i]	索引，返回序列的第i个元素
s[i: j]	分片，返回包含序列s第i到j个元素的子序列（不包含第j个元素）
s[i: j: k]	步骤分片，返回包含序列s第i到j个元素以j为步数的子序列
len(s)	序列s的元素个数（长度）
min(s)	序列s中的最小元素
max(s)	序列s中的最大元素
s.index(x[, i[, j]])	序列s中从i开始到j位置中第一次出现元素x的位置
s.count(x)	序列s中出现x的总次数

大数据分析导论
组合数据类型

提纲

- 组合数据类型概述
- 列表
- **集合**
- 字典

集合类型

- 由于集合元素是无序的，集合的打印效果与定义顺序可以不一致。由于集合元素独一无二，使用集合类型能够过滤掉重复元素。set(x)函数可以用于生成集合。

```
In  [7]: S = {"信息楼", "RUC", (12, "AI"), 100872}
         S
Out[7]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In  [8]: S = {"信息楼", "RUC", (12, "AI"), 100872, "RUC"}
         S
Out[8]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In  [10]: T = set('intelligence')
          T
Out[10]: {'c', 'e', 'g', 'i', 'l', 'n', 't'}

In  [11]: T = set(("python", "learning", "artificial", "intelligence"))
          T
Out[11]: {'artificial', 'intelligence', 'learning', 'python'}
```

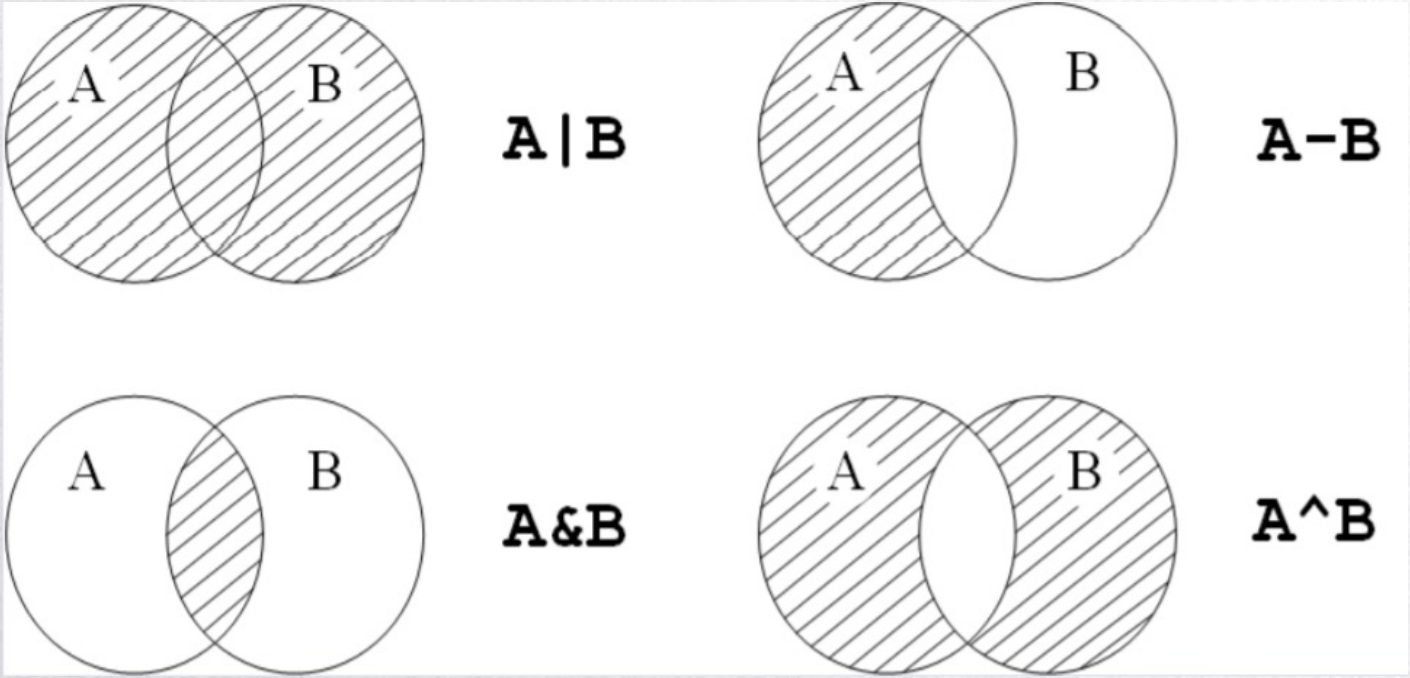
集合类型

- 集合操作符

操作符	描述
S – T 或 S.difference(T)	返回一个新集合，包括在集合S中但不在集合T中的元素
S-=T或S.difference_update(T)	更新集合S，包括在集合S中但不在集合T中的元素
S & T或S.intersection(T)	返回一个新集合，包括同时在集合S和T中的元素
S&=T或S.intersection_update(T)	更新集合S，包括同时在集合S和T中的元素。
S^T或s.symmetric_difference(T)	返回一个新集合，包括集合S和T中元素，但不包括同时在其中的元素
S=^T或s.symmetric_difference_update(T)	更新集合S，包括集合S和T中元素，但不包括同时在其中的元素
S|T或S.union(T)	返回一个新集合，包括集合S和T中所有元素
S=|T或S.update(T)	更新集合S，包括集合S和T中所有元素
S<=T或S.issubset(T)	如果S与T相同或S是T的子集，返回True，否则返回False，可以用S<T判断S是否是T的真子集
S>=T或S.issuperset(T)	如果S与T相同或S是T的超集，返回True，否则返回False，可以用S>T判断S是否是T的真超集

集合类型

- 上述操作符表达了集合类型的4种基本操作，交集（&）、并集（|）、差集（-）、补集（^），操作逻辑与数学定义相同



集合类型

- 集合类型有10个操作函数或方法

函数或方法	描述
S.add(x)	如果数据项x不在集合S中，将x增加到s
S.clear()	移除S中所有数据项
S.copy()	返回集合S的一个拷贝
S.pop()	随机返回集合S中的一个元素，如果S为空，产生KeyError异常
S.discard(x)	如果x在集合S中，移除该元素；如果x不在，不报错
S.remove(x)	如果x在集合S中，移除该元素；不在产生KeyError异常
S.isdisjoint(T)	如果集合S与T没有相同元素，返回True
len(S)	返回集合S元素个数
x in S	如果x是S的元素，返回True，否则返回False
x not in S	如果x不是S的元素，返回True，否则返回False

集合类型

- 集合类型主要用于三个场景：成员关系测试、元素去重和删除数据项。
- 集合类型与其他类型最大的不同在于它不包含重复元素，因此，当需要对一维数据进行去重或进行数据重复处理时，一般通过集合来完成。

```
In [12]: S = {"信息楼", "RUC", (12, "AI"), 100872}
         "RUC" in S   #成员关系测试
Out[12]: True

In [13]: tup = (125, "信息楼", "RUC", (12, "AI"), 100872, 125)
         set(tup)  #元素去重
Out[13]: {(12, 'AI'), 100872, 125, 'RUC', '信息楼'}

In [29]: S = set(tup)-{125}
         S
Out[29]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In [26]: newtup = tuple(S)  # 去重同时删除数据项
         newtup
Out[26]: (100872, '信息楼', (12, 'AI'), 'RUC')

In [33]: S = set(tup)-{'信息楼'}
         S
Out[33]: {(12, 'AI'), 100872, 125, 'RUC'}
```

大数据分析导论
组合数据类型

提纲

- 组合数据类型概述
- 列表
- 集合
- **字典**

什么是字典（Dictionary）？

- 一系列“键-值（key-value）”对
- 通过“键”查找对应的“值”
- 类似纸质字典，通过单词索引表找到其相应的定义
  - C++: map、Java: HashTable or HashMap
- 例如：电话本

姓名（键）	电话号码（值）
John	86411234
Bob	86419453
Mike	86412387
……	……

字典的使用

- 创建字典
  - 使用 { } 创建字典
  - 使用 : 指明 键:值 对
    - my_dict = {'John': 86411234, 'Bob': 86419453,'Mike': 86412387}
  - 键必须是不可变的且不重复，值可以是任意类型
- 访问字典
  - 使用 [ ] 运算符，键作为索引
    - print(my_dict['Bob'])
    - print(my_dict['Tom'] #WRONG!
    - 增加一个新的对
      - my_dict['Tom'] = 86417639

字典运算符和方法

- len(my_dict)
  - 字典中键-值对的数量
- key in my_dict
  - 快速判断 key 是否为字典中的键：O(1)
  - 等价于 my_dict.has_key(key)
- for key in my_dict:
  - 枚举字典中的键，注：键是无序的
- 更多的方法
  - my_dict.items() – 全部的键-值对
  - my_dict.keys() – 全部的键
  - my_dict.values() – 全部的值
  - my_dict.clear() – 清空字典

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'}
```

字典类型的操作

- 获取一个字典中的所有键、值、以及键值对：
  - <d>.keys(), <d>.values(), <d>.items()

```
[13]: dict2 = dict(dict1)
      keys = dict2.keys()
      print(keys)
      values = dict2.values()
      print(values)
      items = dict2.items()
      print(items)

      dict_keys(['RUC', 'THU', 'PKU', 'BIT', 'BUAA'])
      dict_values(['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'])
      dict_items([('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')])
```

字典类型的操作

- 获取一个字典中的所有键、值、以及键值对：
  - <d>.keys(), <d>.values(), <d>.items()
  - 需要注意的是，上述方法获得均为字典中相应信息的一个视图（view），即如果我们修改了字典中的内容，上述视图对象也会相应变化

```
[14]: del dict2['RUC']
      print(keys)
      print(values)
      print(items)

      dict_keys(['THU', 'PKU', 'BIT', 'BUAA'])
      dict_values(['清华大学', '北京大学', '北京理工大学', '北京航空航天大学'])
      dict_items([('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')])
```

字典类型的操作

- 使用for … in … 语句遍历字典

```
[15]: dict2 = dict(dict1)
      for key in dict2: # 该语句会遍历字典中所有键
          print(key)

      for key, value in dict2.items(): # 这样可以遍历所有键值对
          print(key, value)
```

```
RUC
THU
PKU
BIT
BUAA
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
```

字典类型的操作

- 其它一些字典类型的相关操作：

[32]:
```
dict2 = dict(dict1)
print(len(dict2)) # 字典的大小（包含键值对的个数）

dict2.clear() # 清空字典
print(dict2)

dict2 = dict1.copy() # 返回一个字典的拷贝
print(dict2)

dict2.update({'RUC': 'Renmin University of China',
              'THU': 'Tsinghua University'}) # 批量更新字典中键值对
print(dict2)

print(dict2.pop('PKU')) # 按键查找，然后删除相应键值对
print(dict2)
```

```
5
{}
{'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': 'Tsinghua University', 'PKU': '北京大学', 'B
IT': '北京理工大学', 'BUAA': '北京航空航天大学'}
北京大学
{'RUC': 'Renmin University of China', 'THU': 'Tsinghua University', 'BIT': '北京理工大
学', 'BUAA': '北京航空航天大学'}
```

示例：字母计数

- 读取一个字符串，计算每个字母出现的个数
- 方案一
  - 生成 26 个变量，代表每个字母出现的个数
- 方案二
  - 生成具有 26 个元素的列表，将每个字母转化为相应的索引值，注意：ord(a) = 97，如 a ➔ 0，b ➔ 1，…

```
count = [0] * 26
for i in 'abcdad':
    count[ord(i) - 97] += 1
print(count)
```

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

- 方案三
  - 生成一个字典，字母做键，对应出现的次数做值

```
count = {}
for i in 'abcdad':
    if i in count:
        count[i] += 1
    else:
        count[i] = 1

print(count)
```

```
{'a': 2, 'b': 1, 'c': 1, 'd': 2}
```

示例：单词计数

- 读取小说"emma.txt"，打印前 10 个最常见单词
  - 是否还能直观的将每个单词转化为相应的数字？

生成词典

排序，一般需要先导出到列表中，对列表进行排序

```
f = open('emma.txt')
word_freq = {}
for line in f:
    words = line.split()
    for word in words:
        if word in word_freq:
            word_freq[word] += 1
        else:
            word_freq[word] = 1
f.close()
word_freq_lst = []
for word, freq in word_freq.items():
    word_freq_lst.append((freq, word))

word_freq_lst.sort(reverse = True)

for freq, word in word_freq_lst[:10]:
    print(word, freq)
```

```
emma 7
tom 4
book 4
word 2
this 2
note 2
jupyter 2
is 2
another 2
machine 1
```

作业1：单词计数和查找

- （1）给定一段英文文本（自行到网上拷贝或者用附件的big.txt）
- （2）统计其中每一个单词出现的次数（假设单词间都用空格或者标点符号分开）
- （3）任给一个单词，可以快速查找到其在这一段文本中出现的次数

作业2：用组合数据类型重写单词变换

- 任给一个英语单词，例如“university”，找出所有与这个单词“差一”的字符串，“差一”定义如下：
  - 删除：删除掉该单词中的任一字符；
  - 插入：在该单词的任意位置（包括头尾）插入一个英文字符；
  - 替换：将该单词中任意一个字符替换为另一个英文字符；
  - 交换：将该单词中相邻两个字符互换。

注意：

1. 输入的单词转换为小写格式；
2. 26个英文字符'abcdefghijklmnopqrstuvwxyz'

谢谢！