對外經濟貿易大學
University of International Business and Economics
信息学院
School of Information Technology & Management

第二章 序列

任课教师：袁石
办公地点：求索楼1006
电子邮箱：ystone1025@uibe.edu.cn

知识回顾

- Python语言基础：注释、缩进、数据类型、输入输出
- 运算符：成员运算
- 选择结构：if
- 循环结构：for、while

主要内容

- 1、序列概述
- 2、列表
- 3、元组
- 4、字典
- 5、集合

1、序列概述

1/1 序列的定义

序列是一块用于存放多个值的连续内存空间，并且按一定顺序排列

- 每个值（称为元素）都分配一个数字，称为索引或位置
- 通过该索引可以取出相应的值

门牌号码

部门牌子

1/2 序列的操作

- 序列的通用操作包括：索引，切片，相加，相乘，检查元素是否是序列成员，计算序列的长度、最大值和最小值

序列的操作	列表	元组	字典	集合	字符串
索引	√	√	√	×	√
切片	√	√	×	×	√
相加	√	√	×	×	√
乘法	√	√	×	×	√
检查元素是否是序列成员	√	√	√	√	√
序列计算（长度、最大值和最小值）	√	√	√	√	√

1/2 序列的操作

索引

- 序列中的每个元素都有一个编号，也称为索引
- 索引从0开始递增，即下标为0表示第一个元素
- 字典的索引是键，键是定制的，不是连续的、从0开始的编号

序列	元素1	元素2	元素3	元素...	元素n	
索引（下标）	0	1	2	...	n-1	从左向右
索引（下标）	-(n-1)	-(n-2)	-(n-3)	...	-1	从右向左

索引可以是负数，表示索引从右向左计数，即最后一个元素的索引值是-1

```
#索引
course = ['数据结构','数据库','Python与大数据分析','Access数据库']

print(course[1])
print(course[-1])
```

```
数据库
Access数据库
```

1/2 序列的操作

切片

- 切片可以访问一定范围内的元素，通过切片操作可以生成一个新的序列

序列名[开始索引：结束索引：切片步长]

- 开始索引：表示切片的开始位置（包括该位置），如果不指定，则默认为0
- 结束索引：表示切片的截止位置（不包括该位置），如果不指定，则默认为序列的长度
- 切片步长：表示遍历序列元素的步长，如果省略，则默认为1

```
#切片
poetry = ['青青园中葵','朝露待日晞','阳春布德泽','万物生光辉','常恐秋节至',
          '焜黄华叶衰','百川东到海','何时复西归','少壮不努力','老大徒伤悲']

print(poetry[1:6]) #获取第2到第6句诗
print(poetry[1:6:2]) #获取第2、4、6句诗
```

```
['朝露待日晞', '阳春布德泽', '万物生光辉', '常恐秋节至', '焜黄华叶衰']
['朝露待日晞', '万物生光辉', '焜黄华叶衰']
```

1/2 序列的操作

相加

- 序列相加（+运算符）可实现两种相同类型的序列相加操作
- 即两个序列进行连接，合并成一个序列，但不去除重复的元素
- 相同类型：同为列表、元组、字符串等，但序列中的元素可以不同

```
#相加
num = [3,6,9,12,15,18]
course_data = ['数据结构','数据库','Python与大数据分析','Access数据库']
course_information = ['管理学原理','数据结构','微观经济学','数据库']

print(num+course)
print(course_data+course_information) #序列相加不会去掉重复元素
```

```
[3, 6, 9, 12, 15, 18, '数据结构', '数据库', 'Python与大数据分析', 'Access数据
库']
['数据结构', '数据库', 'Python与大数据分析', 'Access数据库', '管理学原理', '数据
结构', '微观经济学', '数据库']
```

1/2 序列的操作

乘法

- 序列与数字n相乘，即可生成新的序列
- 新序列的内容为原来序列被重复n次的结果
- 乘法可实现初始化制定长度列表的功能

```
#乘法
job = ['医生','警察','教师']
print(job*3) #序列重复三次

init_a = [1]*10 #序列初始化
print(init_a)
```

```
['医生', '警察', '教师', '医生', '警察', '教师', '医生', '警察', '教师']
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

1/2 序列的操作

检查元素是否是序列成员

- 检查某个元素是否是序列成员：关键字in

value in sequence —not取反→ **value not in sequence**

- value：表示要检查的元素
- sequence：表示指定的序列
- 该语句返回“True”或者“False”： “True”表示元素在序列中， “False”表示元素不在序列中

```
#检查元素是否是序列成员
course = ['数据结构','数据库','Python与大数据分析','Access数据库']

print('管理学原理' in course) #关键字 in
print('数据库' not in course) #关键字 not in
```

```
False
False
```

1/2 序列的操作

序列计算

- 计算序列长度，即返回序列包含多少个元素：len()
- 返回序列中最大的元素：max()
- 返回序列中最小的元素：min()

```
#序列计算
num = [3, 6, 9, 12, 15, 18]

print('序列长度：',len(num))
print('序列中的最大值：',max(num))
print('序列中的最小值：',min(num))
```

```
序列长度：  6
序列中的最大值：  18
序列中的最小值：  3
```

函数	说明
list()	将序列转换为列表
str()	将序列转换为字符串
sum()	计算序列中元素的和
sorted()	对元素进行排序
reversed()	对序列元素进行反序排列
enumerate()	将序列组合为一个索引序列，多用在for循环中

2、列表

2/1 列表的定义

车次	出发站 到达站	出发时间 到达时间	历时	商务座 特等座	一等座	二等座 二等包座
G5 复	始 北京南 终 上海	07:00 11:40	04:40 当日到达	10	有	有
G105 复	始 北京南 终 上海虹桥	07:20 13:08	05:48 当日到达	12	有	有
G143	始 北京南 终 上海虹桥	07:50 13:12	05:22 当日到达	17	有	有
G107	始 北京南 终 上海虹桥	08:05 13:46	05:41 当日到达	8	有	有
G113 复	始 北京南 终 上海虹桥	08:50 14:33	05:43 当日到达	3	3	有
G1 复	始 北京南 终 上海虹桥	09:00 13:28	04:28 当日到达	11	有	有
G41 复	始 北京南 过 上海虹桥	09:12 14:49	05:37 当日到达	8	有	有
G115 复	始 北京南 终 上海虹桥	09:20 14:58	05:38 当日到达	4	候补	有
G7 复	始 北京南 终 上海虹桥	10:00 14:28	04:28 当日到达	2	有	有
G119	始 北京南 终 上海虹桥	10:05 15:51	05:46 当日到达	8	有	有
G121 复	始 北京南 终 上海虹桥	10:20 16:25	06:05 当日到达	3	候补	有
G125 复	始 北京南 终 上海虹桥	11:10 16:59	05:49 当日到达	7	有	有
G127 复	始 北京南 终 上海虹桥	11:30 17:20	05:50 当日到达	4	5	有
1461	始 北京 终 上海	11:55 07:00	19:05 次日到达	--	--	--
G9 复	始 北京南 终 上海虹桥	12:00 16:36	04:36 当日到达	候补	有	有
G129	始 北京南 终 上海	12:10 18:07	05:57 当日到达	9	有	有

课程号	课程名	课序号	开课单位	学分	上课老师	上课时间	上课地点	课容量	课余量	选课说明
CMP245	Python与大数据分析	1	信息学院	2	刘宁宁	1-16周 51	知403	40	0	含实验;
CMP245	Python与大数据分析	2	信息学院	2	刘宁宁	1-16周 41	知403	40	0	含实验;
CMP245	Python与大数据分析	3	信息学院	2	刘宁宁	1-16周 53	知409	40	0	含实验;
CMP245	Python与大数据分析	4	信息学院	2	李兵	1-16周 54	知407	40	0	含实验;
CMP245	Python与大数据分析	5	信息学院	2	李兵	1-16周 21	知404	40	0	含实验;
CMP245	Python与大数据分析	6	信息学院	2	雷擘	1-16周 24	知502	40	0	含实验;
CMP245	Python与大数据分析	7	信息学院	2	袁石	1-16周 55	知410	40	0	含实验;

- 列表：由一系列按特定顺序排列的元素组成
- 列表是Python内置的可变序列
- 列表的所有元素都放在一对中括号”[]”中，相邻两个元素使用逗号”,”分割
- 可将整数、实数、字符串、列表、元组等任何类型的内容放入列表中
- 同一个列表中的元素的类型可以不同

```
#列表示例
num = [3, 6, 9, 12, 15, 18] #整型列表
course = ['数据结构','数据库','Python与大数据分析','Access数据库'] #字符串型列表
hybrid = [3,'数据结构',[6,9],'数据库'] #混合列表：整数、字符串、列表
```

2/2 列表的创建、删除与访问

1、使用赋值运算符创建列表

```
#列表示例
num = [3,6,9,12,15,18] #整型列表
course = ['数据结构','数据库','Python与大数据分析','Access数据库'] #字符串型列表
hybrid = [3,'数据结构',[6,9],'数据库'] #混合列表：整数、字符串、列表
```

2、创建空列表和数值列表

```
#创建空列表和数值列表

emptylist = [] #空列表
numlist = list(range(10,20,2)) #range函数生成一组数字，再用list()将数字转化为列表
```

3、删除列表

```
#删除列表

course = ['数据结构','数据库','Python与大数据分析','Access数据库'] #字符串型列表
del course #Python自带垃圾回收机制，即自动销毁不用的列表
```

4、访问列表元素

通过索引下标的方式访问列表中对应位置的元素：

course[2]

```
#访问列表元素
import datetime                                    # 导入日期时间类
# 定义一个列表
mot = ["坚持下去不是因为我很坚强，而是因为我别无选择",
       "含泪播种的人一定能笑着收获",
       "做对的事情比把事情做对重要",
       "命运给予我们的不是失望之酒，而是机会之杯",
       "明日永远新鲜如初，纤尘不染",
       "求知若饥，虚心若愚",
       "成功将属于那些从不说“不可能”的人"]
day = datetime.datetime.now().weekday()            # 获取当前星期
print(mot[day])                                    # 输出每日一帖
```

求知若饥，虚心若愚

2/3 遍历列表

1、使用for循环

for循环直接获取列表元素

```
for item in listname:
    print(item)
```

2、使用for循环和enumerate()函数

for循环和enumerate()函数可以同时获取索引值和元素内容

```
for index,item in enumerate(listname):
    print(index,item)
```

例：每两句一行输出《长歌行》

思路：根据句子位置下标进行判定

- ✓ 下标为偶数，则不换行
- ✓ 下标为奇数，则换行

```
#每两句一行输出《长歌行》
print("          长歌行")
verse = ["青青园中葵","朝露待日晞","阳春布德泽","万物生光辉","常恐秋节至","焜黄华叶衰",
         "百川东到海","何时复西归","少壮不努力","老大徒伤悲"]
for index,item in enumerate(verse):
    if index%2 == 0:                          # 判断是否为偶数，为偶数时不换行
        print(item+", ", end='')
    else:
        print(item+"。")                       # 换行输出
```

```
          长歌行
青青园中葵，朝露待日晞。
阳春布德泽，万物生光辉。
常恐秋节至，焜黄华叶衰。
百川东到海，何时复西归。
少壮不努力，老大徒伤悲。
```

2/4 添加、修改和删除列表元素

1、添加一个元素：append()

listname.append(obj)

append表示将obj对象添加到列表listname末尾

2、添加一个列表：extend()

listname.extend(seq)

extend表示将列表seq的内容全部添加到列表listname末尾

3、修改元素

通过索引获取要修改的元素，然后再为其重新赋值即可

Listname[2] = new value

4、删除元素

1）根据索引删除

```
#根据索引删除元素
course = ['数据结构','数据库','Python与大数据分析','Access数据库']
del course[-1]
print(course)
```

```
['数据结构', '数据库', 'Python与大数据分析']
```

2）根据元素删除

```
#根据元素删除元素
course = ['数据结构','数据库','Python与大数据分析','Access数据库']
course.remove('数据结构')
print(course)
```

```
['数据库', 'Python与大数据分析', 'Access数据库']
```

2/5 列表排序

1、列表对象的sort()方法

listname.sort(key=None, reverse=False)

- listname：表示要进行排序的列表
- key：表示指定从每个列表元素中提取一个比较键。例如：key=str.lower，表示在排序时不区分字母大小写
- reverse：True表示降序排列，False表示升序排列。默认是升序排列

```
#排序
char = ['cat','Tom','Angela','pet']
char.sort()#默认：排序时区分字母大小写
print('区分字母大小写：',char)

char.sort(key=str.lower)#排序时不区分字母大小写
print('不区分字母大小写：',char)
```

```
区分字母大小写： ['Angela', 'Tom', 'cat', 'pet']
不区分字母大小写： ['Angela', 'cat', 'pet', 'Tom']
```

2、内置的sorted()函数

sorted(listname, key=None, reverse=False)

sort()和sorted()的区别：

- ✓ **sort()：改变原列表的元素排列顺序**
- ✓ **sorted()：不改变原列表的顺序，会建立一个原列表的副本，表示排序后的列表**

```
#sort与sorted的区别
price = [1200,5330,2988,6200,1998,8888]
price_new = sorted(price)
print('未排序的原列表：',price)
print('使用sorted之后的原列表：',price)#原列表顺序不变
price.sort()
print('使用sort之后的原列表：',price)#原列表顺序改变
```

```
未排序的原列表： [1200, 5330, 2988, 6200, 1998, 8888]
使用sorted之后的原列表： [1200, 5330, 2988, 6200, 1998, 8888]
使用sort之后的原列表： [1200, 1998, 2988, 5330, 6200, 8888]
```

2/6 成员运算

成员运算符：in、not in

```
#成员运算
a = [1, 2, 3, 4, 5]

print(2 in a)
print(6 in a)
```

```
True
False
```

3、元组

3/1 元组的定义

元组与列表的定义比较

	元组	列表
相同点	✓ 由一系列**按特定顺序排列**的元素组成 ✓ 相邻两个元素使用**逗号”,”**分割 ✓ 可将整数、实数、字符串、列表、元组等任何类型的内容放入元组/列表中 ✓ 同一个元组/列表中的元素的类型可以不同	
不同点	✓ 元组是Python内置的**不可变**序列 ✓ 元组的所有元素都放在一对**小括号”()”**中	✓ 列表是Python内置的**可变**序列 ✓ 列表的所有元素都放在一对**中括号”[]”**中

3/2 元组的创建、删除与访问

1、使用赋值运算符创建元组

```
#元组示例
num = (3,6,9,12,15,18) #整型元组
course = ('数据结构','数据库','Python与大数据分析','Access数据库') #字符串型元组
hybrid = (3,'数据结构',[6,9],'数据库') #混合元组：整数、字符串、列表
```

2、创建空元组和数值元组

```
#创建空元组和数值元组

emptytuple = () #空元组
numtuple = tuple(range(10,20,2)) #range函数生成一组数字，再用tuple()将数字转化为元组
```

3、删除元组

```
#删除元组

course = ('数据结构','数据库','Python与大数据分析','Access数据库') #字符串型元组
del course #Python自带垃圾回收机制，即自动销毁不用的元组
```

4、访问元组元素

通过索引下标的方式访问元组中对应位置的元素：

course[2]

- ✓ **元组是用小括号表示**
- ✓ **元组的关键字是tuple()**

例：使用元组实现每两句一行输出《长歌行》

```
#使用元组实现每两句一行输出《长歌行》
print("        长歌行")
verse = ("青青园中葵","朝露待日晞","阳春布德泽","万物生光辉","常恐秋节至","焜黄华叶衰",
         "百川东到海","何时复西归","少壮不努力","老大徒伤悲")
for index,item in enumerate(verse):
    if index%2 == 0:                    # 判断是否为偶数，为偶数时不换行
        print(item+", ", end='')
    else:
        print(item+"。")                 # 换行输出
```

3/3 元组与列表的区别

1）列表属于可变序列，它的元素可以随时修改或者删除；元组属于不可变序列，其中的元素不可以修改

2）列表可以使用append()、extend()、remove()等方法实现添加和修改列表元素。但元组没有上述方法，不能进行元组元素的添加和修改

3）可以使用切片访问和修改列表中的元素。元组也支持切片，但它只支持通过切片访问元组中的元素，不支持修改

4）元组比列表的访问和处理速度快

5）列表不能作为字典的键，而元组可以

元组是不可变的序列，而列表是可变的序列

4、字典

4/1 字典的定义

字典是**无序的可变**序列，保存的内容是以**“键-值”**的形式存放的

字典的特点

- 通过键而不是通过索引来读取
- 字典是任意对象的无序集合
- 字典是可变的，并且可以任意嵌套
- 字典中的键必须唯一
- 字典中的键必须不可变
- 字典的键可以使用数字、字符串或者元组，但不能使用列表

定义字典时：

- ✓ 每个元素都包含“键”和“值”，并且“键”和“值”之间用冒号分割
- ✓ 相邻两个元组用逗号分割
- ✓ 所有元组放在一个大括号中

```
#字典示例
score = {'语文': 120, '数学': 150, '英语': 130, '物理': 108}
```

4/2 字典的创建、删除与访问

1、通过映射函数创建字典

dictionary = dict(zip(list1, list2))

- dictionary：表示字典名称
- zip()函数：将多个列表或元组对应位置的元素组为元组
- dict()函数：字典关键字
- list1：用于生成字典“键”的列表
- list2：用于生成字典“值”的列表
- 若list1和list2的长度不同，则字典与最短的列表长度相同

2、通过“键-值”对创建字典

dictionary = dict(key1=value1, key2=value2, key3=value3, ...)

- dictionary：表示字典名称
- key：表示元素的键，必须是唯一的，且不可变
- value：表示元素的值，可以是任意数据类型，不是必须唯一

3、删除字典

- 删除整个字典，不再保留：del dictionary
- 删除字典全部元素，删除后字典为空：

  dictionary.clear()

4、访问字典

- 通过指定键访问：dictionary[key]
- 通过字典对象get()方法访问：
  - ✓ dictionary.get(key[, default])
  - ✓ key为指定的键
  - ✓ [, default]为可选项，用于当key不存在时，返回一个默认值

4/2 字典的创建、删除与访问

例：星座与性格

某公司新招聘四名员工，现需要根据他们的性格特点来进行安排工作内容。星座与性格的对应关系如下表1所示，四名员工与星座的对应关系如下表2所示。

请编写一个程序，输入一个员工的姓名，输出根据星座匹配结果得到的性格特点。

表1

白羊座	有小脾气
金牛座	理财高手
双子座	追求新鲜感
巨蟹座	情绪敏感
狮子座	有宏伟理想
处女座	完美主义者
天秤座	追求平等
天蝎座	精力旺盛
射手座	崇尚自由
摩羯座	非常勤奋
水瓶座	个人主义色彩浓厚
双鱼座	善良

表2

张三	水瓶座
李四	射手座
王五	双鱼座
赵六	双子座

思路：两个字典

- ✓ “星座-性格”字典
- ✓ “姓名-星座”字典

4/2 字典的创建、删除与访问

例：星座与性格

```
#例：星座与性格
name = ['张三','李四','王五','赵六']          # 作为键的列表
sign_person = ['水瓶座','射手座','双鱼座','双子座']  # 作为值的列表
person_dict = dict(zip(name,sign_person))              # 转换为个人字典
sign_all =['白羊座','金牛座','双子座','巨蟹座','狮子座','处女座','天秤座','天蝎座','射手座','摩羯座','水瓶座','双鱼座']
nature = ['有点小脾气',
          '理财高手',
          '追求新鲜感',
          '情绪敏感',
          '有宏伟理想',
          '完美主义者',
          '追求平等',
          '精力旺盛',
          '崇尚自由',
          '非常勤奋',
          '个人主义色彩浓重',
          '善良。']
sign_dict = dict(zip(sign_all,nature))                  # 转换为星座字典
name_input = input('请输入姓名：')
print(name_input,"的星座是",person_dict.get(name_input))       # 输出星座
print("\n 他的性格特点是：\n\n",sign_dict.get(person_dict.get(name_input)))   # 输出性格特点
```

```
请输入姓名：张三
张三 的星座是 水瓶座

 他的性格特点是：

 个人主义色彩浓重
```

4/3 字典的操作

1、字典遍历

dictionary.items()

- dictionary：表示字典名称
- 返回值为可遍历的“键-值”对的元组列表
- values()：返回字典的“值”组成的列表
- keys()：返回字典的“键”组成的列表

2、添加与修改

dictionary[key] = value

- 当key不在字典中时，表示添加“key-value”
- 当key已在字典中时，表示将key对应的值修改为value

3、删除

del dictionary[key]

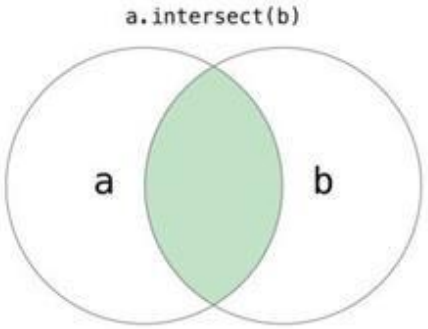
- 删除键为key的元素

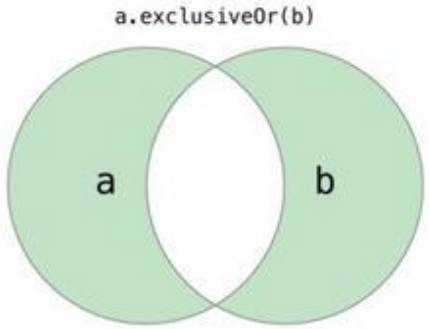
```
#字典遍历
score = {'语文': 120, '数学': 150, '英语': 130, '物理': 108}
print('键值对遍历')
for item in score.items():
    print(item)

print('键遍历')
for item in score.keys():
    print(item)

print('值遍历')
for item in score.values():
    print(item)
```

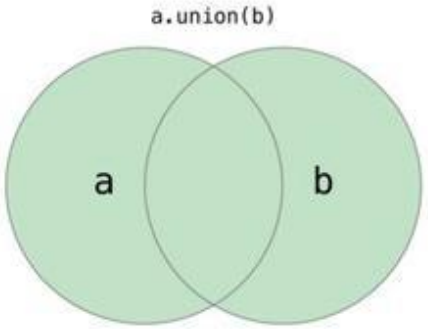
```
键值对遍历
('语文', 120)
('数学', 150)
('英语', 130)
('物理', 108)
键遍历
语文
数学
英语
物理
值遍历
120
150
130
108
```

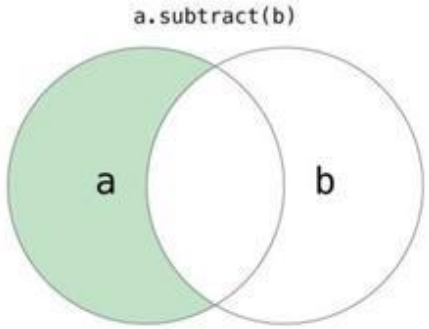
5、集合

5/1 集合的定义









- 集合：用于**保存不重复**的元素
- 集合分为**可变集合（set）**和不可变集合（frozenset）
- 在形式上，集合的所有元素都放在一对**大括号“{}”**中
- 两个相邻元素间使用**逗号“,”**分割
- 集合的关键字：**set()**

5/2 创建集合

1、使用“{}”创建

setname = {element1, element2, ...}

- setname：表示集合名称
- 在创建集合时，若输入了重复元素，Python会自动去重
- 由于集合是无序的，所以每次输出时元素的排列顺序可能不同

✓ **创建空集合：只能用set()实现**
✓ **“{}”表示创建一个空字典**

2、使用set()函数创建

setname = set(iteration)

- setname：表示集合名称
- iteration：表示要转换为集合的可迭代对象，可以是列表、元组、range对象等
- 若iteration是字符串，返回的集合是包含全部不重复字符的集合

```
#集合
set1 = {1,2,3,3,4,5}
set2 = set('python')

print(set1)
print(set2)
```

```
{1, 2, 3, 4, 5}
{'p', 'o', 'n', 'h', 't', 'y'}
```

5/3 集合的操作

1、添加元素

setname.add(element)

- setname：表示要添加元素的集合名称
- element：表示要添加的元素内容
- 添加的元素内容只能使用字符串、数字及布尔类型数据，不能使用列表、元组等可迭代对象

2、删除元素

- 删除指定元素：setname.remove(element)
- 随机删除一个元素：setname.pop()
- 清空集合：setname.clear()

✓ 使用remove()方法时，若集合不存在指定内容，则程序会抛出异常

✓ 在删除指定内容前，最好先判断该内容是否存在

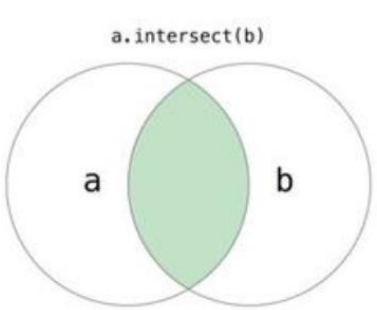
```
#集合删除元素
course_set = set(['数据结构','数据库','Python与大数据分析','Access数据库'])
course_set.remove('数据结构') #移除指定元素
print('使用remove()方法移除指定元素：',course_set)
course_set.pop() #移除一个元素
print('使用pop()方法移除一个元素：',course_set)
course_set.clear() #清空集合
print('使用clear()方法清空集合：',course_set)
```

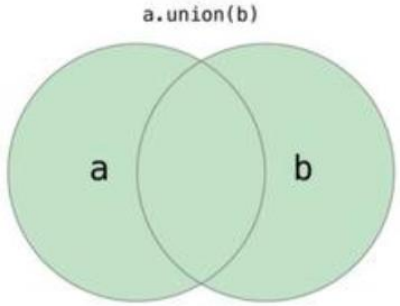
```
使用remove()方法移除指定元素： {'数据库', 'Access数据库', 'Python与大数据分析'}
使用pop()方法移除一个元素： {'Access数据库', 'Python与大数据分析'}
使用clear()方法清空集合： set()
```

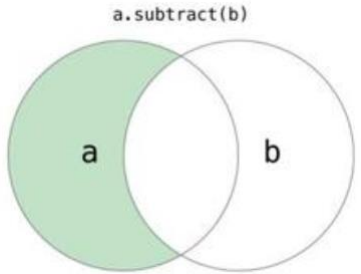
5/3 集合的操作

3、集合的交集、并集和差集运算

- 交集：A & B
- 并集：A | B
- 差集：A - B







例：在学生选课完毕之后，老师要对选课结果进行统计，具体事项如下：

1）既选择了Python又选择了C++的学生

2）选了Python和C++的全部学生

3）选了Python但没有选C++的学生

解决思路：

1）求交集

2）求并集

3）求差集

```
#集合的交并差
python = set(['甲','乙','丙','丁'])          # 保存选择Python语言的学生名字
c = set(['乙','丁','戊','己'])               # 保存选择C语言的学生名字
print('选择Python语言的学生有：',python)          # 输出选择Python语言的学生名字
print('选择C语言的学生有：',c)          # 输出选择C语言的学生名字
print('交集运算：',python & c)          # 输出既选择了Python语言又选择C语言的学生名字
print('并集运算：',python | c)          # 输出参与选课的全部学生名字
print('差集运算：',python - c)          # 输出选择了Python语言但没有选择C语言的学生名字
```

```
选择Python语言的学生有： {'甲', '丙', '乙', '丁'}
选择C语言的学生有： {'戊', '乙', '己', '丁'}
交集运算： {'乙', '丁'}
并集运算： {'甲', '丙', '乙', '己', '戊', '丁'}
差集运算： {'甲', '丙'}
```

随堂练习

- 1、小明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1～1000之间的随机整数(N<=1000),N是用户输入的。对于其中重复的数字，只保留一个，把其余相同的数字去掉，不同的数对应着不同的学生的号，然后再把这些数从小到大排序，按照排好的顺序去找同学做调查，请你协助小明完成“去重”与排序工作
- 2、重复数字统计:
  1) 随机生成1000个整数
  2) 数字范围[20,100]
  3) 升序输出所有不同的数字及其每个数字的重复次数

```
import random #Python随机数模块
n = random.randint(a,b)#产生一个在[a,b)的随机整数，并赋值给n
```