Python语言程序设计
Design and Programming of The Python Language

主讲教师：张小东
联系方式：z_xiaodong7134@163.com
答疑地点：宋健研究院514

第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

➢ 序列

- ◆ 定义：若干有共同特征的数据类型的统称。序列在python中相当于若干元素的一个容器。
- ◆ 分类：列表list、元组tuple、字符串string、Unicode字符串、buffer对象和xrange/range对象
- ◆ 通用操作

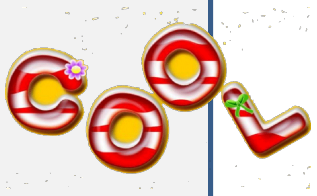
s.index(x[,start[,end])
s.count(x)

索引(indexing)—按下标取值	***s[i]***
分片(sliceing)—取一部分内容	***s[i:j:k]***
加(adding)—连接两个序列	***s+t***
乘(multiplying)—重复连接同一个序列	***s*n***
检查某个元素是否属于这序列(index)	***x in s***
计算序列长度(len)	***len(s)***
找出最大元素和最小元素(min/max)	***min(s)/max(s)***

➢ 序列

◆ 通用操作

```
>>> lst=[1,3,4]
>>> lst*3
>>> s = [1,2,3,4,5,6,7,8,9,10]
>>> print(s[1:10:2])
>>> print(s[7:10])
>>> print(s[-4:])
```



第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

➢ 序列

- ◆ 字符串的创建：声明字符串变量。
- ◆ 作为序列操作字符串：取得片段、字符串连接、字符串的重复生成、字符串之间比较大小、子串判断、求字符串的长度、最大(小)元素、字符串遍历
- ◆ 字符串特有操作：

（1）数字与字符串相互转化—str(), int(), float()

（2）格式化生成字符串

‘目标字符串’%(数据1，数据2，…，数据n)

转换字符串: %[m]s, %[m]d, %[m.n]f, %[.n]e, %%

```
>>> n,f,s=62,5.03,'string'
>>> 'n=%d,f=%f,s=%s'%(n,f,s)
```

➢ 序列

◆ 字符串操作的函数

（1）子串查找与替换函数—str.find(sub), str.rfind(sub), str.replace(old,new)

（2）查找子串的位置—str.index(sub)

（3）统计元素出现的次数

（4）裁掉特定字符的函数—str.lstrip([chars]) str.rstrip([chars]), str.strip([chars])

```
>>> 'www.example.com'.lstrip('cmowz')
>>> 'www.example.com'.rstrip('cmowz')
>>> 'www.example.com'.strip('cmowz')
>>> '   www.example.com  '.strip()
```

➢ 序列

◆ 字符串操作的函数

（5）分割子串—str.split(sep)

```
>>> s='www.example.com'
>>> L=s.split('.')
```

（6）字符串大小定相关的函数—str.lower()……

➢ 序列

【例5-1】已知一个字符串包含许多组英文单词和中文单词，请将中文和英文分别挑出来，组成中文和英文字符串。如I我am很very利害good→我很善良。I am very good.

```
def is_c(uc):
    ac=ord(uc)
    if ac>=0 and ac<128:
        return False
    else:
        return True
```

```
def main():
    s=input("input sentence:")
    s1=""
    s2=""
    deal=1
    for u in s:
        if is_c(u):
            if deal==1:
                s2=s2+' '
                deal=0
            s1=s1+u
        else:
            if deal==0:
                deal=1
            s2=s2+u
    print(s1)
    print(s2)
main()
```

➢ 序列

【例5-2】输入两个字符串，求两个字符串共有的最长子串

```
def substr():
    s1=input('输入子串1：')
    s2=input('输入子串2：')
    r=""
    m=0
    for i in range(0,len(s2)):
        for j in range(i+1,len(s2)+1):
            if s2[i:j] in s1 and m<j-i:
                r=s2[i:j]
                m=j-i
    print("最长公共的子串：",r)
substr()
```

第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

===列表和元组===

```
>>> a=10
>>> b='Wuji'
>>> L=[2,'green','blue',[a,b]]
```

➢ 列表

◆ 创建：

```
>>> L=[]
>>> L=[1,2,3]
>>> L=['red','green','blue']
```

```
>>> b='Wuji'
>>> L=list(b）
```

◆ 作为序列操作：取得片段、连接、重复生成、比较是否相同(=)、长度、最大(小)元素、遍历、是否为成员、修改

◆ 操作列表的函数：

注意区别

（1）添加—append(),extend(),insert()

（2）删除元素—pop(),remove(),del命令

```
>>> L.pop(2)
>>> L.remove('xy')
>>> del(L[1:3]）
```

```
>>> L.append('abc')
>>> L.extend(['r','xy'])
>>> L.insert(2,'three')
```

➢ 列表

◆操作列表的函数：

（3）元素位置查找—index()

（4）统计元素出现的次数—count()

（5）列表排序： sort(key=None,reverse=None), reverse()

```
>>> L=[23,1,43,67,35,7]
>>> L.sort()
>>> L.sort(reverse=True)
```

```
>>> s="This is a test string from Andres".split()
>>> s.sort(key=str.lower)
```

（6）清空列表元素—clear()

➢ 列表

【例5-3】输入一个点分IP地址，即输入形如***.***.***.***的字符串，其中***为0~255之间的整数。编程将IP地址转化为32位二进制形式输出，也就是将***转化为8位二进制数后依次连接起来形成32位二进制数

【问题分析】判断是否为合法IP；将每个地址段转换为二进制数；重新拼接输出

```
def isVIP(L):
    if len(L)!=4:
        return False
    for i in range(4):
        if L[i].isdigit()==False or int(L[i])<0 or int(L[i])>255:
            return False
    return True
```

➢ 列表

【例5-3】IP地址的转换

```
def _10to2(num):
    res=""
    while True:
        res=str(num%2)+res
        num=num//2
        if num==0:
            break
    while len(res)<8:
        res='0'+res
    return res
```

```
num,re = divmod(num, 2)
res=str(re)+res
```

```
def main():
    ipS=input('input IP:')
    L=ipS.split('.')
    while not isVIP(L):
        print("IP is error!")
        ipS=input('input IP,again:')
        L=ipS.split('.')
    s=""
    for i in range(4):
        s=s+' '+_10to2(int(L[i]))
    print(s)
main()
```

```
def _10to2(num):
    s=bin(num)
    res=s[2:len(s)+1].rjust(8,'0')
    return res
```

【练习】扑克牌有四种花色：黑红梅方。扑克牌按点数从小到大是2，3，4，……，J,Q,K,A。编程序实现下列目标：

（1）按花色排序输出（小到大）

（2）按花色排序后，将同色牌按点数排序输出（大到小）

（3）第2步完成后，去除同色的牌

一手牌：['梅花A','方块4','梅花2','方块4','红桃7','黑桃Q','红桃K','梅花9','方块9','红桃5','梅花J','方块8','红桃5','黑桃3','黑桃10','黑桃3','红桃7','黑桃Q']

按花色整理：['黑桃Q', '黑桃3', '黑桃10', '黑桃3', '黑桃Q', '红桃7', '红桃K', '红桃5', '红桃5', '红桃7', '梅花A', '梅花2', '梅花9', '梅花J', '方块4', '方块4', '方块9', '方块8']

['黑桃Q','黑桃Q','黑桃10','黑桃3','黑桃3','红桃K','红桃7','红桃7','红桃5','红桃5','梅花A','梅花J','梅花9','梅花2','方块9','方块8','方块4','方块4']

['黑桃Q','黑桃10','黑桃3','红桃K','红桃7','红桃5','梅花A','梅花J','梅花9','梅花2','方块9','方块8','方块4']

【练习】扑克牌排序输出

```
#给花色编码
def tkey(s):
    if s[0:2]=="方块":
        return 4
    elif s[0:2]=="梅花":
        return 3
    elif s[0:2]=="红桃":
        return 2
    elif s[0:2]=="黑桃":
        return 1
    else:
        return 0
```

```
#转换为数字信息
def gP(s):
    if s[2:]=='A':
        return 14
    elif s[2:]=='K':
        return 13
    elif s[2:]=='Q':
        return 12
    elif s[2:]=='J':
        return 11
    else:
        return int(s[2:])
```

```
def m():
    L=['梅花A','方块4','梅花2','方块4','红桃7','黑桃Q','红桃K',
'梅花9','方块9', '红桃5','梅花J','方块8','红桃5','黑桃3','黑
桃10','黑桃3','红桃7','黑桃Q']
    L.sort(key=tkey) #按花色排序
    print(L)
```

【练习】扑克牌排序输出

```
def reR(L):
    for e in L:
        for i in range(1,L.count(e)):
            L.remove(e)
```

```
i=0
j=0
p=["黑桃","红桃","梅花","方块"]
L2=[]
for k in range(4):
    i=j
    while j<len(L) and p[k] in L[j]:
        j=j+1
    s=L[i:j]
    s.sort(key=gP,reverse=True)
    L2=L2+s
L=L2
print(L)
```

➢ 列表

◆ 用列表表示多维数据

（1）表示方法

```
>>> M=[[1,2,3],
       ['a','b','c'],
       [7,8,9]]
```

✓ 列表的每个维度长度可以不同
✓ 列表的元素类型可以不同

（2）元素添加

```
a=[]
for i in range(2):
    a.append([])
    for j in range(2):
        v=int(input("input element:"))
        a[i].append(v)
print(a)
```

```
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=" ")
    print()
```

➢ 列表

◆ 用列表表示多维数据

【例5-5】已知平面上若干点的坐标：

a0(1,2), a1(-1,3), a2(2,1.5), a3(-2,0) ,a4(4,2)

计算任意两的距离并生成距离矩阵，其中，矩阵元素(i, j)表示ai和aj之间的距离，最后输出距离矩阵和两点之间最大距离

【5-5】

```
from math import *
def d(x1,y1,x2,y2):
    return sqrt((x1-x2)**2+(y1-y2)**2)
```

```
def ma():
  x=[1,-1,2,-2,4]
  y=[2,3,1.5,0,2]
  dd=[]
  s=0
  for i in range(len(x)):
   dd.append([])
   for j in range(len(x)):
      v=d(x[i],y[i],x[j],y[j])
      dd[i].append(v)
      if s<dd[i][j]:
         s=dd[i][j]
  for i in range(len(x)):
     for j in range(len(x)):
        print("%5.2f"%dd[i][j],end="  ")
     print()
  print("max=%5.2f"%s)
```

➢ 元组

◆ 元组的创建

```
>>> t=(1,2,3)
>>> t=('a','b',[1,2])
>>> s='city'
>>> t=tuple(s)
```

◆ 元组的操作

元组是不可变的。可以看做元素固定不变的列表

第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

➢ 字典

```
>>> D={}
>>> D['spam']=2
>>> D['eggs']=3
>>> D1=dict(name='Jack',age=45)
```

◆ 字典的基本操作：

（1）创建

（2）元素修改 **D[key]=value**

（3）元素添加 **D[newkey]=newvalue**

（4）元素删除 **del D[key], D.clear()**

（5）测试元素在字典中 **key in D**

（6）元素遍历

（7）元素个数 **len(D)**

（8）判断两个字典是否相同

```
>>> D1={'Jack':45,'Susan':66}
>>> D2={'Susan':66,'Jack':45}
>>> D1==D2
```

➢ 字典

◆ 字典常用函数

（1）用keys(),values(),items()获取视图

```
>>> D={'a':'1','b':'2','c':'3'}
>>> keyview=D.keys()
>>> itemview=D.items()
>>> valueview=D.values()
```

（2）用get() 获取字典的值

```
>>> D.get('a')
>>> D.get('x','查无此人')
>>> D['x']
```

➢ 字典

◆ 字典常用函数

（3）用pop()删除字典的值

```
>>> D.pop('c')
```

（4）用update()更新或添加元素

```
>>> D.update(c=6)
>>> D.update(c=8)
```

➢ 字典

将输入字符串转化为列表

建立主关键字字典

建立次关键字字典

【例5-5】请用字典类型完成例5-4，[illegible]不

```
L=input("输入多张扑克牌值(花色+点数)以逗号分隔(黑桃3,红桃6)：  ").split(',')
pk={'方块':4,'梅[illegible]
                                  ,'6':6,'5':5,'4':4,'3':3,'2':2,'1':1}
```

用列表中的关键字初始化字典，达到去除重复项的目的

单关键字λ升序，L中[illegible]被映射至k，以k的前两位作为字典pk的下标取出对应值

```
L.sort(key=lambda k:pk[k[0:2]])
print("---按花色从小到大排序---")
print(L)
```

```
print("---去除重复元素---")
dic={}
dic=dic.fromkeys(L)
L1=list(dic.keys())
```

```
L.sort(key=lambda k:(-pk[k[0:2]],sk[k[2:]]),reverse=True)
print("---按花色从小到大，再按点数从大到小排序---")
print(L)
```

再次反序

两个关键字排序，带负号表示从大到小。两个关键字排序正好相反

➢ 字典

【例5-6】输入一段英文文字，统计其中出现的英文单词及其出现次数。要求程序可以过滤掉常见的标点符号，并按下面要求输出：

（1）将出现次数大于2的单词按字典序输出

（2）将出现次数大于2的单词按单词出现次数从大到小排序输出

[问题分析]本例的关键是分离单词，但首先要滤掉标点符号。过滤标点可以将常见标点全部 替换为空格或一种标点，然后按这种唯一的标点分离出每个单词。本例要求统计每个单 词的出现次数，可以用字典表示，单词是键，次数是值。

```
def ma():
    txt=input('input text:')
    wordC={}
    for e in " !;.\t\n\"()-:#@":
        txt=txt.replace(e,',')
    L=txt.split(',')
    L.sort()
    while L[0].isdigit() or L[0]=='':
        del L[0]
```

```
for e in words:
        if wordC[e]>2:
            print(e,wordC[e])
    print('按频率排序输出（>2）:')
    L1=list(wordC.items())
    L1.sort(key=gN,reverse=True)
    for i in range(len(L1)):
        if L1[i][1]>2:
            print(L1[i][0],L1[i][1])
```

```
for e in L:
        if e in wordC:
            wordC[e]=wordC[e]+1
        else:
            wordC[e]=1
    print('按字典输出单词及次数（>2）:')
    words=list(wordC.keys())
    words.sort()
```

```
def gN(x):
    return x[1]
```

第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

```
>>> s=set()
>>> s=set(['a','b'])
```

➢ 集合

元素是无序且不能重复的。分为可变集合**(set),**不可变集合**(Frozenset)**

◆集合的基本操作：

（1）创建

```
>>> s={i for i in range(10)}
```

```
>>> s={1,2,3}
>>> s={('a','b'),1,2}
```

（2）元素添加**s.add(元素)，s.update()**

```
>>> s.add('zxd')
>>> s.update([0,1,2],('a','b')'zxd')
```

（3）元素删除 **s.discard(obj), s.remove(obj),s.pop(),s.clear()**

（4）判断元素是否属于集合

```
>>> s.discard('a')
```

（5）元素个数 **len(D)**

➢ 集合

◆ 判断集合间的关系

所用符号：<, <=, >, >=, ==, !=

设s和t是两个集合，则：

- 若s<t为真，则s是t的真子集
- 若s>t为真，则s真包含子集t
- 若s<=t为真，则s是t的子集 s.issubset(t)
- 若s>=t为真，则s包含子集t s.isuperset(t)
- 若s==t为真，则s与t相同
- 若s!=t为真，则s与t不相同

➢ 集合

◆ 集合的交并差运算

设s和t是两个集合

（1）求两个集合的交集 **intersection() &**

（2）求两个集合的并集 **union() |**

（3）求两个集合的差 **difference() -**

【5-7】通过对集合和列表进行下列操作：

（1）向列表添加**10000**个元素，向集合添加**10000**个元素；

（2）向列表插入**10000**个元素，向集合添加**10000**个元素；

（3）判断**10000**个元素是否在列表中，判断**10000**个元素是否在集合中；

（4）从列表中删除**10000**个元素，从集合中删除**10000**个元素

比较集合类型和列表类型的相似操作在时间效率上的差异

➢ 集合

◆ 集合的交并差运算

【5-8】验证哥德巴赫猜想：任何一个超过**2**偶数都可以写成两个素数之和。

问题分析：

（**1**）将一个超过**2**偶数**N**，分解为两个数之和，如**N=k1+k2**，分别判断它是否为质数。

（**2**）找出小于**N**所有素数，建立一个素数表**L**，取出一个素数**e**，判断**N-e in L**，若为真，则找到此合数素数分解

【5-8】验证哥德巴赫猜想：任何一个超过2偶数都可以写成两个素数之和。

```
def pm():
    N=int(input("请输入待验证的偶数n(n>2):"))
    while N<3 or N%2==1:
        print('不符合要求！ ')
        N=int(input("请输入待验证的偶数n(n>2):"))
```

```
pme=set()
    for i in range(2,N+1):
        pme.add(i)
    for i in range(2,N+1):
        if i in pme:
            for k in range(2*i,N+1,i):
                if k in pme:
                    pme.remove(k)
```

```
for e in pme:
        f=N-e
        if f>=e and f in pme:
            print(N,'=',e,'+',f)
```

第5章 复杂数据类型

主要内容

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器

===可变类型和不可变类型===

➢ 迭代器

◆概述

一种对象，提供在某种容器(列表、字典等)上的遍历元素的方法

__iter__()：返回迭代器对象自身

__next__()：返回容器中的下一个元素。

惰性求值：不需要事先准备好整个迭代过程中的所有元素，迭代至某个元素时才获取该元素。可以遍历巨大文件。

◆使用迭代器

凡是可以通过迭代器进行访问的对象都是可迭代对象

迭代器的获取方式：

iter(某种可迭代的对象**)**

===可变类型和不可变类型===

➢ 迭代器

```
>>> s={1,2}
>>> it=iter(s)
>>> it.__next__()
1
>>> it.__next__()
2
>>> it.__next__()
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    it.__next__()
StopIteration
```

```
lst={1,2}
it=iter(lst)
try:
    while True:
        val=it.__next__()
        print(val)
except StopIteration:
    print("finish!")
```

===可变类型和不可变类型===

➢ 生成器

生成器是一个带有**yield**语句的函数，它用于产生一系列数据。其格式：

def 函数名（参数）：

……

yield 变量

```
def counter(start=0):
    while True:
        yield(start)
        start+=1
```

```
>>> g=counter(0)
>>> g.__next__()
0
>>> g.__next__()
1
>>> g.__next__()
2
>>> g.__next__()
```

```
>>> def get_0_1():
        yield(0)
        yield(1)
>>> g=get_0_1()
>>> g.__next__()
0
>>> g.__next__()
1
>>> g.__next__()
```

- 生成器

【例5-9】利用生成器构造一个**fibonacci**函数，生成**fibonacci**的小于**100**的数。

```
def fibonacci():
    a=b=1
    yield(a)
    yield(b)
    while True:
        a,b=b,a+b
        yield(b)
def m():
    for num in fibonacci():
        if num>100:
            break
        print(num, end=' ')
```

本章小结

- 序列的概念
- 字符串
- 列表和元组
- 字典
- 集合
- 迭代器和生成器