

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4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

===序列的概念===

【定义】若干有共同特征的数据元素的集合，元素容器。

【分类】列表list、元组tuple、字符串string、Unicode字符串、buffer对象和range对象

【通用操作】

索引(indexing)—按下标取值	<i>s[i]</i>
分片(sliceing)—取一部分内容	<i>s[i:j:k]</i>
加(adding)—连接两个序列	<i>s+t</i>
乘(multiplying)—重复连接同一个序列	<i>s*n</i>
检查某个元素是否属于这序列(index)	<i>x in s</i>
计算序列长度(len)	<i>len(s)</i>
找出最大元素和最小元素(min/max)	<i>min(s)/max(s)</i>
元素的位置检索	<i>s.index(x[,start[,end]])</i>
计算元素出现的频率	<i>s.count(x)</i>

第4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

===字符串===

【定义】 Unicode字符序列。三种表示，如：

'abcde'、"super"、"""z"""

—可以相互嵌套

【索引方法】



(1) 顺次：从左向右：0～(n-1)；从右向左：-1～-n

(2) 片选：[N:M] #不包括M，[:M]左到M-1，[N:]N到M-1

【字节序列】数据是一系列的字节，并以'b'开头的字符串。
可以把Unicode字符串转为字节序列（str）。

```
>>> str="abcd字节序列"
```

```
>>> print(str)
```

```
abcd字节序列
```

```
>>> a=str.encode("utf-8")
```

```
>>> a=str.encode("gb2312")
```

===字符串===

【创建】声明字符串变量。

【通用操作】分片、加、乘、比较大小、子串判断、计算长度、最大(小)元素、遍历

【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)
```

===字符串===

【特有操作】

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

(4) 查找子串的位置—`str.index(sub)`

(5) 统计元素出现的次数—`str.count(sub)`

(6) 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()
```

(7) 分割子串—`str.split(sep).....`

```
>>> s='www.example.com'
```

```
>>> L=s.split('.')
```

===字符串===

【例4-1】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→我很利害。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()
```


===字符串===

【例4-2】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4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===

➤ 列表

【创建】

```
>>> L=[]
```

```
>>> b='Wuji'
```

```
>>> L=list(b)
```

```
>>> a=10
```

```
>>> b='Wuji'
```

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

【操作】分片、连接、乘、比较(==)、长度、最大(小)元素、遍历、是否为成员、修改...

【操作函数】

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')
```

```
L=['a','b','b','c','d','b','a']
```

```
y=['b','c']
```

```
for i in L:
```

```
    if i in y:
```

```
        L.remove(i)
```

```
print(L)
```

```
L=['a','b','b','c','d','b','a']
```

```
y=['b','c']
```

```
for i in L[:]:
```

```
    if i in y:
```

```
        L.remove(i)
```

```
print(L)
```

===列表和元组===

➤ 列表

【操作函数】

(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()**

(6) 清空列表元素—**clear()**

```
>>> 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)
```

===列表和元组===

➤ 列表

【例4-3】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
```

===列表和元组===

➤ 列表

【例4-3】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
```

===列表和元组===

【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，去除相同花色中数字重复的牌

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']

===列表和元组===

【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)

===列表和元组===

【练习】扑克牌排序输出

```
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)
```

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

===列表和元组===

➤ 元组

【创建】

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

【操作】

元组是不可变的。可以看做元素固定不变的列表。

第4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===

【定义】 以{key: value}形式组织数据

【基本操作】

- (1) 创建 变量D={}或dict #空字典对象
- (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) 元素个数 len(D)
- (7) 判断两个字典是否相同

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

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

===字典===

【操作函数】

(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']
```

(3) 创建字典**fromkeys()**

```
dict.fromkeys(seq[, value])
```

(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)
```

```
seq = ('Google','Runoob','Taobao')
dict = dict.fromkeys(seq)
```

===字典===

【例4-5】请用字典

将输入字符串转
化为列表

所有元素均不变

建立主关键
字字典建立次关键
字字典

```
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,'梅花':3,'红桃':2,'黑桃':1}
```

```
pk={'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

单关键字升序，L中元素
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4-6】 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】

关键：分离单词。

- (1) 符号替换。标点全部替换为空格或一种标点
- (2) 分离出每个单词。
- (3) 用字典表示，单词是键，次数是值。

===字典===

```
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4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

===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**)

【操作】

(1) 创建

```
>>> s=set()
```

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

```
>>> s={1,2,3}
```

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

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

(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===

【集合间的关系运算】

所用符号: $<$, \leq , $>$, \geq , $==$, $!=$

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 $s \leq t$ 为真, 则 s 是 t 的子集 `s.issubset(t)`
- 若 $s \geq t$ 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
- 求两个集合的交集 `intersection()` &
- 求两个集合的并集 `union()` |
- 求两个集合的差 `difference()` -

===集合===

【例4-8】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 $N=k_1+k_2$ ，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

```
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):"))
```

===集合===

【例4-8】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。

```
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4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

===迭代器和生成器===

➤ 迭代器

【定义】提供在某种容器(列表、字典等)上的遍历元素的方法

【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。

【操作】

(1) 创建: **iter(某种可迭代的对象)**

(2) 迭代器对象查看: **__iter__()/iter**

(3) 获取迭代器元素: **__next__()/next**

```
>>> s={1,2}
>>> it=iter(s)
>>> it.__next__()
```

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

===迭代器和生成器===

➤ 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
```

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

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

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


===迭代器和生成器===

➤ 生成器

【例4-9】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