



实用Python 程序设计

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕！



Python组合数据类型(二) 列表



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表基本操作



冰岛杰古沙龙冰河湖

列表的增删和修改

➤ 列表可以增删元素，列表的元素可以修改，列表元素可以是任何类型

```
empty = []    #[] 表示空表
```

```
list1 = ['Pku', 'Huawei', 1997, 2000];
```

```
list1[1] = 100 #列表元素可以赋值
```

```
print (list1)    #>>['Pku', 100, 1997, 2000]
```

```
del list1[2]      #删除元素    list1.pop(2) 等效
```

```
print (list1)    #>>['Pku', 100, 2000]
```

```
list1 += [100,110]
```

#添加另一列表的元素100和110，在list1原地添加，没有新建一个列表

```
list1.append(200) #添加元素 200，append用于添加单个元素
```

列表的增删和修改

```
print(list1)    #>>['Pku', 100, 2000, 100, 110, 200]
```

```
list1.append(['ok',123]) #添加单个元素
```

```
print(list1)    #>>['Pku', 100, 2000, 100, 110, 200, ['ok', 123]]
```

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

```
n = [1, 2, 3]
```

```
x = [a, n]      #a若变, x也变
```

```
a[0] = 1
```

```
print(x)        #>>[[1, 'b', 'c'], [1, 2, 3]]
```

```
print(x[0])     #>>[1, 'b', 'c']
```

```
print(x[0][1])  #>>b
```

列表相加

➤ 列表相加可以得到新的列表

```
a = [1,2,3,4]
```

```
b = [5,6]
```

```
c = a + b
```

```
print(c)      #>>[1, 2, 3, 4, 5, 6]
```

```
a[0] = 100
```

```
print(c)      #>>[1, 2, 3, 4, 5, 6]
```

列表和+=

➤ 对列表来说，`a+=b` 和 `a=a+b`不同

```
b = a = [1,2]
```

```
a += [3]          #b和a指向相同地方，在a末尾添加元素，b也受影响
```

```
print(a,b)       #>>[1, 2, 3] [1, 2, 3]
```

```
a = a + [4,5]    #对a重新赋值，不会影响到b
```

```
print(a)         #>>[1, 2, 3, 4, 5]
```

```
print(b)         #>>[1, 2, 3]
```

列表乘法

```
print([True] * 3)
```

```
#>>[True, True, True]
```

```
a = [1,2]
```

```
b = a * 3
```

```
print(b)
```

```
#>>[1, 2, 1, 2, 1, 2]
```

```
print([a*3])
```

```
#>>[[1, 2, 1, 2, 1, 2]]
```

```
c = [a] * 3
```

```
print(c)
```

```
#>>[[1, 2], [1, 2], [1, 2]]
```

```
a.append(3)
```

```
print(c)
```

```
#>>[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

```
print(b)
```

```
#>>[1, 2, 1, 2, 1, 2]
```


列表的切片

➤ 列表的切片返回新的列表，用法和字符串切片相同

```
a = [1,2,3,4]
```

```
b = a[1:3]
```

```
print(b)          #>>[2, 3]
```

```
b[0] = 100
```

```
print(b)          #>>[100, 3]
```

```
print(a)          #>>[1, 2, 3, 4]
```

```
print(a[::-1])    #>>[4, 3, 2, 1]
```

```
print([1,2,3,4,5,6][1:5:2])  #[2,4]
```

```
print(a[:])       #>>[1,2,3,4]
```

`a = [1,2,3,4]`

接下来哪条语句不正确?

- ☐ A `a.append([5])`
- ☐ B `a += [5,6]`
- ☐ C `a += [5]`
- ☐ D `a += 5`

提交

`a = [1,2,3,4]`
接下来哪条语句不正确?

- ☐ A `a.append([5])`
- ☐ B `a += [5,6]`
- ☐ C `a += [5]`
- ☒ D `a += 5`

提交

列表比大小

- 两个列表比大小，就是逐个元素比大小，直到分出胜负。
- 如果有两个对应元素不可比大小，则出 `runtime error`。

```
print([1,'a',12 ] < [1,'b',7])      #>>True
print([1,'a' ] < [1,'a',13])        #>>True
print([2,'a' ] > [1,'b',13])        #>>True
print([2,'a' ] < ['ab','b',13])     # runtime error
```

列表的遍历

```
lst = [1,2,3,4]

for x in lst:
    print(x,end = " ")
    x = 100    #不会修改列表的元素

print(lst)    #>>[1, 2, 3, 4]

for i in range(len(lst)):
    lst[i] = 100

print(lst)    #>>[100, 100, 100, 100]
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表基本应用



河南云台山

列表应用例题:校门外的树

某校大门外长度为 L 的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在 L 的位置；数轴上的每个整数点，即0, 1, 2, …, L ，都种有一棵树。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，马路上还有多少棵树。

列表应用例题:校门外的树

输入

第一行有两个整数 L ($1 \leq L \leq 10000$) 和 M ($1 \leq M \leq 100$)， L 代表马路的长度， M 代表区域的数目， L 和 M 之间用一个空格隔开。接下来的 M 行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标。

输出

包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

列表应用例题:校门外的树

样例输入

```
500 3  
150 300  
100 200  
470 471
```

样例输出

```
298
```

列表应用例题:校门外的树

```
s = input().split()
L,M = int(s[0]),int(s[1])
good = [True] * (L+1) #good[i] 为True表示坐标i的树还在
for i in range(M):
    s = input().split()
    start,end = int(s[0]),int(s[1])
    for k in range(start,end + 1):
        good[k] = False #坐标k处的树被移走了
print(sum(good)) #sum是python函数, 可以求列表元素和

#True就是1,False就是0
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

郭炜

列表的朴素排序算法



瑞士阿尔卑斯山少女峰

列表的朴素排序算法：选择排序

如果有 N 个元素需要排序，那么首先从 N 个元素中找到最小的那个放在下标0处（可以通过让它和原来的下标为0的元素交换位置来实现），然后再从剩下的 $N-1$ 个元素中找到最小的放在下标1处，然后再从剩下的 $N-2$ 个元素中找到最小的放在下标2处……直到剩下最后2个元素中最小的被放在了下标 $n-2$ 处，所有的元素即都就位。

列表的选择排序

```
def SelectionSort(a): #选择排序
    #将列表a从小到大排序
    n = len(a)
    for i in range(n-1):
        #每次从a[i]及其右边的元素里选出最小的，放在a[i]这个位置
        for j in range(i+1,n): #依次考察a[i]右边元素
            if a[j] < a[i]:
                a[i],a[j] = a[j],a[i]

lst = [1,12,4,56,6,2]
SelectionSort(lst)
print(lst) #>>[1, 2, 4, 6, 12, 56]
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

用排序函数 对简单列表排序



新加坡金沙酒店

用缺省的比较规则排序

- `a.sort()` 可以对列表a从小到大排序
- `sorted(a)` 返回a经过从小到大排序后的新列表，a不变

```
a = [5,7,6,3,4,1,2]
```

```
a.sort()           #对a从小到大排序
```

```
print(a)           #>>[1, 2, 3, 4, 5, 6, 7]
```

```
a = [5,7,6,3,4,1,2]
```

```
b = sorted(a)      #a不因此而改变
```

```
print(b)           #>>[1, 2, 3, 4, 5, 6, 7]
```

```
print(a)           #>>[5, 7, 6, 3, 4, 1, 2]
```

```
a.sort(reverse = True)  #对a从大到小排序
```

```
print(a)           #>>[7, 6, 5, 4, 3, 2, 1]
```

用缺省的比较规则排序

```
students = [('John', 'A', 15), # 姓名, 成绩, 年龄
            ('Mike', 'C', 19),
            ('Mike', 'B', 12),
            ('Mike', 'C', 18),
            ('Bom', 'D', 10)]
```

```
students.sort() #先按姓名, 再按成绩, 再按年龄排序
```

```
print(students)
```

```
#>>[('Bom', 'D', 10), ('John', 'A', 15), ('Mike', 'B', 12),
      ('Mike', 'C', 18), ('Mike', 'C', 19)]
```


自定义比较规则的排序

➤ 自定义关键字函数 key

```
def myKey(x):    #关键字函数
    return x % 10
```

```
a = [25,7,16,33,4,1,2]
```

```
a.sort(key = myKey)
```

key是函数，sort按对每个元素调用该函数的返回值从小到大排序

[1, 2, 33, 4, 25, 16, 7] 按个位数排序

```
sorted("This is a test string from Andrew".split(),
      key=str.lower))
```

```
# ['a', 'Andrew', 'from', 'is', 'string', 'test', 'This']
```

不区分大小写排序



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

复杂列表的 自定义排序



挪威峡湾边的小镇

自定义比较规则的排序

➤ 用不同关键字排序

```
students = [  
    ('John', 'A', 15), # 姓名, 成绩, 年龄  
    ('Mike', 'B', 12),  
    ('Mike', 'C', 18),  
    ('Bom', 'D', 10)]  
  
students.sort(key = lambda x: x[2] ) # 按年龄排序  
  
#[('Bom', 'D', 10), ('Mike', 'B', 12), ('John', 'A', 15), ('Mike', 'C', 18)]  
  
students.sort(key = lambda x: x[0] ) # 按姓名排序  
  
# [('Bom', 'D', 10), ('John', 'A', 15), ('Mike', 'B', 12), ('Mike', 'C', 18)]
```

lambda 表达式

```
lambda x: x[2]
```

表示一个函数，参数是`x`，返回值是 `x[2]`

```
k = lambda x,y : x + y
```

`#k是一个函数，参数是x, y，返回值是x+y`

```
print(k(4,5))
```

`#>>9`

自定义比较规则的排序

➤ 多级排序

```
def f(x):
```

```
    return (-x[2], x[1], x[0])
```

```
students = [('John', 'A', 15), ('Mike', 'C', 19),  
            ('Wang', 'B', 12), ('Mike', 'B', 12),  
            ('Mike', 'C', 12),  
            ('Mike', 'C', 18),  
            ('Bom', 'D', 10)]
```

```
students.sort(key = f ) #先按年龄从高到低，再按成绩从高到低，再按姓名字典序
```

```
print(students)
```

```
#>>[('Mike', 'C', 18), ('John', 'A', 15), ('Mike', 'B', 12), ('Wang',  
    'B', 12), ('Mike', 'C', 12), ('Bom', 'D', 10)]
```

Python元组的排序

- 元组不能修改, 因此无sort函数, 可以用sorted得到新的排序后的列表

```
def f(x):  
    return (-x[2], x[1], x[0])  
  
students = (('John', 'A', 15), ('Mike', 'C', 19),  
            ('Wang', 'B', 12), ('Mike', 'B', 12),  
            ('Mike', 'C', 12), ('Mike', 'C', 18),  
            ('Bom', 'D', 10))    #students是元组  
  
print(sorted(students, key = f))    #sorted的结果是列表  
  
#>>[('Mike', 'C', 19), ('Mike', 'C', 18), ('John', 'A', 15), ('Mike', 'B',  
    12), ('Wang', 'B', 12), ('Mike', 'C', 12), ('Bom', 'D', 10)]
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表相关函数



美国马蹄湾

列表相关函数

- `append(x)` 添加元素x到尾部
- `count(x)` 计算列表中包含多少个x
- `extend(x)` 添加列表x中的元素到尾部
- `insert(i,x)` 将元素x插入到下标i处
- `remove(x)` 删除元素x，如果x不存在，则引发异常
- `reverse()` 颠倒整个列表
- `index(x)` 查找元素x，找到则返回第一次出现的下标，找不到则引发异常
- `index(x,s)` 从下标s开始查找x

列表相关函数

```
a,b = [1,2,3],[5,6]
```

```
a.append(b)
```

```
print(a)                #>>[1, 2, 3, [5, 6]]
```

```
b.insert(1,100)
```

```
print(a)                #>>[1, 2, 3, [5, 100, 6]]
```

```
a.extend(b)
```

```
print(a)                #>>[1, 2, 3, [5, 100, 6], 5, 100, 6]
```

```
a.insert(1,'K')
```

```
a.insert(3,'K')
```

```
print(a)                #>>[1, 'K', 2, 'K', 3, [5, 100, 6], 5, 100, 6]
```

列表相关函数

```
a.remove('K')  
print(a)                #>>[1, 2, 'K', 3, [5, 100, 6], 5, 100, 6]  
a.reverse()  
print(a)                #>>[6, 100, 5, [5, 100, 6], 3, 'K', 2, 1]  
print(a.index('K'))     #>>5  
try:  
    print(a.index('m'))  #找不到'm', 会引发异常  
except Exception as e:  
    print(e)            #>>'m' is not in list
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表映射和过滤



挪威峡湾

列表映射

- `map(function, sequence)`，可用于将一个序列(列表、元组、集合...)映射到另一个序列
- 返回一个延时求值对象，可以转换成`list`, `tuple`, `set`...

```
def f(x):  
    print(x,end="")  
    return x*x  
  
a = map(f,[1,2,3])  
print(list(a))           #>>123[1, 4, 9]  
print(tuple(a))          #>>()  
  
a = list(map(lambda x:2*x, [2,3,4]))  
print(a)                  #>>[4, 6, 8]
```

列表映射

➤ map 用于输入

```
x,y,z = map(int,input().split())  
print(x,y,z)
```

输入: 1 23 45

输出: 1 23 45

列表映射

➤ map 映射多个序列

```
list1 = [1,2,3,100]
```

```
list2 = {10,20,30}
```

```
tuple1 = [100,200,300,'ok','me']
```

```
x = list(map(lambda x,y,z : x+y+z, list1, list2, tuple1))
```

```
print(x)          #>>[111, 222, 333]
```

列表过滤

- `filter(function, sequence)`，抽取序列中令`function(x)`为True的元素`x`
- 返回一个延时求值对象，可以转换成`list`, `tuple`, `set`...

```
def f(x):
```

```
    return x % 2 == 0
```

```
lst = tuple(filter(f, [1, 2, 3, 4, 5]))
```

#抽取出偶数

```
print(lst)      #>> (2, 4)
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表生成式



挪威dalsnibba

列表生成式

```
[x * x for x in range(1, 11)]
```

```
⇒ [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
[x * x for x in range(1, 11) if x % 2 == 0]
```

```
⇒ [4, 16, 36, 64, 100]
```

```
[m + n for m in 'ABC' for n in 'XYZ']
```

```
⇒ ['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']
```

```
[[m + n for m in 'ABC'] for n in 'XYZ']
```

```
⇒ [['AX', 'BX', 'CX'], ['AY', 'BY', 'CY'], ['AZ', 'BZ', 'CZ']]
```

```
L = ['Hello', 'World', 18, 'Apple', None]
```

```
[s.lower() for s in L if isinstance(s, str)]
```

```
⇒ ['hello', 'world', 'apple']
```

```
[s for s in L if isinstance(s, int)] => [18]
```

元组生成式

```
print(tuple(x * x for x in range(1, 11)))
```

```
#>>(1, 4, 9, 16, 25, 36, 49, 64, 81, 100)
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

二维列表



挪威大洋路

二维列表

➤ 二维列表a可以看作是矩阵， $a[i][j]$ 就是第i行第j列的元素

错误的生成二维列表的方法：

```
a = [0, 0, 0]
```

```
b = [a] * 3
```

#b有三个元素，都是指针，都和a指向同一地方

```
print(b)
```

```
#>>[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
b[0][1] = 1
```

```
a[2] = 100
```

```
print(b)
```

```
#>>[[0, 1, 100], [0, 1, 100], [0, 1, 100]]
```

定义二维列表

➤ 正确做法:

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
print(matrix)    #>>[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
print(matrix[1][2],matrix[2][2])    #>>6 9
```

```
matrix[1][1] = 100
```

```
print(matrix)    #>>[[1, 2, 3], [4, 100, 6], [7, 8, 9]]
```

```
matrix = [[0 for i in range(3)] for i in range(3)]
```

```
print(matrix)    #>>[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
matrix = [[i*3+j for j in range(3)] for i in range(2)]
```

```
print(matrix)    #>>[[0, 1, 2], [3, 4, 5]]
```

定义二维列表

➤ 正确做法:

#生成一个3行4列的矩阵，所有元素都是0

```
lst = []
```

```
for i in range(3):
```

```
    lst.append([0] * 4)
```

```
lst[0][0] = lst[2][3] = 100
```

定义二维元组

```
matrix = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
```

```
print(matrix) #>>((1, 2, 3), (4, 5, 6), (7, 8, 9))
```

```
matrix = tuple(tuple(0 for i in range(3)) for i in range(3))
```

```
print(matrix) #>>((0, 0, 0), (0, 0, 0), (0, 0, 0))
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表拷贝



挪威奥勒松

列表拷贝

```
a = [1,2,3,4]
```

```
b = a[:]    #b是a的拷贝，b和a不是同一个对象，指向不同东西
```

```
b[0] = 5
```

```
print(a)    #>>[1, 2, 3, 4]
```

```
b += [10]
```

```
print(a)    #>>[1, 2, 3, 4]
```

```
print(b)    #>>[5, 2, 3, 4, 10]
```

列表深拷贝

```
a=[1,[2]]
```

```
b=a[:]
```

```
b.append(4)
```

```
print(b) #>>[1, [2], 4]
```

```
a[1].append(3)
```

```
print(a) #>>[1, [2, 3]]
```

```
print(b) #>>[1, [2, 3], 4]
```

未能进行深拷贝！

列表深拷贝

```
import copy

a = [1, [2]]

b = copy.deepcopy(a)

b.append(4) #>>[1, [2], 4]

print(b)

a[1].append(3)

print(a) #>>[1, [2, 3]]

print(b) #>>[1, [2], 4]
```



北京大学
PEKING UNIVERSITY

信息科学技术学院 郭炜

列表转换



冰岛众神瀑布

元组和列表互转

```
a=[1,2,3]
```

```
b=tuple(a)      #b:  (1,2,3)
```

```
c=list(b)       #c:  [1,2,3]
```

```
t = (1, 3, 2)
```

```
(a, b, c) = t    # a = 1, b = 3, c = 2
```

```
s = [1,2,3]
```

```
[a,b,c] = s      # a = 1, b = 2, c = 3
```

元组、列表和字符串互转

```
print(list("hello"))          #>>['h', 'e', 'l', 'l', 'o']
```

```
print("".join(['a','44','c'])) #>>a44c
```

```
print(tuple("hello"))         #>>('h', 'e', 'l', 'l', 'o')
```

```
print("".join(('a','44','c'))) #>>a44c
```