# 基础

## 列表和元组

### 创建列表和元组

创建列表和元组的语法  
区别：  
 创建列表使用方括号  
 创建元组使用圆括号  
  
创建列表的语法格式如下:  
[ele1,ele2,ele3,...]  
创建元组的语法格式如下：  
(ele1,ele2,ele3,...)

*# 使用方括号定义列表*my\_list = [**'crazyit'**, 20, **'Python'**]  
print(my\_list)  
*# 使用圆括号定义元组*my\_tuple = (**'crazyit'**, 20, **'Python'**)  
print(my\_tuple)

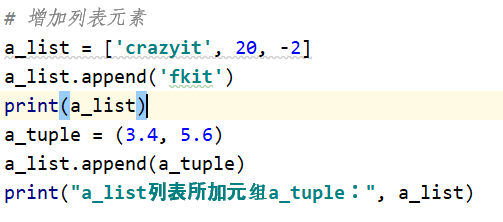
### 列表和元组通用方法

a\_tuple = (**'crazyit'**, 20, 5.6, **'fkit'**, -17)  
print(a\_tuple)  
print(**"访问元组第一个元素："**, a\_tuple[0])  
print(**"访问元组第二个元素："**, a\_tuple[1])  
print(**"访问元组倒数第一个元素:"**, a\_tuple[-1])  
print(**"访问元组倒数第二个元素"**, a\_tuple[-2])  
*# 子序列*print(**"访问第2个至第4个元素:"**, a\_tuple[2:4])  
print(**"访问倒数第3个至倒数第一个的元素:"**, a\_tuple[-3:-1])  
print(**"访问第2个至倒数第2个:"**, a\_tuple[1:-2])  
print(**"访问倒数第3个至第5个："**, a\_tuple[-3:4])  
b\_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)  
print(**"访问第3个至第9个元素，间隔为2:"**, b\_tuple[2:8:2])  
print(**"访问第3个至第9个，间隔为3:"**, b\_tuple[2:8:3])  
print(**"访问第3个至倒数第2个，间隔为2："**, b\_tuple[2:-2:2])  
print(**"+++++++++++++++加法+++++++++++++++"**)  
a\_tuple = (**'crazyit'**, 20, -1.2)  
b\_tuple = (127, **'crazyit'**, **'fkit'**, 3.33)  
*# 计算元组相加*sum\_tuple = a\_tuple + b\_tuple  
print(sum\_tuple)  
print(a\_tuple)  
print(b\_tuple)  
*# 两个元组相加*print(a\_tuple + (-20, -30))  
a\_list = [20, 30, 50, 100]  
b\_list = [**'a'**, **'b'**, **'c'**]  
*# 计算列表相加*sum\_list = a\_list + b\_list  
print(sum\_list)  
print(a\_list + [**'fkit'**])  
  
print(**"++++++++++++乘法++++++++++++"**)  
a\_tuple = (**'crazyit'**, 20)  
*# 执行乘法*mul\_tuple = a\_tuple \* 3  
print(mul\_tuple)  
a\_list = [30, **'Python'**, 2]  
mul\_list = a\_list \* 5  
print(mul\_list)  
print(**"++++++++长度、最大值和最小值+++++++++++++"**)  
*# 元素都是数值的元组*a\_tuple = (29, 49, 13, 22, 33, 10, 4, 102)  
print(**"元素都是数值，最大值："**, max(a\_tuple))  
print(**"元素都是数值，最小值:"**, min(a\_tuple))  
print(**"元素都是数值，长度:"**, len(a\_tuple))  
b\_tuple = (  
 **'crazyit'**, **'Codes'**, **'Hibernate'**, **'struts'**, **'Spring'**, **'Appium'**, **'Ruby'**, **'Seleinum'**, **'Java'**, **'C++'**, **'Python'**, **'fkit'**)  
print(**"元素都是字符串，最大值:"**, max(b\_tuple))  
print(**"元素都是字符串，最小值:"**, min(b\_tuple))  
print(**"元素都是字符串，长度:"**, len(b\_tuple))

### 创建列表

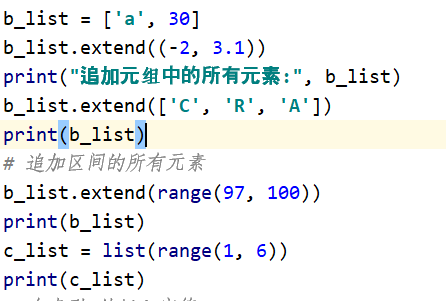
*# 创建列表*a\_tuple = (**'crazyit'**, 20, -1.2)  
print(**"将元组转换成列表:"**, list(a\_tuple))  
a\_range = range(1, 5)  
print(a\_range)  
b\_list = list(a\_range)  
print(**"将区间转换成列表:"**, b\_list)  
c\_list = list(range(4, 20, 3))  
print(c\_list)

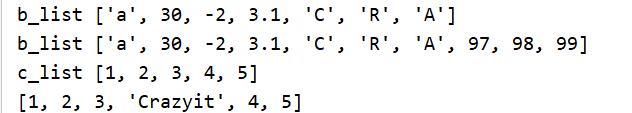
### 列表添加元素append()



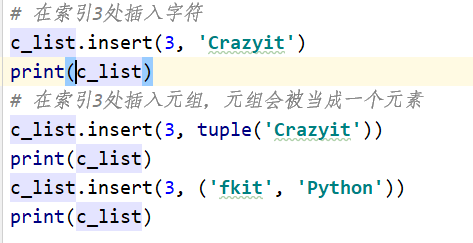


### 列表追加元素extend()

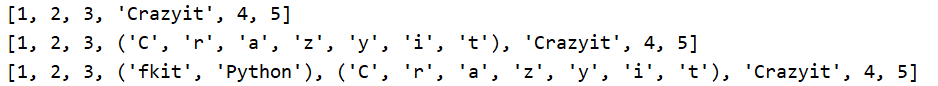




### 列表插入字符insert()

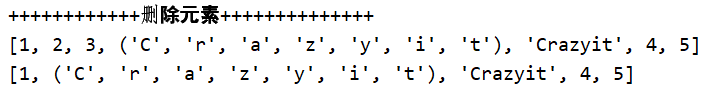


结果:

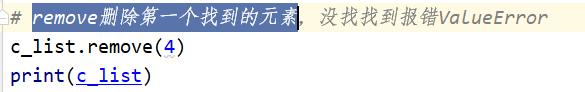


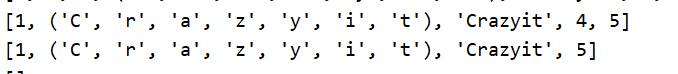
### 删除元素del

*# 删除元素*print(**"++++++++++++删除元素++++++++++++++"**)  
**del** c\_list[3]  
print(c\_list)  
**del** c\_list[1:3]  
print(c\_list)  
*# del c\_list 清空列表*



### remove删除第一个找到的元素





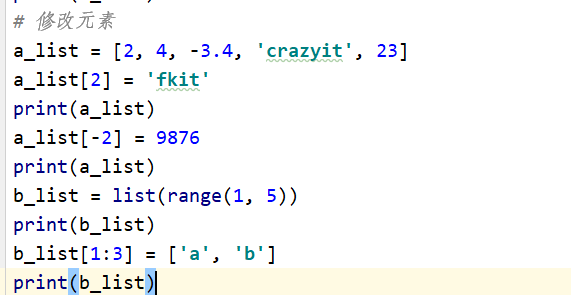
说明:第一列为删除前的元素列表;第二列为删除元素’4’之后的列表

### 清空列表clear()

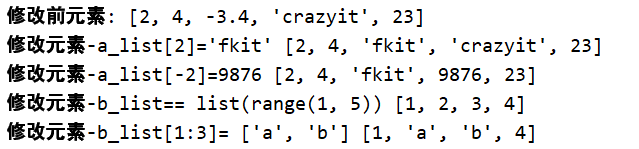
c\_list.clear()  
print(c\_list)

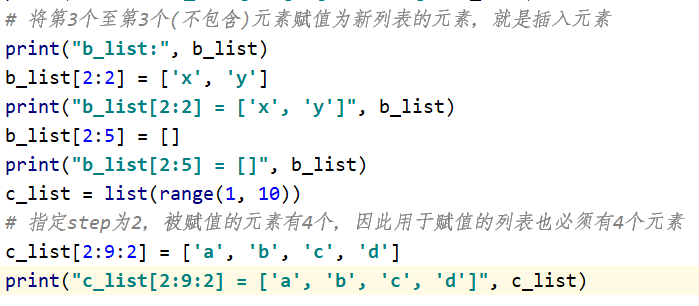


### 修改元素

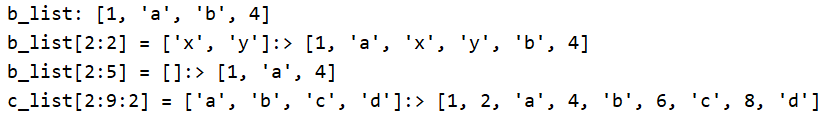


运行结果:



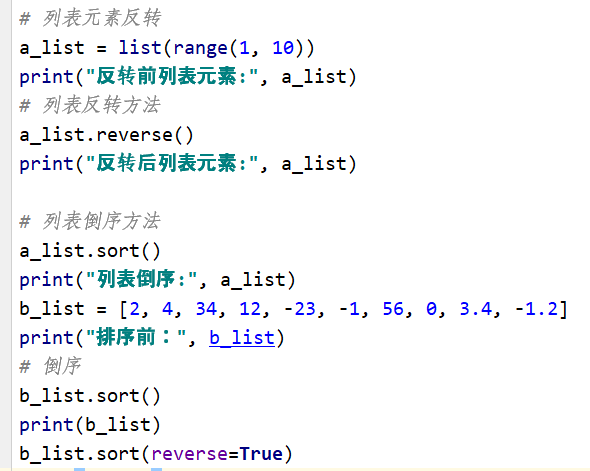


结果:

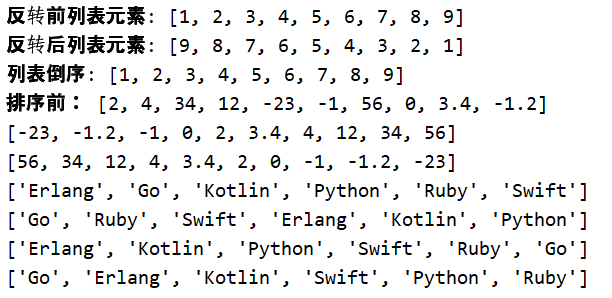


### 其他方法

count():用于统计列表中某个元素出现的次数  
index():用于判断某个元素在列表中出现的位置  
pop():用于寺将列表当成“栈”使用，实现元素出栈功能  
reverse():用于将列表元素反向存放  
sort():用于对列表元素排序



结果:



### reversed-反转序列

该方法是将列表,元组,区间进行序列反转

### sorted-排序,默认按照从小到大

reverse=True(从大到小排序)

c\_list.sort(key=len, reverse=True)

## 字典

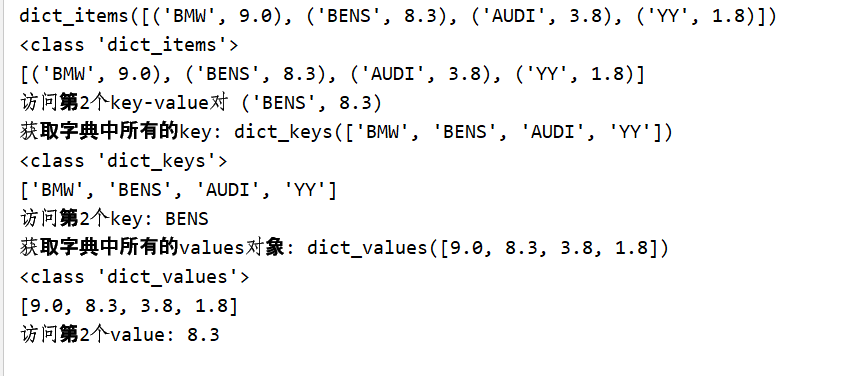
### 常用方法

* clear():用于清空字典中所有的key-value对
* get()方法:用来获取value

使用get()方法访问不存在的key时,不会报错,但返回None

* update():更新已有的key-value对,如果不存在,则进行添加
* items():用于获取字典中所有的key-value对,返回dict\_items
* keys():用于获取字典所有的的key,返回dict\_keys
* values():用于获取字典所有的value,返回dict\_values

ims = cars.items()  
print(ims)  
print(type(ims))  
print(list(ims))  
print(**"访问第2个key-value对"**, list(ims)[1])  
kys = cars.keys()  
print(**"获取字典中所有的key:"**, kys)  
print(type(kys))  
print(list(kys))  
*# 访问第2个key*print(**"访问第2个key:"**, list(kys)[1])  
vals = cars.values()  
print(**"获取字典中所有的values对象:"**, vals)  
print(type(vals))  
print(list(vals))  
print(**"访问第2个value:"**, list(vals)[1])



* pop():用于获取指定key对应的value,并删除这个key-value对
* popitem():用于随机弹出字典中的一个key-value对,并删除
* setdefault():用于根据key来获取value,如果key不存在,则设置一个默认的value并返回,然后再返回该value,如果存在则直接返回value
* fromkeys():使用给定的多个key创建字典,key对应的value都是None

## 流程控制

### If分支结构

第一种形式:

If expression :

Statement…

第二种形式:

If expression :

Statement…

else :

Statement…

第三种形式

If expresseion :

Statement…

elif expression :

Statements…

else :

statements…

说明:False,None,0,””,[],{},()会被当作False处理

Pass语句就是空语句,有时候程序需要一个位,放一条语句,但又不希望这条语句做任何事情,此时就可以通过pass语句来实现,通过pass语句,可以使程序更加完整.

### While循环循环

语法格式:

[init\_statements]

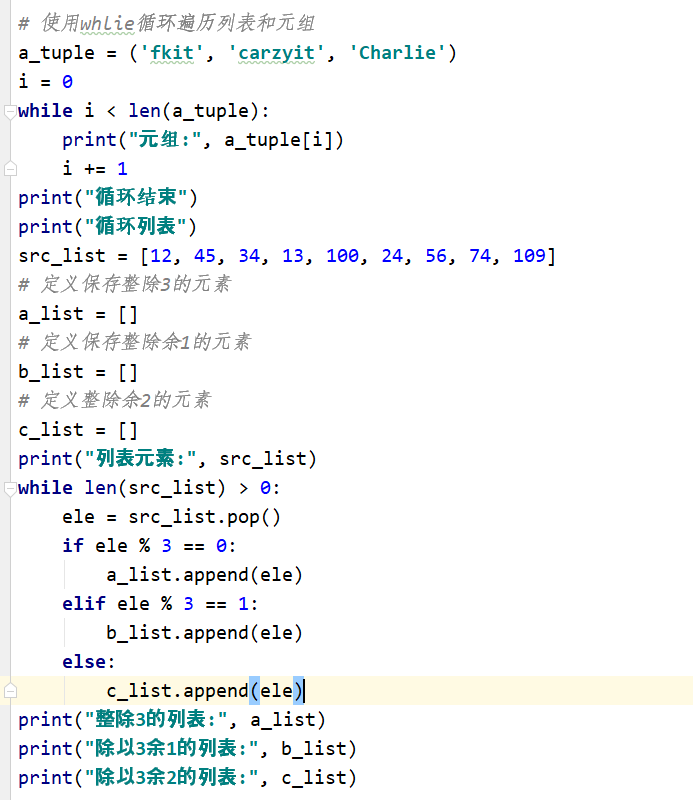
While test\_expression:

Body\_statements

[iteration\_statements]

count\_i = 0  
**while** count\_i < 10:  
 print(**"count\_i:"**, count\_i)  
 count\_i += 1  
print(**"循环结束"**)

循环遍历列表和元组



### For-in循环

s\_max = input(**"请输入您想计算的阶乘:"**)  
mx = int(s\_max)  
reslut = 1  
**for** num **in** range(1, mx + 1):  
 reslut \*= num  
print(reslut)

* for-in循环遍历列表和元组

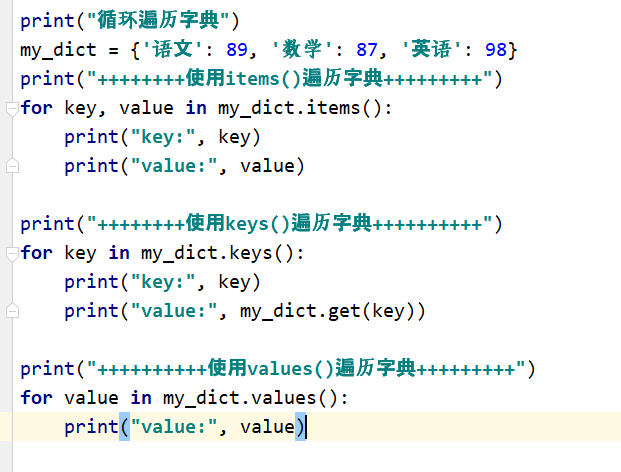
a\_tuple = (**'crazyit'**, **'fkit'**, **'Charlie'**)  
**for** ele **in** a\_tuple:  
 print(**"当前元素是:"**, ele)

#遍历数组

a\_list = [330, 1.4, 50, **'crazyit'**, -3.5]  
**for** num **in** range(0, len(a\_list)):  
 print(**"第%d个元素是:%s"** % (num, a\_list[num]))

* 遍历字典
* Items():返回字典所有key-value对的列表.
* Keys():返回字典中所有key的列表
* Values():返回返回字典所有value的列表

print(**"循环遍历字典"**)  
my\_dict = {**'语文'**: 89, **'数学'**: 87, **'英语'**: 98}  
print(**"++++++++使用items()遍历字典+++++++++"**)  
**for** key, value **in** my\_dict.items():  
 print(**"key:"**, key)  
 print(**"value:"**, value)  
  
print(**"++++++++使用keys()遍历字典++++++++++"**)  
**for** key **in** my\_dict.keys():  
 print(**"key:"**, key)  
 print(**"value:"**, my\_dict.get(key))  
  
print(**"++++++++++使用values()遍历字典+++++++++"**)  
**for** value **in** my\_dict.values():  
 print(**"value:"**, value)



### for表达式

格式:

[表达式 for 循环计数器 in 可迭代对象]

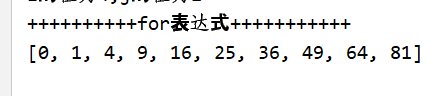
For表达式与普通for循环的区别:

* 在for关键字之前定义一个表达式,该表达式通常会包含循环计数器
* for表达式没有循环体,因此不需要冒号

for表达式最终返回的是列表,因此for表达式也被称为列表推导式.

print(**"++++++++++for表达式+++++++++++"**)  
a\_range = range(10)  
a\_list = [x \* x **for** x **in** a\_range]  
print(a\_list)

运行结果:



还可以在for表达式后面添加if条件,这样for表达式将只迭代那些符合条件的元素,例如:

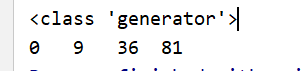
b\_list = [x \* x **for** x **in** a\_range **if** x % 2 == 0]  
print(b\_list)



For表达式使用圆括号,返回的是生成器,称为生成器推导式.

c\_generator = (x \* x **for** x **in** a\_range **if** x % 3 == 0)

print(type(c\_generator))  
**for** i **in** c\_generator:  
 print(i,end=”\t)



For表达式,for之前的为循环体,for之后的循环条件

### 常用工具函数zip

a = [**'a'**, **'b'**, **'c'**]  
b = [1, 2, 3]  
c = [x **for** x **in** zip(a, b)]  
print(c)  
print(type(c))

