# python3中tuple（元组）的常用方法示例

Python 的元组与列表类似，不同之处在于元组的元素不能修改。

元组使用小括号，列表使用方括号。

元组创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。

创建一个元组是将不同的逗号分隔值。也可以选择把这些逗号分隔值也括号内。例如

tup1 = ('physics', 'chemistry', 1997, 2000)  
tup2 = (1, 2, 3, 4, 5 )  
tup3 = "a", "b", "c", "d"

空元组写成不含任何值的两个空括号 -

tup1 = ()

为了编写含有一个单一的值，必须包含逗号，即使只有一个值的元组 –

tup1 = (50,)

类字符串索引，元组索引从0开始，并且它们可以被切片，联接等等。

## 访问元组的组：

要访问值元组，用方括号带索引或索引切片来获得可用的索引值。例如 -

tup1 = ('physics', 'chemistry', 1997, 2000)  
tup2 = (1, 2, 3, 4, 5, 6, 7 )  
print ("tup1[0]: ", tup1[0])  
print ("tup2[1:5]: ", tup2[1:5])

## 更新元组

元组是不可变的，这意味着我们不可以更新或更改元组元素的值。如下面的例子说明了可以把现有的元组创建新的元组的部分 -

tup1 = (12, 34.56)  
tup2 = ('abc', 'xyz')  
tup3 = tup1 + tup2  
print (tup3)

## 删除元组元素

移除个元组的别元素是不可能的。还有就是，当然无可厚非组建另一元组来排放的不需要的元素。

如要明确删除整个元组，只需要用 del 语句。 例如：

tup = ('physics', 'chemistry', 1997, 2000)  
print(tup)  
del tup  
print(tup)

注意引发异常，这是因为经过 del tup 元组就不再存在了

## 元组基本操作

元组响应 + 和 \* 操作就像字符串; 它们意味着串联和重复这里也一样，所不同的是，结果是一个新的元组，而不是字符串。

事实上，元组应对各种操作，我们在先前章字符串中使用顺序类似 -

|  |  |  |
| --- | --- | --- |
| **Python表达式** | **结果** | **描述** |
| len((1, 2, 3)) | 3 | 长度 |
| (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | 拼接 |
| ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | 重复 |
| 3 in (1, 2, 3) | True | 成员关系 |
| for x in (1,2,3) : print (x, end=' ') | 1 2 3 | 迭代 |

## 索引，切片和矩阵

由于元组是序列，索引和切片的工作方式与字符串所做的操作相同。假设以下的输入 -

T=('C++', 'Java', 'Python')

|  |  |  |
| --- | --- | --- |
| **Python表达式** | **结果** | **描述** |
| T[2] | 'Python' | 偏移从零开始 |
| T[-2] | 'Java' | 负值：从右边计数 |
| T[1:] | ('Java', 'Python') | 切片取部分 |

## 内置元组函数

Python包括以下元组函数 -

1.len(tuple) 计算元组元素个数。

tuple1 = ('Google', 'Runoob', 'Taobao')  
print(len(tuple1))

2. max(tuple) 返回元组中元素最大值。

tuple1 = ('5', '4', '8')  
print(max(tuple1))

3. min(tuple) 返回元组中元素最小值。

tuple1 = ('5', '4', '8')  
print(min(tuple1))

4. tuple(seq) 将列表转换为元组。

list1=[1,2,3,4]  
print(tuple(list1))

5.index(value) 获取value的索引位置

tuple1 = ('5', '4', '8')  
print(tuple1.index('5'))

6.count(value) 获取value在tuple出现的次数

tuple1 = ('a', 'b', 'c')  
print(tuple1.count('a'))

# [python3中set（集合）的常用方法示例](http://blog.csdn.net/zhongbeida_xue/article/details/51211506)

集合set是Python中一种基本数据类型，它分为可变集合（set）和不可变集合（frozenset）两种。类似于其他语言，集合是一个无序不重复元素集，包括创建集合set、向集合中添加元素、删除集合中的元素、求集合的交集、并集、差集等操作。下面就让我们一一来学习集合的这些知识吧。

**1、set创建集合**

set集合类需要的参数必须是迭代器类型的，如：序列、字典等，然后转换成无序不重复的元素集。由于集合是不重复的，所以可以对字符串、列表、元组进行去重操作。

**1.1创建空集合**

# 字符串  
print(set(''))  
# 列表  
print(set([]))  
# 元组  
print(set(()))

**1.2 创建非空集合**

集合中set括号中需要的参数的数据类型有：序列（包括字符串、列表、元组），字典或者是另一个集合，但是不能是数值类型，如int类型。

字符串做参数

print(set('createSet'))

list做参数

print(set([1, 2, 3, 1, 3, 4, 5]))

元组做参数

print(set((1, 2, 3, 4, 1)))

字典做参数

print(set({1: 2, 2: 3}))

字典转set集合，需要注意的是，只取了字典的key，相当于将字典中的dict.keys()列表转成set集合。

set集合做参数

print(set(set([1, 3, 2, 3, 4, 5])))

**2、集合添加**

集合的添加有两种方式，分别是add和update。但是它们在添加元素时是有区别的：

**2.1 add()方法**

是把要传入的元素作为一个整体添加到集合中，如：

s = set('one')  
print(s)  
s.add('two')  
print(s)

**2.2 update()方法**

是把要传入的元素拆分成单个字符，存于集合中，并去掉重复的字符。如：

s = set('one')  
print(s)  
s.add('two')  
print(s)  
s.update('132')  
print(s)

**3、集合删除**

集合的删除操作使用的方法跟列表是一样的，使用的也是remove方法。如：

s = set('one')  
print(s)  
s.add('two')  
print(s)  
s.update('o32')  
print(s)  
s.remove('2')  
print(s)

**4、集合的遍历**   
集合的遍历跟序列的遍历方法完全一样。

st = set([1, 2, 'a', 'c', 4, 'd'])  
for i in st:  
 print(i)

另一种遍历方式：

st = set([1, 2, 'a', 'c', 4, 'd'])  
for idx, i in enumerate(st) :  
 print(idx,i)

变量idx表示集合中元素i的索引。

**5、集合与序列间转换**

集合可以和序列之间的转换跟序列之间互转是一样的，唯一不同的就是序列转成集合后，重复的元素被去掉了。

**5.1 集合与字符串转换**

字符串转集合

s = set('python')  
print(s)

集合转字符串

s = set('python')  
print(s)  
print(str(s))

需要注意的是，使用str()函数直接对集合转字符串，结果是集合的字符串形式。   
我们也可以使用字符串的连接函数join()函数，进行集合转字符串，但是这样得到的字符串种的元素也是无序的。

s = set('python')  
set(['h', 'o', 'n', 'p', 't', 'y'])  
print(''.join(s))

**5.2 集合与list转换**

list转set

list1 = [1,2,3,3,4,5]  
print(set(list1))

set转list

sList={1, 2, 3, 4, 5}  
list2 = list(sList)  
print(list2)

集合转list结果是得到一个去重后的list，这种去重比用之前讲过的利用字典key唯一和利用list自身特定去重的方法都要简单快捷。

**5.3 集合转元组**

tuple转set

tup = (1, 2, 3, 3, 45, 6)  
sTup = set(tup)  
print(sTup)

set转tuple

sTup={1, 2, 3, 6, 45}  
tup1 = tuple(sTup)  
print(tup1)

set转tuple后，得到的元组也是去重后的元组

**6、集合其他常用方法**

**6.1 discard()**   
函数原型：   
setVar.discard(element)

参数说明：   
setVar :为一个set类型的变量   
element :表示要查找并删除的元素   
函数作用：

在集合setVar中查找element元素，如果存在则删除；如果没找到，则什么也不做。

sTup = {1, 2, 3, 6, 6, 45}  
sTup.discard(1)  
print(sTup)

**6.2 pop()方法**

函数原型：   
s.pop()   
参数说明：   
s：为set类型的变量   
函数作用：   
删除并返回set类型的s中的一个不确定的元素，如果为空引发KeyError错误。

sTup = {1, 2, 3, 6, 6, 45}  
print(sTup.pop())  
print(sTup)

函数原型：   
s.clear()   
参数说明：   
s：set类型的变量   
函数作用：   
清空s集合中的所有元素

sTup = ([1, 2, 3, 6, 6, 45])  
sTup.clear()  
print(sTup)

**7、集合的一些操作符**

既然是集合，那就会遵循集合的一些操作方法，如求交集、并集、差集等。   
**7.1 交集**   
Python中求集合的交集使用的符号是“&”，返回连个集合的共同元素的集合，即集合的交集。

st1 = set('python')  
st2 = set('htc')  
print(st1 & st2)

**7.2 并集（合集）**   
Python中求集合的并集用的是符号“|”，返回的是两个集合所有的并去掉重复的元素的集合。

st1 = set('python')  
st2 = set('htc')  
print(st1 | st2)

**7.3 差集**   
Python中差集使用的符号是减号“-”。

st1 = set('python')  
st2 = set('htc')  
print(st1 - st2)

返回的结果是在集合st1中但不在集合st2中的元素的集合。

**7.4 集合的不同**   
查看两个集合的不同之处，使用的difference函数，等价于差集。如：   
**s1.difference(s3)**   
这种不同指的是集合s3相对于集合s1，不同的地方，也就是所有在集合s1中，而不再集合s2中的的元素组成的新集合。

s1 = set([1, 2, 3, 4, 5])  
s2=set([1, 2, 3, 4])  
print(s1.difference(s2))

**8、集合的范围判断**

集合可以使用大于（>）、小于（<）、大于等于（>=）、小于等于（<=）、等于（==）、不等于（！=）来判断某个集合是否完全包含于另一个集合，也可以使用子父集判断函数。   
定义三个集合s1，s2，s3：

大于（>）或大于等于（>=）

s1 = set([1, 2, 3, 4, 5])  
s2=set([1, 2, 3, 4])  
s3=set(['1', '8', '9', '5'])  
print(s1>s2)  
print(s1>=s3)

表示左边集合是否完全包含右边集合，如集合s1是否完全包含集合s2。   
小于（<）或 小于等于（<=）

s1 = set([1, 2, 3, 4, 5])  
s2=set([1, 2, 3, 4])  
s3=set(['1', '8', '9', '5'])  
print(s1<s2)  
print(s1<=s3)

表示左边的集合是否完全包含于右边的集合，如集合s1是否完全包含于集合s2。

等于（==）、不等于（！=）

s1 = set([1, 2, 3, 4, 5])  
s2=set([1, 2, 3, 4])  
s3=set(['1', '8', '9', '5'])  
print(s1==s2)  
print(s1!=s3)

判断两个集合是否完全相同。

**9、集合的成员运算符**

集合里也可以使用成员运算符，in和not in，判断某个对象是否是集合中的成员。

s1 = set([1, 2, 3, 4, 5])  
print(1 in s1)  
print(1 not in s1)

**10、集合拷贝**

集合的拷贝函数是：copy()

s1 = set([1, 2, 3, 4, 5])  
s = s1.copy()  
print(s)

将集合s1拷贝一份并赋给变量sc。

**11、求集合长度**

跟序列一样，求集合的长度也是使用len()函数。

s1 = set([1, 2, 3, 4, 5])  
print(len(s1))

**12、求集合的关系**

**12.1 a.issuperset(b)**   
判断集合a是否是b的父集。

s1 = set([1, 2, 3, 4])  
s2 = set([1, 2, 3, 4, 5])  
print(s2.issuperset(s1))

判断s1集合是否是集合s2的父集，是返回True，否则返回False。

**12.2 b.issubset(a)**   
判断b是否是a的子集。

s1 = set([1, 2, 3, 4])  
s2 = set([1, 2, 3, 4, 5])  
print(s1.issubset(s2))

判断s2集合是否是集合s1的子集，是返回True，否则返回False。

**不可变集合frozenset**

Python中还有一种不可改变的集合，那就是frozenset，不像set集合，可以增加删除集合中的元素，该集合中的内容是不可改变的，类似于字符串、元组。

f = frozenset()  
print(f)  
f = frozenset('asdf')  
print(f)  
f = frozenset([1, 2, 3, 4])  
print(f)  
f = frozenset((1, 2, 3, 4))  
print(f)  
f = frozenset({1: 2, 'a': 2, 'c': 3})  
print(f)

如果试图改变不可变集合中的元素，就会报AttributeError错误。   
不可变集合，除了内容不能更改外，其他功能及操作跟可变集合set一样。   
**注意：**   
无论是可变集合还是不可变集合，参数除了支持字符、数字、还可以是汉字，类等。

# Python3中简单的文件操作

**前言：**

首先介绍一下什么叫做相对路径和绝对路径

**相对路径：**

相对路径是相对于文件当前的工作路径而言的

**绝对路径：**

绝对路径是由文件名和它的完整路径以及驱动器字母组成的，如果是Windows系统，那么某一个文件的绝对路径可能是:

c:\pythonworkspace\firstpy.py

**文件类型：**

文件大概可以分为文本文件和二进制文件。在不同操作系统下，可以用文本编辑器编辑的文件，都称为文本文件，那么其他的文件就属于二进制文件。而二进制文件相比与文本文件的优势在于二进制文件的处理效率更高一些。

**读取文件的开始**

读取一个文件的思路永远都是相同的，第一步自然就是打开一个文件。在python中我们通过如下代码使用open函数来打开一个文件。

input = open(filepath,mode)

我们的mode主要由以下几种方式。

|  |  |
| --- | --- |
| **模式** | **描述** |
| r | 打开一个文件为只读。文件指针被放置在文件的开头。这是默认模式。 |
| rb | 打开一个文件只能以二进制格式读取。文件指针被放置在文件的开头。这是默认模式。 |
| r+ | 打开用于读和写文件。文件指针置于该文件的开头。 |
| rb+ | 打开用于读取和写入二进制格式的文件。文件指针置于该文件的开头。 |
| w | 打开一个文件只写。如果文件存在覆盖该文件。如果该文件不存在，则创建写入新文件。 |
| wb | 打开一个文件只能以二进制格式写入。如果文件存在覆盖该文件。如果该文件不存在，则创建写入新文件。 |
| w+ | 打开文件为写入和读取模式。如果文件存在覆盖现有文件。如果该文件不存在，创建用于读写操作的新文件。 |
| wb+ | 打开用于以二进制格式写入和读出文件。如果文件存在覆盖现有文件。如果该文件不存在，创建用于读写操作的新文件。 |
| a | 打开用于追加的文件。文件指针是在文件是否存在该文件的末尾。也就是说，该文件是在追加模式。如果该文件不存在，它会创建一个用于写入的新文件。 |
| ab | 打开文件用于二进制格式追加。文件指针是在文件是否存在该文件的末尾。也就是说，文件是在追加模式。 如果该文件不存在，它会创建一个用于写入的新文件。 |
| a+ | 打开文件为追加和读取方式。文件指针是在文件是否存在该文件的末尾。该文件以追加模式打开。如果该文件不存在，它将创建用于读写操作的新文件。 |
| ab+ | 打开一个文件以附加和二进制格式读取模式。如果该文件存在文件指针在该文件的末尾。该文件以追加模式打开。如果该文件不存在，它将创建读写操作的新文件。 |

同样我们有两种途径来打开文件。

通过绝对路径

open(**'** F:\project\test\login.txt**'**)

通过相对路径（需要注意的是，我们通过相对路径是可以打开当前工作目录下的文件的，也就是说如果我的.py文件存在F:\project下的话，我们通过相对路径打开的文件也同样存在当前路径下）

open(**'../login.txt'**,**'r'**)

**注意**

在Windows下我们通过绝对路径来打开文件的时候，我们需要在绝对文件名前加上一个 r 前缀，来表示这个字符串是一个行字符串，这样可以让python解释器将文件中的反斜线理解成字面意义上的反斜线。例如：

open(r**'** F:\project\test\login.txt**'**)

如果我们不添加 r 作为前缀，则需要用转义字符把上面的语句修改成如下所示：

open(**'** F:\\project\\test\\login.txt**'**)

读取内容方式

f.read() # 一次读取整个文件，文件大不适用

f.readline() # 一次只读取一行，占内存小，速度慢

f.readlines() # 一次性读取，将内容分析成一个行的列表，可以由for...in...处理

写文件方式

f.write(content) # 不会换行哦

f.writelines(content) # 下次会写在下一行

f.close() # 关闭文件

f **=** open(**'login.txt'**, **'r'**) *# 以读方式打开文件*result **=** list()   
**for** line **in** f.**readlines**()**:** *# 依次读取每行* line **=** line.**strip**() *# 去掉每行头尾空白* **if not** len(line) **or** line.**startswith**(**'#'**)**:** *# 判断是否是空行或注释行* **continue** *# 是的话，跳过不处理* result.**append**(line) *# 保存*result.**sort**() *#排序结果*print(result)  
f.**close**() *# 关闭文件*

**with** open(**'new\_file.txt'**,**'w'**) **as** fw**:** # with方式不需要再进行close  
 **for** x **in** result**:** fw.**write**(**'%s' % '-'**.**join**(x)) *# 保存入结果文件*

如果读取txt文件出现乱码怎么办

**import** codecs  
**with** codecs.**open**(**'login.txt'**,**'w'**,**'utf8'**) **as** f**:** f.**write**(**'第三方士大夫'**)

**文件位置**

*#encoding:utf-8***import** codecs  
  
fo **=** codecs.**open**(**"login.txt"**, **'r'**,**'utf8'**)  
str **=** fo.**read**(10)  
print (**"Read String is : "**, str)  
  
*# tell() 方法告诉你该文件中的当前位置；换句话说，下一个读或写将发生在从该文件的开头之后多个字节*position **=** fo.**tell**()  
print (**"Current file position : "**, position)  
  
*# seek(offset[, from]) 方法改变当前文件的位置。 offset 参数指示要移动字节数。from 参数指定字节从哪移动的参考位置。*position **=** fo.**seek**(10, 0)  
str **=** fo.**read**()  
print (**"Again read String is : "**, str)  
fo.**close**()

判断中文占用的字节数大小

**import** sys  
print(sys.**getsizeof**(**'是'**))

**重命名和删除文件**

os.rename(current\_file\_name, new\_file\_name)：rename()方法有两个参数，当前文件名和新的文件名

**import** os  
os.**rename**( **"login.txt"**, **"test2.txt"** )

os.remove(file\_name)：可以使用 remove()方法通过提供参数文件名称(file\_name)来删除文件

**import** os  
os.**remove**(**'test2.txt'**)

**Python目录**

os.mkdir("newdir")：可以使用os模块中的 mkdir()方法在当前目录下创建目录。需要提供一个参数到这个方法指定要创建的目录名。

**import** os  
os.**mkdir**(**"newdir"**)

os.chdir("newdir")：可以使用 chdir() 方法来改变当前目录。chdir() 方法接受一个参数，就是你想在当前目录创建的目录的目录名称。

os.getcwd()显示当前的工作目录。

**import** os  
print(os.**getcwd**())

os.rmdir('dirname')删除目录

**import** os  
os.**rmdir**( **"newdir"** )