Python tuple 元组详解

Tuple 是不可变 list。 一旦创建了一个 tuple 就不能以任何方式改变它。

Tuple 与 list 的相同之处

定义 tuple 与定义 list 的方式相同, 除了整个元素集是用小括号包围的而不是方括号。

Tuple 的元素与 list 一样按定义的次序进行排序。 Tuples 的索引与 list 一样从 0 开始, 所以一个非空 tuple 的第一个元素总是 t[0]。

负数索引与 list 一样从 tuple 的尾部开始计数。

与 list 一样分片 (slice) 也可以使用。注意当分割一个 list 时, 会得到一个新的 list ；当分割一个 tuple 时, 会得到一个新的 tuple。

Tuple 不存在的方法

您不能向 tuple 增加元素。Tuple 没有 append 或 extend 方法。

您不能从 tuple 删除元素。Tuple 没有 remove 或 pop 方法。

您不能在 tuple 中查找元素。Tuple 没有 index 方法。

然而, 您可以使用 in 来查看一个元素是否存在于 tuple 中。

用 Tuple 的好处

Tuple 比 list 操作速度快。如果您定义了一个值的常量集，并且唯一要用它做的是不断地遍历它，请使用 tuple 代替 list。

如果对不需要修改的数据进行 “写保护”，可以使代码更安全。使用 tuple 而不是 list 如同拥有一个隐含的 assert 语句，说明这一数据是常量。如果必须要改变这些值，则需要执行 tuple 到 list 的转换。

Tuple 与 list 的转换

Tuple 可以转换成 list，反之亦然。内置的 tuple 函数接收一个 list，并返回一个有着相同元素的 tuple。而 list 函数接收一个 tuple 返回一个 list。从效果上看，tuple 冻结一个 list，而 list 解冻一个 tuple。

Tuple 的其他应用

一次赋多值

>>> v = ('a', 'b', 'e')

>>> (x, y, z) = v

解释：v 是一个三元素的 tuple, 并且 (x, y, z) 是一个三变量的 tuple。将一个 tuple 赋值给另一个 tuple, 会按顺序将 v 的每个值赋值给每个变量。

python dictionary

(2010-08-30 08:46:41)

|  |  |
| --- | --- |
| 标签：  [杂谈](http://search.sina.com.cn/?c=blog&q=%D4%D3%CC%B8&by=tag) | 分类： [Python](http://blog.sina.com.cn/s/articlelist_1775414535_6_1.html) |

字典（Dictionary）是一种映射结构的数据类型，由无序的“键－值对”组成。字典的键必须是不可改变的类型，如：字符串，数字，tuple；值可以为任何python数据类型。  
  
  
1、新建字典  
>>> dict1={}      #建立一个空字典  
>>> type(dict1)  
<type 'dict'>  
  
2、增加字典元素：两种方法  
>>> dict1['a']=1    #第一种  
>>> dict1  
{'a': 1}  
#第二种：setdefault方法  
>>> dict1.setdefault('b',2)  
2  
>>> dict1  
{'a': 1, 'b': 2}  
  
3、删除字典  
#删除指定键－值对  
>>> dict1  
{'a': 1, 'b': 2}  
>>> del dict1['a']      #也可以用pop方法，dict1.pop('a')  
>>> dict1  
{'b': 2}  
#清空字典  
>>> dict1.clear()  
>>> dict1         #字典变为空了  
{}  
#删除字典对象  
>>> del dict1  
>>> dict1  
Traceback (most recent call last):  
  File "<interactive input>", line 1, in <module>  
NameError: name 'dict1' is not defined  
  
4、字典的方法  
1)get(key,default=None)     
返回键值key对应的值；如果key没有在字典里，则返回default参数的值，默认为None  
  
>>> dict1         #空的字典  
{}  
>>> dict1.get('a')   #键‘a’在dict1中不存在，返回none  
>>> dict1.get('d1','no1')   #default参数给出值'no1'，所以返回'no1'  
'no1'  
>>> dict1['a']='no1'      #插入一个新元素  
>>> dict1  
{'a': '1111'}  
>>> dict1.get('a')      #现在键'a'存在，返回其值  
'1111'  
  
2)clear  
清空字典  
  
3)has\_key(key)  
如果key出现在dict里则返回True；否则返回False  
>>> dict1  
{'a': '1111'}  
>>> dict1.has\_key('b')  
False  
>>> dict1.has\_key('a')  
True  
  
4)items  
返回dict的（键，值）tuple对的一个列表  
>>> dict1  
{'a': 'no1', 'b': '2222'}  
>>> dict1.items()  
[('a', 'no1'), ('b', '2222')]  
  
5)keys   返回dict的键列表  
6)values 返回dict的值列表  
>>> dict1  
{'a': 'no1', 'b': '2222'}  
>>> dict1.keys()  
['a', 'b']  
>>> dict1.values()  
['no1', '2222']  
  
7)setdefault(key,default=None)  
如果dict中有key，则返回key值，如果没有找到key，则在dict中加上该key，值由default参数给出，默认None  
  
8)update(dict2)  
把dict2的元素加入到dict中去，键字重复时会覆盖dict中的键值  
>>> dict2  
{'c': '3333', 'b': 'no2'}  
>>> dict1                  #dict2和dict1的键‘b'重复  
{'a': 'no1', 'b': '2222'}  
>>> dict1.update(dict2)     #调用update后，dict1的键'b'值被覆盖了  
>>> dict1  
{'a': 'no1', 'c': '3333', 'b': 'no2'}  
  
9)popitem  
删除任意键－值对，并返回该键－值对，如字典为空，则产生异常  
>>> dict1  
{'b': 'no2'}  
>>> dict1.popitem()  
('b', 'no2')  
>>> dict1  
{}  
>>> dict1.popitem()  
Traceback (most recent call last):  
  File "<interactive input>", line 1, in <module>  
KeyError: 'popitem(): dictionary is empty'  
  
10)pop(key,[d])  
删除指定键字的键－值对，并返回该键对应的值   ＃第二个参数不知道怎么用  
>>> dict1  
{'a': 'no1', 'c': '3333', 'b': 'no2'}  
>>> dict1.pop('a')  
'no1'  
>>> dict1  
{'c': '3333', 'b': 'no2'}  
  
11)copy  
返回字典的一个浅拷贝  
  
#以下方法目前还不知道怎么用  
12)fromkeys  
13)iteritems  
14)iterkeys  
15)itervalues