python数据类型

Python中常见的数据结构可以统称为容器（container）。序列（如列表和元组）、映射（如字典）以及集合（set）是三类主要的容器。

**一、序列（列表、元组和字符串）**

序列中的每个元素都有自己的编号。Python中有6种内建的序列。其中列表和元组是最常见的类型。其他包括字符串、Unicode字符串、buffer对象和xrange对象。下面重点介绍下列表、元组和字符串。

**1、列表**

列表是可变的，这是它区别于字符串和元组的最重要的特点，一句话概括即：列表可以修改，而字符串和元组不能。

**（1）、创建**

通过下面的方式即可创建一个列表：

|  |  |
| --- | --- |
| 1  2  3  4 | list1=['hello','world']  print(list1)  list2=[1,2,3]  print(list2) |

输出：  
['hello', 'world']  
[1, 2, 3]

可以看到，这中创建方式非常类似于javascript中的数组。

**（2）、list函数**

通过list函数（其实list是一种类型而不是函数）对字符串创建列表非常有效：

|  |  |
| --- | --- |
| 1  2 | list3=list("hello")  print(list3) |

输出：

['h', 'e', 'l', 'l', 'o']

**2、元组**

元组与列表一样，也是一种序列，唯一不同的是元组不能被修改（字符串其实也有这种特点）。

**（1）、创建**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | t1=1,2,3  t2="jeffreyzhao","cnblogs"  t3=(1,2,3,4)  t4=()  t5=(1,)  print(t1,t2,t3,t4,t5) |

输出：

(1, 2, 3) ('jeffreyzhao', 'cnblogs') (1, 2, 3, 4) () (1,)

从上面我们可以分析得出：

a、逗号分隔一些值，元组自动创建完成；

b、元组大部分时候是通过圆括号括起来的；

c、空元组可以用没有包含内容的圆括号来表示；

d、只含一个值的元组，必须加个逗号（,）；

**（2）、tuple函数**

tuple函数和序列的list函数几乎一样：以一个序列（注意是序列）作为参数并把它转换为元组。如果参数就算元组，那么该参数就会原样返回：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | t1=tuple([1,2,3])  t2=tuple("jeff")  t3=tuple((1,2,3))  print(t1)  print(t2)  print(t3)  t4=tuple(123)  print(t45) |

输出：

(1, 2, 3)  
('j', 'e', 'f', 'f')  
(1, 2, 3)

Traceback (most recent call last):  
  File "F:\Python\test.py", line 7, in <module>  
    t4=tuple(123)  
TypeError: 'int' object is not iterable

**3、字符串**

**（1）创建**

|  |  |
| --- | --- |
| 1  2  3  4  5 | str1='Hello world'  print(str1)  print(str1[0])  for c in str1:      print(c) |

输出：  
Hello world  
H  
H  
e  
l  
l  
o  
   
w  
o  
r  
l  
d

**（2）格式化**

字符串格式化使用字符串格式化操作符即百分号%来实现。

|  |  |
| --- | --- |
| 1  2 | str1='Hello,%s' % 'world.'  print(str1) |

输出：

Hello,world.

格式化操作符的右操作数可以是任何东西，如果是元组或者映射类型（如字典），那么字符串格式化将会有所不同。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | strs=('Hello','world') #元组  str1='%s,%s' % strs  print(str1)  d={'h':'Hello','w':'World'} #字典  str1='%(h)s,%(w)s' % d  print(str1) |

输出：

Hello,world  
Hello,World

注意：如果需要转换的元组作为转换表达式的一部分存在，那么必须将它用圆括号括起来：

|  |  |
| --- | --- |
| 1  2 | str1='%s,%s' % 'Hello','world'  print(str1) |

输出：

Traceback (most recent call last):  
  File "F:\Python\test.py", line 2, in <module>  
    str1='%s,%s' % 'Hello','world'  
TypeError: not enough arguments for format string

如果需要输出%这个特殊字符，毫无疑问，我们会想到转义，但是Python中正确的处理方式如下：

|  |  |
| --- | --- |
| 1  2 | str1='%s%%' % 100  print(str1) |

输出：100%

对数字进行格式化处理，通常需要控制输出的宽度和精度：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | from math import pi  str1='%.2f' % pi #精度2  print(str1)  str1='%10f' % pi #字段宽10  print(str1)  str1='%10.2f' % pi #字段宽10，精度2  print(str1) |

输出：

3.14  
  3.141593  
      3.14

字符串格式化还包含很多其他丰富的转换类型，可参考官方文档。

Python中在string模块还提供另外一种格式化值的方法：模板字符串。它的工作方式类似于很多UNIX Shell里的变量替换，如下所示：

|  |  |
| --- | --- |
| 1  2  3  4 | from string import Template  str1=Template('$x,$y!')  str1=str1.substitute(x='Hello',y='world')  print(str1) |

输出：

Hello,world!

如果替换字段是单词的一部分，那么参数名称就必须用括号括起来，从而准确指明结尾：

|  |  |
| --- | --- |
| 1  2  3  4 | from string import Template  str1=Template('Hello,w${x}d!')  str1=str1.substitute(x='orl')  print(str1) |

输出：

Hello,world!

如要输出$符，可以使用$输出：

|  |  |
| --- | --- |
| 1  2  3  4 | from string import Template  str1=Template('$x$$')  str1=str1.substitute(x='100')  print(str1) |

输出：100$

除了关键字参数之外，模板字符串还可以使用字典变量提供键值对进行格式化：

|  |  |
| --- | --- |
| 1  2  3  4  5 | from string import Template  d={'h':'Hello','w':'world'}  str1=Template('$h,$w!')  str1=str1.substitute(d)  print(str1) |

输出：

Hello,world!

除了格式化之外，Python字符串还内置了很多实用方法，可参考官方文档，这里不再列举。

**4、通用序列操作（方法）**

从列表、元组以及字符串可以“抽象”出序列的一些公共通用方法，这些操作包括：索引（indexing）、分片（sliceing）、加（adding）、乘（multiplying）以及检查某个元素是否属于序列的成员。除此之外，还有计算序列长度、最大最小元素等内置函数。

**（1）索引**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | str1='Hello'  nums=[1,2,3,4]  t1=(123,234,345)  print(str1[0])  print(nums[1])  print(t1[2]) |

输出

H  
2  
345

索引从0（从左向右）开始，所有序列可通过这种方式进行索引。索引也可以从最后一个位置（从右向左）开始，右边第一位索引值为-1：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | str1='Hello'  nums=[1,2,3,4]  t1=(123,234,345)  print(str1[-1])  print(nums[-2])  print(t1[-3]) |

输出：

o  
3  
123

**（2）分片**

分片操作用来访问一定范围内的元素。分片通过冒号相隔的两个索引来实现：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | nums=range(10)  print(nums)  print(nums[1:5])  print(nums[6:10])  print(nums[1:])  print(nums[-3:-1])  print(nums[-3:]) #包括序列结尾的元素，置空最后一个索引  print(nums[:]) #复制整个序列 |

输出：

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[1, 2, 3, 4]  
[6, 7, 8, 9]  
[1, 2, 3, 4, 5, 6, 7, 8, 9]  
[7, 8]  
[7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

不同的步长，有不同的输出：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | nums=range(10)  print(nums)  print(nums[0:10])  #默认步长为1 等价于nums[1:5：1]  print(nums[0:10:2])  #步长为2  print(nums[0:10:3])  #步长为3    ##print nums[0:10:0])  #步长为0  print(nums[0:10:-2])  #步长为-2 |

输出：

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[0, 2, 4, 6, 8]  
[0, 3, 6, 9]  
[]

**（3）序列相加**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | str1='Hello'  str2=' world'  print(str1+str2)  num1=[1,2,3]  num2=[2,3,4]  print(num1+num2)  print(str1+num1) |

输出：

Hello world  
[1, 2, 3, 2, 3, 4]

Traceback (most recent call last):  
  File "F:\Python\test.py", line 7, in <module>  
    print (str1+num1)  
TypeError: cannot concatenate 'str' and 'list' objects

**（4）乘法(数乘)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | print([None]\*10)  str1='Hello'  print(str1\*2)  num1=[1,2]  print(num1\*2)  print(str1\*num1) |

输出：

[None, None, None, None, None, None, None, None, None, None]

HelloHello  
[1, 2, 1, 2]

Traceback (most recent call last):  
  File "F:\Python\test.py", line 5, in <module>  
    print(str1\*num1)  
TypeError: can't multiply sequence by non-int of type 'list'

**（5）成员资格**

in运算符会用来检查一个对象是否为某个序列（或者其他类型）的成员（即元素）：

|  |  |
| --- | --- |
| 1  2  3  4  5 | str1='Hello'  print('h' in str1)  print('H' in str1)  num1=[1,2]  print(1 in num1) |

输出：

False  
True  
True

**（6）长度、最大最小值**

通过内建函数len、max和min可以返回序列中所包含元素的数量、最大和最小元素。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | str1='Hello'  print(len(str1)  print(max(str1)  print(min(str1)  num1=[1,2,1,4,123]  print(len(num1)  print(max(num1)  print(min(num1) |

输出：

5  
o  
H  
5  
123  
1

**二、映射（字典）**

映射中的每个元素都有一个名字，如你所知，这个名字专业的名称叫键。字典（也叫散列表）是Python中唯一内建的映射类型。

**1、键类型**

字典的键可以是数字、字符串或者是元组，键必须唯一。在Python中，数字、字符串和元组都被设计成不可变类型，而常见的列表以及集合（set）都是可变的，所以列表和集合不能作为字典的键。键可以为任何不可变类型，这正是Python中的字典最强大的地方。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | list1=["hello,world"]  set1=set([123])  d={}  d[1]=1  print(d)  d[list1]="Hello world."  d[set1]=123  print(d) |

输出：

{1: 1}

Traceback (most recent call last):  
  File "F:\Python\test.py", line 6, in <module>  
    d[list1]="Hello world."  
TypeError: unhashable type: 'list'

**3、自动添加**

即使键在字典中并不存在，也可以为它分配一个值，这样字典就会建立新的项。

**4、成员资格**

表达式item in d（d为字典）查找的是键（containskey），而不是值（containsvalue）。

**三、集合**

集合（Set）在Python 2.3引入，通常使用较新版Python可直接创建，如下所示：

strs=set(['jeff','wong','cnblogs'])

nums=set(range(10))

看上去，集合就是由序列（或者其他可迭代的对象）构建的。集合的几个重要特点和方法如下：

**1、副本是被忽略的**

集合主要用于检查成员资格，因此副本是被忽略的，如下示例所示,输出的集合内容是一样的。

|  |  |
| --- | --- |
| 1  2  3  4  5 | set1=set([0,1,2,3,0,1,2,3,4,5])  print(set1)    set2=set([0,1,2,3,4,5])  print(set2) |

输出如下：

set([0, 1, 2, 3, 4, 5])  
set([0, 1, 2, 3, 4, 5])

**2、集合元素的顺序是随意的**

这一点和字典非常像，可以简单理解集合为没有value的字典。

|  |  |
| --- | --- |
| 1  2 | strs=set(['jeff','wong','cnblogs'])  print(strs) |

输出如下：

set(['wong', 'cnblogs', 'jeff'])

**3、集合常用方法**

a、union操作返回两个集合的并集，不改变原有集合。使用按位与（OR）运算符“|”可以得到一样的结果：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | set1=set([1,2,3])  set2=set([2,3,4])  set3=set1.union(set2)  set4=set1|set2  print(set1)  print(set2)  print(set3)  print(set4) |

输出：

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

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

其他常见操作包括&（交集）,<=,>=,-,copy()等等，这里不再列举。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | set1=set([1,2,3])  set2=set([2,3,4])  set3=set1&set2  print(set1)  print(set2)  print(set3)  print(set3.issubset(set1)  set4=set1.copy()  print(set4)  print(set4 is set1) |

输出如下：

set([1, 2, 3])  
set([2, 3, 4])  
set([2, 3])  
True  
set([1, 2, 3])  
False

b、add和remove

和序列添加和移除的方法非常类似，可参考官方文档：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | set1=set([1])  print(set1)  set1.add(2)  print(set1)  set1.remove(2)  print(set1)  print(set1)  print(29 in set1)  set1.remove(29) #移除不存在的项 |

输出：

set([1])  
set([1, 2])  
set([1])  
set([1])  
False

Traceback (most recent call last):  
  File "F:\Python\test.py", line 9, in <module>  
    set1.remove(29) #移除不存在的项  
KeyError: 29

**4、frozenset**

集合是可变的，所以不能用做字典的键。集合本身只能包含不可变值，所以也就不能包含其他集合：

|  |  |
| --- | --- |
| 1  2  3 | set1=set([1])  set2=set([2])  set1.add(set2) |

输出如下：

Traceback (most recent call last):  
  File "F:\Python\test.py", line 3, in <module>  
    set1.add(set2)  
TypeError: unhashable type: 'set'

可以使用frozenset类型用于代表不可变（可散列）的集合：

|  |  |
| --- | --- |
| 1  2  3  4 | set1=set([1])  set2=set([2])  set1.add(frozenset(set2))  print(set1) |

输出：

set([1, frozenset([2])])