链滴

python 学习 --- 数据结构（一）

作者：oldcaptain

原文链接：https://ld246.com/article/1374308206928

来源网站：链滴

 这篇文章简单的说一下python的数据结构。我也是初学者有些地方可能不够深入。
<h1> <a>简介 </h1>
      我们这里提到的数据结构为将数据以某种结构存储，便于我们的程序够很好的处理。如果你学过数据结构，或者了解java。你将会知道常见的数据结构有：

  数组（Array）&bsp;   特点：1、一旦在内存中请求建立空间后，分配的空间大小不能调整，否则会出现数据溢的情况。2、具有数据连续性的表现，中间的数据不能随意删除修改。（需要下标来控制）3、序在运行的时候不会检查数组下标，会存在数组越界的风险。

       注 数组：优点是插入快，如果知道下标，可以非常快地存取。缺点是找慢，删除慢，大小固定。有序数组：优点是比无序的数据查找快。缺点是删除和插入慢，大小固定。

  堆栈 （Stack）nbsp; 特点：1、只能允许在堆栈的一端进行推入，弹出操作的（线性的）。优点是提供后进先出方的存取。缺点是存取其他项很慢。
  队列（queue）nbsp; 特点：1、先进先出，只允许在后端进行插入操作，在前端进行删除操作。2、操作方式和堆栈似，但是队列只允许在后端插入数据。提供先进先出方式的存取。缺点是存取其他项很慢。
  链表 （list） &nsp;   特点：1、可以动态开辟空间。2、和数组相比不能随机读取，每一次读取数据都要循环迭代，时间复杂度较高。优点是插入快，删除快。缺点是查找慢
  树   （tree）   特点：1、每个节点有0个多给子节点。2、没有前驱的结点称为根结点。3、每一个非根结点有且只有一个父结点。除了根结点，每个子结点可以分为m个不相交的子树。关于树延伸起来就很多，各种数据的查找，删除，操作。里就不多说了
  图   （Graph）特点：1、优点是对现实世界建模。缺点是有些算法且复杂
  堆   （heap）   特点：1、堆中某个节点的总是不大于或不小于其父节点的值；2、堆总是一棵完全树nbsp;优点是插入、删除快，对最大数据项的存取很快。缺点是对其他数据项存取慢。
  散列表<a> （hash） 特点：1、是根据关键字(Key value)而直接进行访问的数据结构。优点是如果关键字知则存取极快。插入快。缺点是删除慢，如果不知道关键字则存取很慢，对存储空间使用不充分

<h2> python的数据结构： </h2>
<h3> 1.列表 </h3>
 上面说了一些常见的数据结构的一些特点，以及优缺点。接下来看看python中的数据结构。我从数据的存储，以及输出来分析
<pre>>>> shoplist = ["apple","mango","carrot""banana"]
>>> print len(shoplist)
4
>>> for item in shoplist:
    print item,

apple mango carrot banana

>>> shoplist.sort()

>>> print shoplist

['apple', 'banana', 'carrot', 'mango']

>>> shoplist.append("
ice")

>>> print shoplist

['apple', 'banana', 'carrot', 'mango', 'rice']

>>> del shoplist[0]

>>> print shoplist

['banana', 'carrot', 'mango', 'rice']

>>> </pre>

 在python中把第一行定义的这种数据结构叫做列表。一旦你创建了一个<a>列表你可以添加、删除或是搜索<a>列表中的项目。由于你可以增加或删除项目，我们说<a>表是 可变的 数据类型，即这种类型是可以被改变的。（这里是不是很灵活，可以通过标来访问，删除数据。并且可以增加数据。）
 可以通过命令help(list)来查看python对list实了那些方法。
<h3>
 2、元组

 </h3>
<pre>>>> zoo = ("wolf","elephant","penguin")
>>> print len(zoo)
3
>>> new_zoo = ("monkey","dolphin",zoo)
>>> print new_zoo[2]
('wolf', 'elephant', 'penguin')
>>> print new_zoo[2][2]
penguin
>>> </pre>

 变量zoo是一个<a>元组，我们看到len函数可以用来获取<a>元组的长度。也表明<a>元组也是一个序列。我们在new_zoo 中放置了zoo，但是zoo还是通过下标可以访的。这说明元组在元组中是不会失去身份的。
 空的元组用 zoo=(),但是一个元素的元组应该用zo=("animail",)，需要在后面跟一个逗号。
<h3> 3、字典 </h3>
<pre>>>> ab ={ "swaroop" : "hello@swaroop.com",
    "larry" : "larry@larry.org",
    "matsumoto" : "matz@matsuoto.com",
    }
>>> print ab["swaroop"]
hello@swaroop.com
>>>
>>> del ab["swaroop"]
>>> len(ab)
2
>>> for name,addressin ab.items():

SyntaxError: invalid syntax

>>> for name,address in ab.items() :

print (name,address)

('matsumoto', 'matz@matsuoto.com')

('larry', 'larry@larry.org')

>>> </pre>

  这种字典数据结构类似于json的数据格式，通过key/value键值对的方式来操作数据。可以使用help(dict)来查看dict类的完整方法列表

<h3> 4、使用序列 </h3>
<blockquote>
 列表、元组和字符串都是<a>序列，但是<a>序列是什么，它们为什么如此特别呢<a>序列的两个主要特点是索引操作符和切片操作。索引操作符让我们可以从<a>序列中抓取一个特定项目。切片操作符让我们能够获取<a>序列的一个切片，即一部分<a>序
</blockquote>
<pre>>>> shoplist = ['apple', 'mango', 'carrot', 'banana']
>>> print shoplist[1:3]
['mango', 'carrot']
>>> print shoplist[1:-1]
['mango', 'carrot']
>>> print shoplist[:]
['apple', 'mango', 'carrot', 'banana']
>>> </pre>

 首先，我们来学习如何使用索引来取得<a>序列中的单个项目。这也被称作是下标操作。当你用方括号中的一个数来指定一个<a>序列的时候，Python会为你抓取<a>序列中对位置的项目。记住，Python从0开始计数。因此，shoplist[0]抓取第一个项目，shoplist[3]抓取shoplist<a>序列中的第四个元素。
 索引同样可以是负数，在那样的情况下，位置是从<a>序列尾开始计算的。因此，shoplist[-1]表示<a>序列的最后一个元素而shoplist[-2]抓取<a>序列的倒数第二个项目。<a>序列的神奇之处在于你可以用相同的方法访问元组、列表和字符串
<h3> 4、对象与参考 </h3>
 当你创建一个对象并给它赋一个变量的时候，这个变量仅仅 <a>参考 那个象，而不是表示这个对象本身！也就是说，变量名指向你计算机中存储那个对象的内存。这被称作名到对象的绑定。
<pre>>>> shoplist = ['apple', 'mango', 'carrot', 'banana']
>>> mylist = shoplist
>>> del shoplist[0]
>>> print shoplist
['mango', 'carrot', 'banana']
>>> print mylist
['mango', 'carrot', 'banana']
>>> mylist = shoplist[:]
>>> del mylist[0]
>>> print shoplist
['mango', 'carrot', 'banana']
>>> print mylist
['carrot', 'banana']</pre>

 你需要记住的只是如果你想要复制一个列表或者类似的序列或者其他复杂的对（不是如整数那样的简单 对象 ），那么你必须使用切片操作符来取得拷贝。如果你只想要使用另一个变量名，两个名称都 <a>参考 同一个对象，那么如果你不小心话，可能会引来各种麻烦。

<h3> 5、字符串的方法 </h3>
<pre>>>> name ="guopeng"

```
&gt;&gt;&gt; if name.startswith('guo') :
    print 'yes'

yes
&gt;&gt;&gt; if 'u' in name :
print 'yes'

yes
&gt;&gt;&gt; if name.find('uop') != -1 :
print 'yes'

yes
&gt;&gt;&gt; delimiter = &quot;                                                *&quot;
&gt;&gt;&gt; mylist = [&quot;guo
quot;,&quot;pem&quot;,&q
ot;indei&quot;]
&gt;&gt;&gt; print delimiter.join(mylist)
guo_pem_indei
&gt;&gt;&gt;                                </pre>

<p>   </p>
<p>   </p>
```