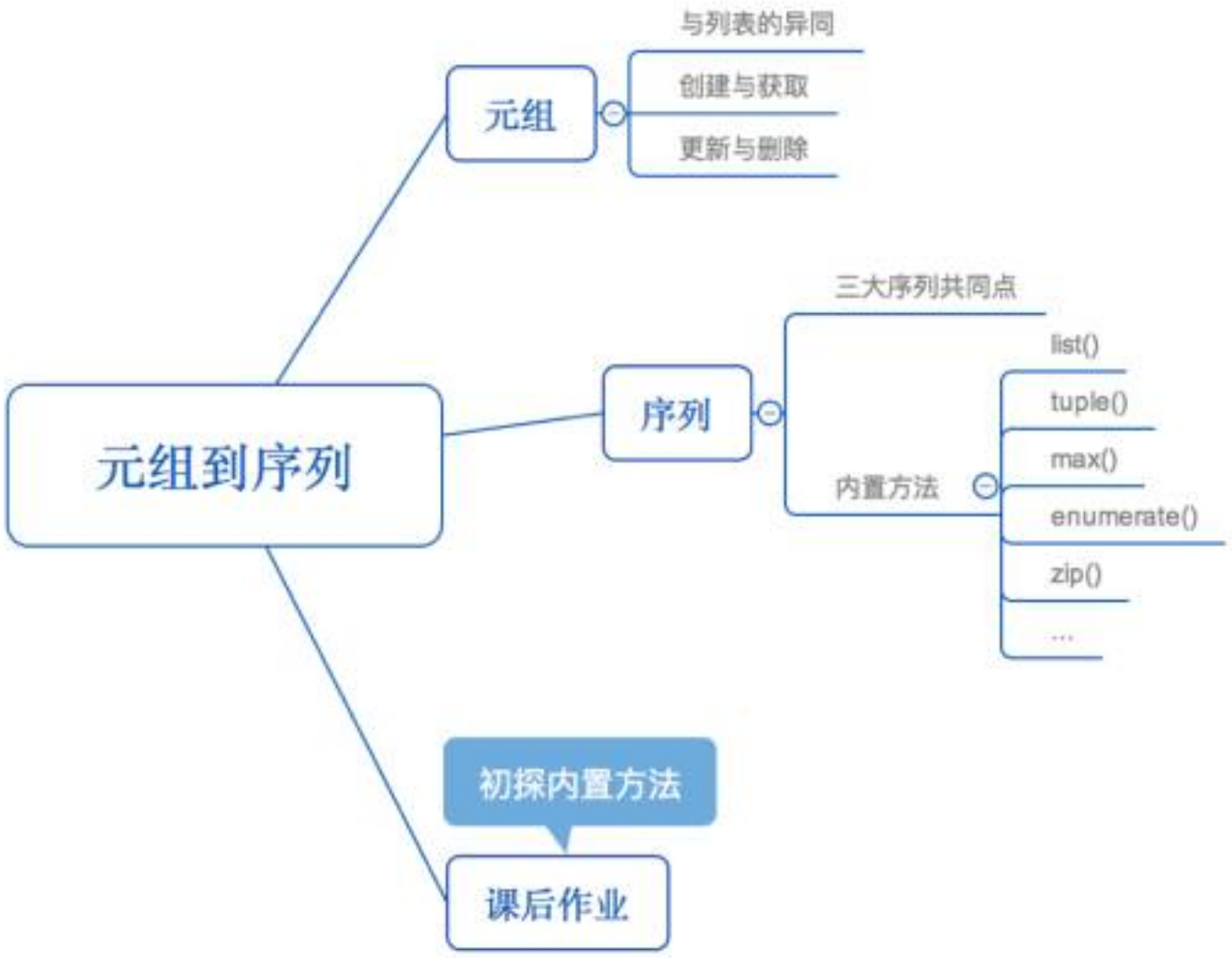
Python第四课（下）－自列表被“囚禁”后，让我知道原来你是这样的序列

Original 2016-10-23 哈哈小 菜猿观世界

加关注这种话银家怎么好意思说出口嘛

通过本节的学习，你将了解以下内容：



1

“囚禁”的列表－元组（tuple）

通过上节课对列表的讲解，我们有了解到列表的强势，列表可谓无所不能，无所不纳，权大实强。

“一切有权力的人都容易滥用权利，这是万古不易的一条经验。有权力的人们使用权利一直遇到有界限的地方才休止”，列表的权利如此之大，这样不好，广大群众表示不满。

为响应民意，我们在这非常有必要“把权利关进笼子里”，因为只有这样，才能防止专制和腐败的蔓延，公民的权利才能得到保证啊，哈哈～

于是乎，被削权囚禁的列表，“沦为”元组。

1）与列表异同

通过上文的描述，元组与列表的异同，想必大家都能猜出一二。哈哈小，一言以蔽之：

同为序列，异为权限。

同为序列，意思是它具备序列的一些共性，后文会提到。异为权限，我们都已经知道，列表增删改查，无所不能，而元组却是不可被修改的。

与列表的区别，看完全文，相信了然于心。

2）创建与获取

讲了这么多，也没见到元组真容。

别急，这就让我们看看元组长什么样？

元组的创建：

```
>>> tuple1=(1,2,3,'3',[1,2],0)
>>> tuple1
(1, 2, 3, '3', [1, 2], 0)
>>>
```

似乎可以看出，元组也是无所不纳。整型，字符串，元组，列表等等好像都能往里放。俨然还是个大杂缸，真是缸性难移啊～

这里要提出一个问题，我们都知道，世间万物，皆有成其之的殊性或标志。就好比，哈哈哈标志着哈哈小就要出场，黑暗标志着黎明的即将到来，太阳雨标志着彩虹的可能到来，哈哈～

中括号可能就标志着是python列表那老伙计，看到单双引号可能立马想到字符串，那么元组的标志会是什么呢？机灵鬼可能立马回答道：这个答案很明显，是小括号。

不好意思，答错了。举几个例子：

```
>>> tuple2=(5)
>>> type(tuple2)
<class 'int'>
>>> tuple2
5
>>> tuple2=5,3,4
>>> type(tuple2)
<class 'tuple'>
>>> tuple2
(5, 3, 4)
>>> tuple2=(5,)
>>> tuple2
(5,)
>>> tuple2=()
>>> tuple2
()
>>> type(tuple2)
<class 'tuple'>
>>>
```

通过对比，好像小括号并不决定元组，有无小括号都能创建一个元组。显而易见，元组的标志是逗号。如果你非要带括号创建只有一个元素的元组，以tuple2=(5,)这种方式也是可行的。上述有一个值得注意的例外是：如果要创建一个空元组，tuple2=()即可，无需逗号符。

元组的获取：

讲过了字符串和列表，元组的获取不能再简单了，直接索引与切片获取即可。

```
>>> tuple1=(1,2,3,4,5,6)
>>> tuple1[5]
6
>>> tuple1[0]
1
>>> tuple1[3]
4
>>> tuple1[2:]
(3, 4, 5, 6)
>>> tuple1[:]
(1, 2, 3, 4, 5, 6)
>>>
```

so easy, get done~

2）更新与删除

等等，之前哈哈小好像讲过，元组已经被削权，不再有被修改的本领。这“更新与删除”，这是要搞事情啊。

好吧，来看看哈哈小能不能搞出点事情来：

```
>>> caiyuan=('桑桑','秋词','疏桐')
>>> caiyuan=caiyuan[:1]+('哈哈小',)+caiyuan[1:]
>>> caiyuan
('桑桑', '哈哈小', '秋词', '疏桐')
>>>
```

哦，原来是这么个更新法啊。caiyuan元组被切成两段caiyuan[:1]与caiyuan[1:]，中间插入('哈哈小',)，通过拼接把最终结果再次赋给caiyuan元组，这样的话caiyuan把上一次定义好的元组覆盖，输出结果，达到伪改元组的目的。

我们再来看看删除：

```
>>> caiyuan
('桑桑', '哈哈小', '秋词', '疏桐')
>>> del caiyuan
>>> caiyuan
Traceback (most recent call last):
  File "<pyshell#51>", line 1, in <module>
    caiyuan
NameError: name 'caiyuan' is not defined
>>>
```

好好的caiyuan元组通过使用关键字被呜呜地删除了，再去寻找caiyuan时，已经发现python给我们报了一个错：name 'caiyuan' **is not** defined，意思就是已经找不到caiyuan元组了。

这里哈哈小要强调的是，我们很少以这样一种方式去删除一个元组。因为对于python编译器而言，它存在一种回收机制，意思就是当一个变量闲置（没有标签贴在它身上时），因为python会每隔一段时间去检测标签贴附情况，所以一旦发现这种闲置变量，python将立马把它回收（删除）。

序列

继上一节caiyuan元组被删之后，哈哈小准备让它在这一节重生。

序列，很陌生的一个名字，好像是初次见面。下面掌声有请序列给我们来一个详细的自我介绍。

下文序列言：

大家好，我叫序列，英文名叫sequence，像之前你们所了解的字符串，列表，元组，可以说，我是他们中的任何一个。通过对他们的了解，你们应该已经对我在python中的重要地位有所感知了。

1）三大序列共同点

虽然你们已经对我的列表，元组，字符串有所认识，但我还是希望你们能不厌烦地听我讲讲他们的共同点：

都可以通过索引或切片的方式得到其中的元素

默认情况下，索引值总是从0开始的

大部分的操作符，像重复操作符，成员关系操作符，拼接操作符等等都是可以适用的

好吧，共同点我能想到的就这么多，接下来我想隆重介绍一下我的一些比较好用的内置方法

2）内置方法

第一个是list，诶，这方法有一点忘了怎么用，尴尬，我来查查看：

```
>>> help(list)
Help on class list in module builtins:
class list(object)
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable's items
...
```

哦，有两种方法，第一个list（）不带任何参数，作用是new empty list（创建一个空列表）。

第二个是list（iterable），带了一个奇怪的叫做iterable的参数，作用是把一个可迭代对象转换成列表。iterable这个我再熟悉不过，叫迭代，因为我序列生来就是可迭代的。所以大家不用当心当你传入一个字符串或元组时，python编译器会给你报错。

大家不防跟我来试一下：

```
>>> choubuyaolian_string='hahaxiao is a 666 boy'
>>> list(choubuyaolian_string)['h', 'a', 'h', 'a', 'x', 'i', 'a', 'o', ' ', 'i', '
s', ' ', 'a', ' ', '6', '6', '6', ' ', 'b', 'o', 'y']
>>> choubuyaolian_tuple=(['hahaxiao','is'],'a',666,'boy')
>>> list(choubuyaolian_tuple)[['hahaxiao', 'is'], 'a', 666, 'boy']
>>>
```

没错，像choubuyaolian_string字符串，choubuyaolian_tuple元组都是可迭代对象，作为参数传入list函数，最终转换为成一个列表打印出来。

如果这个对象是字符串，这个方法将把字符串所包含的字符一个一个迭代地取出来再封装成列表，最后打印出来，上例一结果大家一目了然。

如果这个对象是元组，这就比较暴力了，转换成列表直接打印出来，上例二结果大家一目了然。

除了我的这个list方法，它还有两个好兄弟，分别叫做str和tuple

他们发挥的功能与list相似：

str（iterable）：将一个可迭代对象转换为字符串

tuple（iterable）：将一个可迭代对象转换为元组

```
>>> choubuyaolian_list=[['hahaxiao','is'],'a',666,'boy']
>>> str(choubuyaolian)
"[['hahaxiao', 'is'], 'a', 666, 'boy']"
>>> tuple(choubuyaolian)
(['hahaxiao', 'is'], 'a', 666, 'boy')
>>>
```

大家应该有发现，通过这三个方法，可以实现，字符串，元组，列表三大序列之间的转换。

此外，在认识我之前，想必大家对python交互式解析器的强大已经有所了解（别问我是怎么知道的，哈哈小昨天告诉我的），直接可以用来当作计算器，做基本的数学运算。但是请注意“基本”这两个字，下面就给大家介绍我的几款好用的跟数学处理相关的内置方法，看能不能把这个“基本”提升到一个层次。他们分别是len(),min(),max(),sum(),sorted()。

这些函数都有什么用呢?

大家从这些函数的名字上应该也能生出点意来

len方法，返回序列中元素的个数，大家前面一定多次接触到过

max方法，返回序列或参数集合的最大值

min方法，返回序列或参数集合的最小值

sum方法的语法是：sum(iterable[,start=0])，start参数为可选且默认值是0，返回序列和可选参数start的总和。大家可以自行试一下sum([1,3,4],4)，看看输出结果会是什么。

sorted方法，返回序列的元素由小到大排列的列表

针对这些方法给出几个样例：

```
>>> max(2,3,4,1,2)
4
>>> example_list=[2,3,-11,33,2,0]
>>> max(example_list)
33
>>> min(example_list)
-11
>>> len(example_list)
6
>>> example_string='12345580'
>>> max(example_string)
'8'
>>> example_tuple=(1,2,3,4,5,5,4,3,2,1)
>>> sum(example_tuple)
30
>>> sorted(example_tuple)
[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]
```

在这里非常有必要提一下下面这种特殊情况

```
>>> special_example='1a2b3d'
>>> max(special_example)
'd'
>>>
```

打印的结果为字符'd'，何故?

传入的参数为一个字符串，我们把它看作由'1','a','2','b','3','d'这六个字符组成的字符串。

我们都知道，所有的信息，包括字母，数字，音乐，图片，视频等等，在计算机底层都是以0与1的形式存储

的。就以视频为例，他是由一种叫流的东西（stream）组成，播放器的功能就是按着一定的规则对这些流做一个转化（也就是我们常说的编码），最终变成我们看到视频的那个样子。

这些流又是什么东东呢？其实就是一长串一长串字符组成，以一个字母a为例，这个字母最后存放到计算机内存上肯定也是0与1的形式，但它貌似跟数字不沾边啊，看来又得来一次编码了？

编码无非就是找到某种对应（或者你也可以说映射），这些字母字符啊，每一个都与一个特定的数配对，有了数字，换成二进制数那就容易多了。比如下面这个编码方式：

ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

高四位 / 低四位		ASCII控制字符 0000						ASCII控制字符 0001						ASCII打印字符 0010		ASCII打印字符 0011		ASCII打印字符 0100		ASCII打印字符 0101		ASCII打印字符 0110		ASCII打印字符 0111		
		0						1						2		3		4		5		6		7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	*Backspace 代码：DEL

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。 2013/08/08

这种编码方式叫做ASCII码，由图可找到小写字母a对应的十进制ascall值是97，同时也可以找到他其他字符的ASCII码值。

在python中，同c语言，当比较字符值时，都是采用这种编码方式进行计算的。所以也就不奇怪`max(special_example_string)`的输出值为d了（比较的是这些字符的ASCII值，可以看出d的ASCII码值为100，明显是最大的）。

怎么样，有了我这些函数的存在，有没有对python的数据处理能力倍增自信。

最后，再给大家讲两个好玩的方法，压箱底的东西都拿出来跟大家分享。

一个是enumerate函数－枚举，生成由每个元素的index值和item值组成的元组。

表示并不知道我在说什么，好吧，给你举个例子：

```
>>> example_enumerate=[4,3,2,1,0]
>>> list(enumerate(example_enumerate))
[(0, 4), (1, 3), (2, 2), (3, 1), (4, 0)]
>>>
```

第二个是zip方法－打包，返回的是由各个参数的序列组成的元组。

表示更加不知道你在说什么，好吧，再给你一例：

```
>>> zip1=[6,5,4,5,3,2,1]
>>> zip2=[1,2,3]
>>> list(zip(zip1,zip2))
[(6, 1), (5, 2), (4, 3)]
>>>
```

通过我这么长篇累牍的介绍，相信大家对我已经有所了解了。也希望在以后的学习过程中，能跟大家👬友好和平相处。

最后总结成一句话，那就是：作为python的一员，我的作用可是不容小视的哦。

下面把时间交给哈哈小，谢谢大家～

好的，sequence不仅给我们带来了详细的自我介绍，还给我们补充了ASCLL码知识点，希望大家能输入与输出并用。再次用热烈的掌声感谢序列毫无保留的介绍，哈哈～

课后作业

哈哈小认为，一切客观题都是阻碍思想走向自由独立的劲敌。

所以，今天的课后作业是。

我们前文用到的内置方法，都是python已经预定义好的，拿过来直接就能用。

大家有没有想过这些方法的具体实现呢?

以max(tuple1) 为例，哈哈小会想到它大概的实现过程会是这个样子的：

```
tuple1=(1,2,3,4,8,4,3,2,1)
max=tuple1[0]

for each in tuple1:
    if each>max:
        max=each

print("max is %d"%max)
```

好的，哈哈小已经做了一个实例了，其他的，像list(str2)，min(list1), len(str1), sum(tuple1)，sorted(list2)等等，大家赶紧去尝试吧。

要说明的是，这里只是让大家把这些方法的具体实现初步地展现出来，想哈哈小给的实例一样，以所学内容用代码大致地描绘出来。因为这些方法的完整实现会牵扯到函数的很多知识，想返回值，参数啊等等，关于函数的讲解哈哈小会在下期进入。

结束语：我们不想给你带来多少多少的知识，我们只想尽绵薄之力给你带来可能的智慧的启迪。

END