python核心编程

# 1 开篇词

hello，大家好，Python 是这个时代最流行、也必须要掌握的编程语言之一。Python 可以运用在数据处理、Web 开发、人工智能等多个领域，它的语言简洁、开发效率高、可移植性强。

因此，我决定开设这门课程，从工程的角度去讲解 Python 这门编程语言，我会从实际出发，以工作中遇到的实例为主线，去讲解 Python 的核心技术和应用。

本课程的所有内容都基于 Python 最新的 3.7 版本，使用pycharm作为IDE，其中有大量独家解读、案例。

# 2 列表与元组

## 2.1 基本概念

对于每一门编程语言来说，数据结构都是其根基。了解掌握 Python 的基本数据结构，对于学好这门语言至关重要。今天我们就一起来学习，Python 中最常见的两种数据结构：列表（list）和元组（tuple）。

首先，我们需要弄清楚最基本的概念，什么是列表和元组呢？实际上，列表和元组，都是一个可以放置任意数据类型的有序集合。

其中，有两个关键词：任意数据类型和有序

在绝大多数编程语言中，集合的数据类型必须一致。不过，对于 Python 的列表和元组来说，并无此要求。

lt = [1, 2, 'hello', 'world'] # "列表"中同时含有int和string类型的元素

tup = ('jason', 22) # "元组"中同时含有int和string类型的元素

其次，我们必须掌握它们的区别。

列表是动态的，长度大小不固定，可以随意地增加、删减或者改变元素（mutable）。而元组是静态的，长度大小固定，无法增加删减或者改变（immutable）。

下面的例子中，我们分别创建了一个列表与元组。你可以看到，对于列表，我们可以很轻松地让其最后一个元素，由 4 变为 40；但是，如果你对元组采取相同的操作，Python 就会报错，原因就是元组是不可变的。

2.1-1.py

另外，列表和元组都可以随意嵌套：

lt = [[1, 2, 3], [4, 5]] # 列表的每一个元素也是一个列表

tup = ((1, 2, 3), (4, 5, 6)) # 元组的每一个元素也是一个元组

当然，两者也可以通过 list() 和 tuple() 函数相互转换：

list((1, 2, 3))

tuple([1, 2, 3])

## 2.2 引用还是拷贝

1. =

在 python 中**赋值语句**总是建立对象的引用值，而不是复制对象。因此，python 变量更像是指针，本质上是是“标签”，是“引用”，而不是数据存储区域。当执行如下代码时，

a = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

Python 做的事情是首先创建一个“列表对象”，然后给它贴上名为 a 的标签。

[2.2-1.py]

这段代码中，用同一个列表分别赋值给变量a和b，那么这两条语句都会输出True吗？从打印结果来看，虽然变量a和变量b的值相等，但是分别指向不同的对象。

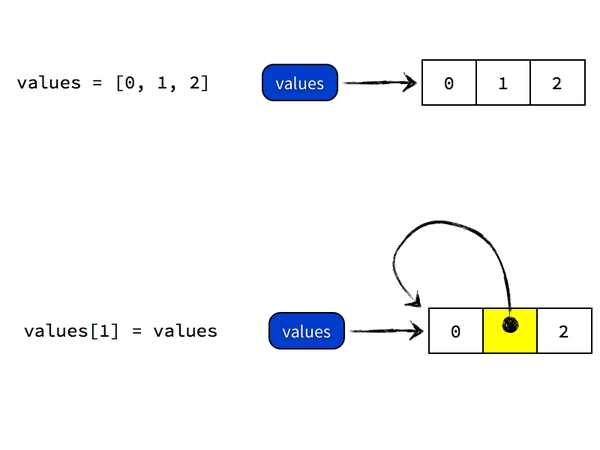
[2.2-2.py]

请看这段，values是期望的值吗？看来事实并非如此

这是为什么呢？

[2.2-2.png]

因为执行这条赋值语句时，Python 做的事情则是把 values 这个标签所引用的列表对象的第二个元素指向 values 所引用的列表对象本身，如图所示：



[2.2-3.py]

要想得到希望的结果（[0, [0, 1, 2], 2]），使用values[:]把 [0, 1, 2] 「复制」一遍，得到一个新对象，再将 values[1] 指向这个复制后的对象。

1. 浅复制与深复制

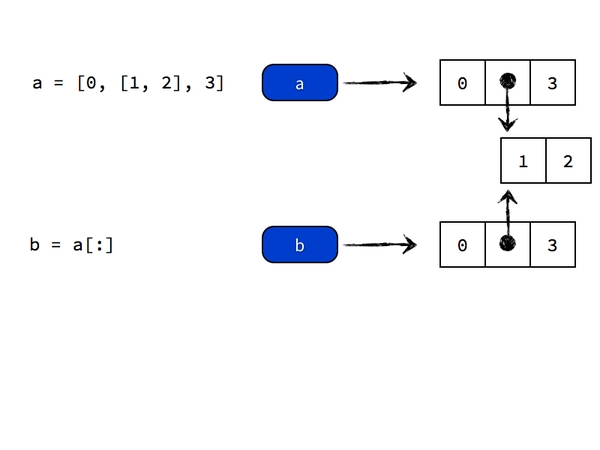
但是，values[:]是所谓的“浅复制”，当列表对象有嵌套的时候也会产生出乎意料的错误。

[v2.2-4]

来看这段代码，执行这两条赋值语句后，变量b的值会有变化吗？从输出结果来看，变量b的内嵌列表的值居然也发生了改变。这是为什么？

因为在b = a[:]执行完毕后的数据结构如图所示，所以b的内嵌列表的值会和变量a的同步发生改变：

[2.2-4.png]



[v2.2-5]

正确的复制嵌套元素的方法是使用module copy进行「深复制」(deep copy)，执行一下，可以看到，变量b的值并未发生任何改变：

## 2.3 切片

* ~~《第5讲》~~
* ~~《第6讲》~~

首先，和其他语言不同，Python 中的列表和元组都支持负数索引，-1 表示最后一个元素，-2 表示倒数第二个元素，以此类推。

a = [1, 2, 3, 4]

print('The last one: ', a[-1])

除了基本的初始化，索引外，列表和元组都支持切片操作，使得开发者能够轻易地访问序列中的某些元素构成的子集。切割操作的基本写法是somelist[start:end]，其中start所指的原始涵盖在切割后的范围内，而end所指的元素则不包括在切割结果之中。

a = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

print('First four:', a[:4])

print('Last four: ', a[-4:])

print('Middle two: ', a[3:-3])

“如果从列表开头获取切片，那就不要在start写上0，而是应该留空，见第5行”

“如果切片一直要取到列表末尾，那就把end留空，见第6行”

“在指定切片起止索引时，若要从列表尾部向前算，则可以使用负值来表示相关偏移量，见第7行”

1. 在=两侧的不同含义
2. 在=右侧

对原列表进行切割之后，会产生一份全新的列表，在新列表上进行修改，不会影响原列表。

[2.3-1.py]

1. 在=左侧

切片操作出现在赋值语句左侧时，有两种情况，请大家务必注意：

(1)指定了起止索引, 会把该列表中处在指定范围内的对象替换为新的值，

[2.3-2.py]

(2)没有指定起止索引，把右侧的新值复制一份，用这份拷贝替换左侧列表的全部内容，而不会重新分配新的列表，

[2.3-3.py]

## 2.4 列表推导式

* ~~《第7讲 列表推导式》~~
* 《第8讲》

2.4-1.py

## 2.5 遍历

* ~~《第10讲 enumerate》~~
* ~~《第11讲 zip》~~

2.5-1.py

定义了您最喜爱的冰淇淋口味的范围；

2.5-2.py

请注意，enumerate函数每次产生一对输出值，前者表示循环下标，即索引，在这里赋值给变量i，后者表示这个下标对应的列表元素，在这里，赋值给变量flavor

2.5-3.py

这段代码的功能是“找出列表中字母数量最多的那个名字”。定义了一个姓名列表，然后用2.4节讲过的列表推导式定义了一个姓名长度的列表，然后把这两个列表作为参数传给zip函数，在每一次迭代中，names列表的元素值赋值给变量name，而letters列表的元素值赋值给变量count，请看打印结果。

## 2.6 排序

* ~~sphinx~~

2.6-1.py

2.6-2.py

# 3 字典与集合

* 《极客》4

# 4 字符串

* bytes, str
* json串的loads&dumps
* 《第3讲 bytes, str,与unicode的区别》
* 《极客》5

# 5 异常处理

* 《第13讲》
* 《极客》8

# 6 python模块化

# 7 python参数传递

《第17讲》

《第18讲》

《第19讲》

《第20讲》

《第21讲》

# 8 装饰器

* 《极客》17

# 9 迭代器与生成器

* 《极客》19
* 《第9讲 生成器》
* 《第10讲》
* 《第11条 zip》
* 《第16讲》改写函数返回
* 《sphinx》

# 10 协程

* 《极客》20

# 11 IO

* 《极客》06

# 12 函数与闭包

Function

《第14讲》函数返回值

《第15条》

# 13 协作开发