《Python实用教程系列-推导式和Lambda表达式》

上次推文我们一起学习了python中的高级函数，推文传送门-《Python实用教程系列-高阶函数Map,Filter,Reduce》。推文中重点介绍了map,filter和reduce中相关概念，也通过实例的方式介绍了这些函数的实际用法，大家可以好好学习一下。

关注的小伙伴在后台告诉我自己对推导式和Lambda的相关知识不是很清楚，需要我写一些容易理解的教程，所以这次推文我们就来聊一聊推导式的相关知识。

**一．推导式定义**

推导式 comprehensions（又称解析式），是 python 的一种独有特性。推导式最主要的特点就是可以从一个数据序列构建另一个新的数据序列。在Python 中目前常用的推导式有列表推导式、字典推导式和集合推导式。

**二．列表推导式**

列表推导式是我们最常使用的，因为列表是我们最常使用的，其基本的语法示意如下：

▪ [表达式 for 变量 in 列表]

▪ [表达式 for 变量 in 列表 if 条件]

上述表达式中”[]”是表示python中的列表。从基本的语法形式上我们看得出来，变量是我们预先就存在的列表的值，或者是对存在的列表进行条件筛选后的值。

我们来看几个具体的例子就清晰了:

**第一种语法形式：**

list\_a = range(1,10)  
list\_b = [each for each in list\_a]  
print(list\_b)

*# 输出 [1, 2, 3, 4, 5, 6, 7, 8, 9]*

**第二种语法形式：**

*# 输出 list\_a中的基数*

list\_a = range(1,10)  
list\_b = [each for each in list\_a if each % 2 ==1]  
print(list\_b)  
*# 输出 [1, 3, 5, 7, 9]*

第二种语法形式生成的结果与第一种不同的原因在于，第二种语法形式中使用了筛选条件：each % 2 ==1

注意list\_b使用了”[]”，这就说明了其为一个列表。有的小伙伴还没看出来的话，我们使用for将第二种形式进行改写：

list\_b = []  
list\_a = range(1,10)  
for each in list\_a:  
 if each % 2 == 1:  
 list\_b.append(each)  
print(list\_b)

输出结果为list\_a中的奇数组成的列表，if each % 2 == 1在第二种形式中也完全显示了，可以看出使用列表推导式可以精简代码，阅读起来更加有Python的风格。

这里给大家设置两个小题目，大家试着解决：

1. 过滤掉列表names = ['gongsunli’,’machao’,’Gongben’,’Yuji’]中长度大于或等于4的字符串列表，并将剩下字符的转换成大写字母，结果保存为列表。
2. 给定列表A=[a,b,c], B=[d,e,f]]，找出A和B中元素可以组成的字符串的所有可能，如ab.

**三．字典推导式**

学习了列表推导式以后，学习字典推导式就没什么困难了，字典推导式语法差不多，只不过我们使用”{}”产生的是字典而已，唯一要注意的就是在字典中有键和值两个关键的属性，仿似列表推导式，那么其基本的语法可以被写作这样：

▪ { 键:值 for 键,值 in 数据结构}

▪ { 键:值 for 键 in 数据结构1 for 值 in 数据结构2 }

同样我们来看几个实际的小例子：

**第一种语法形式：**list\_c = [**'我爱'**,**'Python学堂'**]  
dict\_c = {k:v for v,k in enumerate(list\_c)}  
print(dict\_c)  
*# 输出：{'我爱': 0, 'Python学堂': 1}*

**第二种语法形式：**

list\_a = [**'我爱'**]  
list\_b = [**'Python学堂'**]  
dict\_a = {key: value for key in list\_a for value in list\_b}  
print(dict\_a)  
*#输出：{'我爱': 'Python学堂'}*

上述代码块1中使用了Python的内置函数enumerate()。有基础的小伙伴一看就知道，enumerate()可将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据下标和数据。代码块2的最后生成的结果是一个字典（或常使用生成Json数据），第二种的字典推导式是非常常见的，大家要掌握好。

再看一个例子：

*# 遍历一个有键值关系的可迭代对象*pro\_city = [(**'江苏'**, **'南京'**), (**'福建'**, **'福州'**), (**'北京'**, **'北京'**)]  
dict\_d = {key: value for key, value in pro\_city}  
print(dict\_d)

*# {'江苏': '南京', '福建': '福州', '北京': '北京'}*

再如：

dict\_e = {key: key \* key for key in [1,2,3]}  
print(dict\_e)

基本上字典推导式就这些，只不过在平时的使用中我们使用一些其他的技巧就能产生一些奇妙的现象。

**四．集合推导式**

集合我们都很了解，就是一个没有重复元素的汇集，跟列表和字典推导式类似，其基本的语法形式可以表示为：

▪ { 表达式for 项in 序列if 条件}

我们先看看集合“{}”，因为其是不重复的，因此如果有语句:print({1,2,3,4,4,5})会输出：1,2,3,4,5，重复的4将会被删除。

接着我们来看集合的相关例子：

string\_var = **'我爱Python学堂'**res = {value for value in string\_var}  
print(res)

*输出为{'h', 'y', 'o', 'P', '我', '爱', '学', 'n', '堂', 't'}*

由于集合是无序性的，因此我们再次运行上述代码的时候，会得到不同的结果的，比如我们可能得到：

{'我', 'y', 'o', 't', 'h', 'n', '学', '爱', 'P', '堂'}，{'我', 't', 'o', 'y', 'n', '堂', 'h', '爱', '学', 'P'}等等。

以上就是推导式相关的内容了，我们再来看一个很好玩的东西，很酷炫的表达式：lambda表达式。

**五．Lambda表达式**

在Java 和Python中均提供了一个lambda表达式，这个表达式又被称为匿名函数，是现代各种编程语言争相引入的一种语法，它设计精巧，在很大程度上可以精简代码，就像推导式一样。

我们之前的推文讲述了map、reduce、filter等函数都支持函数作为参数，而lambda函数就可以应用在函数式编程中。其基本语法可以简述如下：

▪ lambda [参数1[参数2, … 参数n]]: 表达式

参数的数量是不固定的，可选的。

我们来看看这个函数和我们自定义的函数之间有什么差异，假设我们现在计算两个数的和：

**自定义方式：**

def add\_two(x,y):  
 return x + y  
print(add\_two(1,2))  
  
**lambda方式：**  
res = lambda x,y:x+y  
print(res(1,2))

他们的差别很明显：lambda能够出现在Python语法不允许def出现的地方，def定义的函数一般用来处理比较复杂的功能，而lambda用来处理一些简单的操作。

之前的推文中，我们讲述了一些map、reduce、filter函数，如果这些函数和lambda结合起来使用就会产生一些比较高级的操作，比如：

**与map结合：**

list\_1=[1,2,3]  
list\_2=[10,20,30]  
print(list(map(lambda x,y:x+y,list\_1,list\_2)))

*#输出 [11, 22, 33]*

**与reduce结合：**

from functools import reduce  
print(reduce(lambda a,b: a+b ,[1,2,3,4]))  
*# 输出10*

**与filter结合：**

num\_list = [1,2,3,3,4,5,6,7,8,9,10]  
print(list(filter(lambda x : x % 2 == 0, num\_list)))  
*# 输出 [2, 4, 6, 8, 10]*

实际使用的中，我们可能会遇道更加复杂的操作，但是没关系，只要理解好lambda的精髓这些问题就不是问题啦！

最后总结一下使用lambda的好处：

（1）使用 lambda 表达式可省去定义函数的过程，让代码更加简洁。

（2）lambda 表达式具有使用后即释放的特点，从这个角度上提升了程序的性能（如内存使用）。

**六．总结**

本次推文我们讲述了Python中的推导式和Lambda表达式，这些表达式能很好精简代码并且使得代码更加的具又python风格，实际项目中我们可使用它们很好解决问题，**比如：leetcode中的第937题《重新排列日志》**,有兴趣的小伙伴可以研究一下。

以上就是本次推文的全部内容啦！如果觉得作者写的还不错的话，欢迎三连点赞分享在看支持，您的支持就是我写作的最大动力，谢谢！