## 高级特性

## 迭代

如果给定一个list或tuple，我们可以通过for循环来遍历这个list或tuple，这种遍历我们称为迭代（Iteration）。

在Python中，迭代是通过for ... in来完成的，而很多语言比如C语言，迭代list是通过下标完成的，比如Java代码：

for (i=0; i<list.length; i++) {

n = list[i];

}

可以看出，Python的for循环抽象程度要高于C的for循环，因为Python的for循环不仅可以用在list或tuple上，还可以作用在其他可迭代对象上。

list这种数据类型虽然有下标，但很多其他数据类型是没有下标的，但是，只要是可迭代对象，无论有无下标，都可以迭代，比如dict就可以迭代：

>>> d = {'a': 1, 'b': 2, 'c': 3}

>>> **for** key **in** d:

... print(key)

...

a

c

b

因为dict的存储不是按照list的方式顺序排列，所以，迭代出的结果顺序很可能不一样。

默认情况下，dict迭代的是key。如果要迭代value，可以用for value in d.values()，如果要同时迭代key和value，可以用for k, v in d.items()。

由于字符串也是可迭代对象，因此，也可以作用于for循环：

>>> **for** ch **in** 'ABC':

... print(ch)

...

A

B

C

所以，当我们使用for循环时，只要作用于一个可迭代对象，for循环就可以正常运行，而我们不太关心该对象究竟是list还是其他数据类型。

那么，如何判断一个对象是可迭代对象呢？方法是通过collections模块的Iterable类型判断：

>>> **from** collections **import** Iterable

>>> isinstance('abc', Iterable) *# str是否可迭代*

True

>>> isinstance([1,2,3], Iterable) *# list是否可迭代*

True

>>> isinstance(123, Iterable) *# 整数是否可迭代*

False

最后一个小问题，如果要对list实现类似Java那样的下标循环怎么办？Python内置的enumerate函数可以把一个list变成索引-元素对，这样就可以在for循环中同时迭代索引和元素本身：

>>> **for** i, value **in** enumerate(['A', 'B', 'C']):

... print(i, value)

...

0 A

1 B

2 C

上面的for循环里，同时引用了两个变量，在Python里是很常见的，比如下面的代码：

>>> **for** x, y **in** [(1, 1), (2, 4), (3, 9)]:

... print(x, y)

...

1 1

2 4

3 9

### 练习

请使用迭代查找一个list中最小和最大值，并返回一个tuple：

窗体顶端

# -\*- coding: utf-8 -\*-

def findMinAndMax(L):



# 测试

if findMinAndMax([]) != (None, None):

print('测试失败!')

elif findMinAndMax([7]) != (7, 7):

print('测试失败!')

elif findMinAndMax([7, 1]) != (1, 7):

print('测试失败!')

elif findMinAndMax([7, 1, 3, 9, 5]) != (1, 9):

print('测试失败!')

else:

print('测试成功!')

## 列表生成器

列表生成式即List Comprehensions，是Python内置的非常简单却强大的可以用来创建list的生成式。

举个例子，要生成list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]可以用list(range(1, 11))：

>>> list(range(1, 11))

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

但如果要生成[1x1, 2x2, 3x3, ..., 10x10]怎么做？方法一是循环：

>>> L = []

>>> **for** x **in** range(1, 11):

... L.append(x \* x)

...

>>> L

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

但是循环太繁琐，而列表生成式则可以用一行语句代替循环生成上面的list：

>>> [x \* x **for** x **in** range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

写列表生成式时，把要生成的元素x \* x放到前面，后面跟for循环，就可以把list创建出来，十分有用，多写几次，很快就可以熟悉这种语法。

for循环后面还可以加上if判断，这样我们就可以筛选出仅偶数的平方：

>>> [x \* x **for** x **in** range(1, 11) **if** x % 2 == 0]

[4, 16, 36, 64, 100]

还可以使用两层循环，可以生成全排列：

>>> [m + n **for** m **in** 'ABC' **for** n **in** 'XYZ']

['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']

三层和三层以上的循环就很少用到了。

运用列表生成式，可以写出非常简洁的代码。例如，列出当前目录下的所有文件和目录名，可以通过一行代码实现：

>>> **import** os *# 导入os模块，模块的概念后面讲到*

>>> [d **for** d **in** os.listdir('.')] *# os.listdir可以列出文件和目录*

['.emacs.d', '.ssh', '.Trash', 'Adlm', 'Applications', 'Desktop', 'Documents', 'Downloads', 'Library', 'Movies', 'Music', 'Pictures', 'Public', 'VirtualBox VMs', 'Workspace', 'XCode']

for循环其实可以同时使用两个甚至多个变量，比如dict的items()可以同时迭代key和value：

>>> d = {'x': 'A', 'y': 'B', 'z': 'C' }

>>> **for** k, v **in** d.items():

... print(k, '=', v)

...

y = B

x = A

z = C

因此，列表生成式也可以使用两个变量来生成list：

>>> d = {'x': 'A', 'y': 'B', 'z': 'C' }

>>> [k + '=' + v **for** k, v **in** d.items()]

['y=B', 'x=A', 'z=C']

最后把一个list中所有的字符串变成小写：

>>> L = ['Hello', 'World', 'IBM', 'Apple']

>>> [s.lower() **for** s **in** L]

['hello', 'world', 'ibm', 'apple']

### if ... else

使用列表生成式的时候，有些童鞋经常搞不清楚if...else的用法。

例如，以下代码正常输出偶数：

>>> [x **for** x **in** range(1, 11) **if** x % 2 == 0]

[2, 4, 6, 8, 10]

但是，我们不能在最后的if加上else：

>>> [x **for** x **in** range(1, 11) **if** x % 2 == 0 **else** 0]

File "<stdin>", line 1

[x **for** x **in** range(1, 11) **if** x % 2 == 0 **else** 0]

^

SyntaxError: invalid syntax

这是因为跟在for后面的if是一个筛选条件，不能带else，否则如何筛选？

另一些童鞋发现把if写在for前面必须加else，否则报错：

>>> [x **if** x % 2 == 0 **for** x **in** range(1, 11)]

File "<stdin>", line 1

[x **if** x % 2 == 0 **for** x **in** range(1, 11)]

^

SyntaxError: invalid syntax

这是因为for前面的部分是一个表达式，它必须根据x计算出一个结果。因此，考察表达式：x if x % 2 == 0，它无法根据x计算出结果，因为缺少else，必须加上else：

>>> [x **if** x % 2 == 0 **else** -x **for** x **in** range(1, 11)]

[-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]

上述for前面的表达式x if x % 2 == 0 else -x才能根据x计算出确定的结果。

可见，在一个列表生成式中，for前面的if ... else是表达式，而for后面的if是过滤条件，不能带else。

### 练习

如果list中既包含字符串，又包含整数，由于非字符串类型没有lower()方法，所以列表生成式会报错：

>>> L = ['Hello', 'World', 18, 'Apple', None]

>>> [s.lower() for s in L]

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 1, in <listcomp>

AttributeError: 'int' object has no attribute 'lower'

使用内建的isinstance函数可以判断一个变量是不是字符串：

>>> x = 'abc'

>>> y = 123

>>> isinstance(x, str)

True

>>> isinstance(y, str)

False

请修改列表生成式，通过添加if语句保证列表生成式能正确地执行：

窗体顶端

# -\*- coding: utf-8 -\*-

L1 = ['Hello', 'World', 18, 'Apple', None]



# 测试:

print(L2)

if L2 == ['hello', 'world', 'apple']:

print('测试通过!')

else:

print('测试失败!')

 Run

窗体底端

### 小结

运用列表生成式，可以快速生成list，可以通过一个list推导出另一个list，而代码却十分简洁。