Python函数式编程——map()、reduce()

更多 12

提起map和reduce想必大家并不陌生，Google公司2003年提出了一个名为MapReduce的编程模型[1]，用于处理大规模海量数据，并在之后广泛的应用于Google的各项应用中，2006年Apache的Hadoop项目[2]正式将MapReduce纳入到项目中。

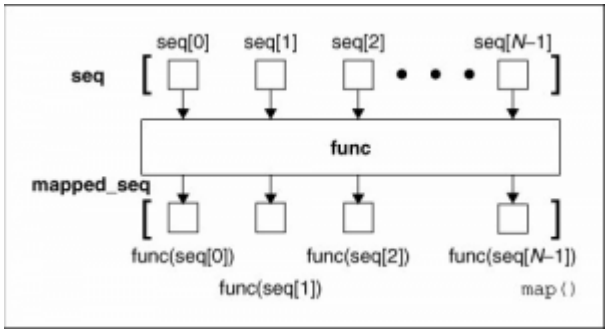
好吧，闲话少说，今天要介绍的是Python函数式编程中的另外两个内建函数 `map()` 和 `reduce()` ，而不是Google的MapReduce。

1.map()

格式： `map( func, seq1[, seq2...] )`

Python函数式编程中的 `map()` 函数是将func作用于seq中的每一个元素，并用一个列表给出返回值。如果func为None，作用同 `zip()` 。

当seq只有一个时，将func函数作用于这个seq的每个元素上，得到一个新的seq。下图说明了只有一个seq的时候map()函数是如何工作的（本文图片来源：《Core Python Programming (2nd edition)》）。



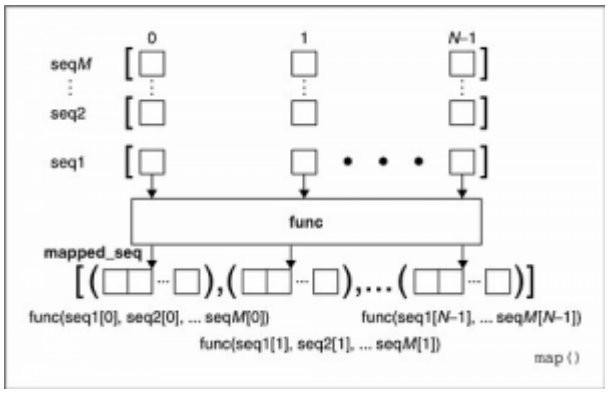
可以看出，seq中的每个元素都经过了func函数的作用，得到了func(seq[n])组成的列表。

下面举一个例子进行说明。假设我们想要得到一个列表中数字%3的余数，那么可以写成下面的代码。

```
函
1 # 使用m
2 print r
3
4 #使用列
5 print
6
```

这里又和上次的 `filter()` 一样，使用了列表解析的方法代替map执行。那么，什么时候是列表解析无法代替map的呢？

原来，**当seq多于一个时**，map可以并行地对每个seq执行如下图所示的过程：



也就是说每个seq的同一位置的元素在执行过一个多元的func函数之后，得到一个返回值，这些返回值放在一个结果列表中。

下面的例子是求两个列表对应元素的积，可以想象，这是一种可能会经常出现的状况，而如果不是用map的话，就要使用一个for循环，依次对每个位置执行该函数。

```
函
1 print r
```

上面是返回值是一个值的情况，实际上也可以是一个元组。下面的代码不止实现了乘法，也实现了加法，并把积与和放在一个元组中。

```
函
1 print r
```

还有就是上面说的**func是None**的情况，它的目的是将多个列表相同位置的元素归并到一个元组，在现在已经有了专用的函数 `zip()` 了。

```
函
1 print r
2
3 print z
```

需要注意的是，不同长度的多个seq是无法执行map函数的，会出现类型错误。

2.reduce()

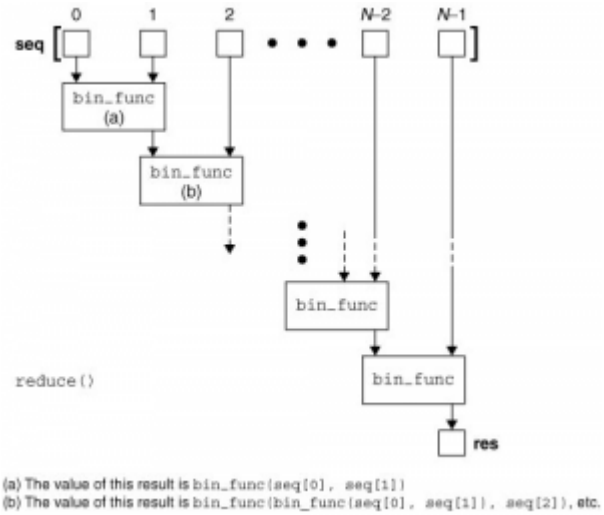
格式： `reduce( func, seq[, init] )`

reduce函数即为化简，它是这样一个过程：每次迭代，将上一次的迭代结果（第一次时为init的元素，如没有init则为seq的第一个元素）与下一个元素一同执行一个二元的func函数。在reduce函数中，init是可选的，如果使用，则作为第一次迭代的第一个元素使用。

简单来说，可以用这样一个形象化的式子来说明：

`reduce( func, [1, 2,3] ) = func( func(1, 2), 3)`

下面是reduce函数的工作过程图：



举个例子来说，**阶乘**是一个常见的数学方法，Python中并没有给出一个阶乘的内建函数，我们可以使用reduce实现一个阶乘的代码。