map/reduce

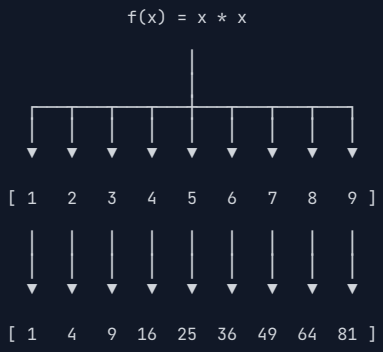
廖雪峰
资深软件开发工程师，业余马拉松选手。

Python内建了 `map()` 和 `reduce()` 函数。

如果你读过Google的那篇大名鼎鼎的论文“MapReduce: Simplified Data Processing on Large Clusters”，你就能大概明白map/reduce的概念。

我们先看map。 `map()` 函数接收两个参数，一个是函数，一个是 `Iterable` ， `map` 将传入的函数依次作用到序列的每个元素，并把结果作为新的 `Iterator` 返回。

举例说明，比如我们有一个函数$f(x)=x^2$，要把这个函数作用在一个list `[1, 2, 3, 4, 5, 6, 7, 8, 9]` 上，就可以用 `map()` 实现如下：



现在，我们用Python代码实现：

```
>>> def f(x):
...     return x * x
...
>>> r = map(f, [1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> list(r)
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

`map()` 传入的第一个参数是 `f` ，即函数对象本身。由于结果 `r` 是一个 `Iterator` ， `Iterator` 是惰性序列，因此通过 `list()` 函数让它把整个序列都计算出来并返回一个list。

你可能会想，不需要 `map()` 函数，写一个循环，也可以计算出结果：

```
L = []
for n in [1, 2, 3, 4, 5, 6, 7, 8, 9]:
    L.append(f(n))
print(L)
```

的确可以，但是，从上面的循环代码，能一眼看明白“把f(x)作用在list的每一个元素并把结果生成一个新的list”吗？

所以， `map()` 作为高阶函数，事实上它把运算规则抽象了，因此，我们不但可以计算简单的$f(x)=x^2$，还可以计算任意复杂的函数，比如，把这个list所有数字转为字符串：

```
>>> list(map(str, [1, 2, 3, 4, 5, 6, 7, 8, 9]))
['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

只需要一行代码。

再看 `reduce` 的用法。 `reduce` 把一个函数作用在一个序列 `[x1, x2, x3, ...]` 上，这个函数必须接收两个参数， `reduce` 把结果继续和序列的下一个元素做累积计算，其效果就是：

```
reduce(f, [x1, x2, x3, x4]) = f(f(f(x1, x2), x3), x4)
```

比方说对一个序列求和，就可以用 `reduce` 实现：

```
>>> from functools import reduce
>>> def add(x, y):
```

```
...     return x + y
...
>>> reduce(add, [1, 3, 5, 7, 9])
25
```

当然求和运算可以直接用Python内建函数 `sum()` ，没必要动用 `reduce` 。

但是如果要把序列 `[1, 3, 5, 7, 9]` 变换成整数 `13579` ， `reduce` 就可以派上用场：

```
>>> from functools import reduce
>>> def fn(x, y):
...     return x * 10 + y
...
>>> reduce(fn, [1, 3, 5, 7, 9])
13579
```

这个例子本身没多大用处，但是，如果考虑到字符串 `str` 也是一个序列，对上面的例子稍加改动，配合 `map()` ，我们就可以写出把 `str` 转换为 `int` 的函数：

```
>>> from functools import reduce
>>> def fn(x, y):
...     return x * 10 + y
...
>>> def char2num(s):
...     digits = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}
...     return digits[s]
...
>>> reduce(fn, map(char2num, '13579'))
13579
```

整理成一个 `str2int` 的函数就是：

```
from functools import reduce

DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}

def str2int(s):
    def fn(x, y):
        return x * 10 + y
    def char2num(s):
        return DIGITS[s]
    return reduce(fn, map(char2num, s))
```

还可以用lambda函数进一步简化成：

```
from functools import reduce

DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}

def char2num(s):
    return DIGITS[s]

def str2int(s):
    return reduce(lambda x, y: x * 10 + y, map(char2num, s))
```

也就是说，假设Python没有提供 `int()` 函数，你完全可以自己写一个把字符串转化为整数的函数，而且只需要几行代码！

lambda函数的用法在后面介绍。

练习

利用 `map()` 函数，把用户输入的不规范的英文名字，变为首字母大写，其他小写的规范名字。输入： `['adam', 'LISA', 'barT']` ，输出： `['Adam', 'Lisa', 'Bart']` ：

```
def normalize(name):
    pass

# 测试:
L1 = ['adam', 'LISA', 'barT']
```

```
L2 = list(map(normalize, L1))
print(L2)
```

Python提供的 `sum()` 函数可以接受一个list并求和，请编写一个 `prod()` 函数，可以接受一个list并利用 `reduce()` 求积：

```
from functools import reduce

def prod(L):
    pass

print('3 * 5 * 7 * 9 =', prod([3, 5, 7, 9]))
if prod([3, 5, 7, 9]) == 945:
    print('测试成功!')
else:
    print('测试失败!')
```

利用 `map` 和 `reduce` 编写一个 `str2float` 函数，把字符串 `'123.456'` 转换成浮点数 `123.456` ：

```
from functools import reduce

def str2float(s):
    pass

print('str2float(\'123.456\') =', str2float('123.456'))
if abs(str2float('123.456') - 123.456) < 0.00001:
    print('测试成功!')
else:
    print('测试失败!')
```

参考代码

do_map.py

do_reduce.py

小结

`map` 用于将一个函数作用于一个序列，以此得到另一个序列；

`reduce` 用于将一个函数依次作用于上次计算的结果和序列的下一个元素，以此得到最终结果。

« 高阶函数　　filter »

Comments

Loading comments...