7.9 将单方法的类转换为函数¶

问题¶

你有一个除 `__init__()` 方法外只定义了一个方法的类。为了简化代码，你想将它转换成一个函数。

解决方案¶

大多数情况下，可以使用闭包来将单个方法的类转换成函数。 举个例子，下面示例中的类允许使用者根据某个模板方案来获取到URL链接地址。

```
from urllib.request import urlopen

class UrlTemplate:
    def __init__(self, template):
        self.template = template

    def open(self, **kwargs):
        return urlopen(self.template.format_map(kwargs))

# Example use. Download stock data from yahoo
yahoo = UrlTemplate('http://finance.yahoo.com/d/quotes.csv?s={names}&f={fields}')
for line in yahoo.open(names='IBM,AAPL,FB', fields='sl1c1v'):
    print(line.decode('utf-8'))
```

这个类可以被一个更简单的函数来代替：

```
def urltemplate(template):
    def opener(**kwargs):
        return urlopen(template.format_map(kwargs))
    return opener

# Example use
yahoo = urltemplate('http://finance.yahoo.com/d/quotes.csv?s={names}&f={fields}')
for line in yahoo(names='IBM,AAPL,FB', fields='sl1c1v'):
    print(line.decode('utf-8'))
```

讨论¶

大部分情况下，你拥有一个单方法类的原因是需要存储某些额外的状态来给方法使用。 比如，定义UrlTemplate类的唯一目的就是先在某个地方存储模板值，以便将来可以在open()方法中使用。

使用一个内部函数或者闭包的方案通常会更优雅一些。简单来讲，一个闭包就是一个函数， 只不过在函数内部带上了一个额外的变量环境。闭包关键特点就是它会记住自己被定义时的环境。 因此，在我们的解决方案中，`opener()` 函数记住了 `template` 参数的值，并在接下来的调用中使用它。

任何时候只要你碰到需要给某个函数增加额外的状态信息的问题，都可以考虑使用闭包。 相比将你的函数转换成一个类而言，闭包通常是一种更加简洁和优雅的方案。