链滴

Python-CookBook：41、执行精确的小数计算

作者：zhaolixiang
原文链接：https://ld246.com/article/1567157844568
来源网站：链滴

问题

我们需要对小数进行精确计算，不希望因为浮点数天生的误差而带来影响。

解决方案

关于浮点数，一个尽人皆知的问题就是它们无法精确表达出所有的十进制小数位。此外，甚至连简单数学计算也会引入微小的误差。例如：

```
>>> a = 4.2
>>> b = 2.1
>>> a + b
6.300000000000001
>>> (a + b) == 6.3
False
>>>
```

这些误差实际上是底层CPU的浮点运算单元和IEEE 754浮点数算术标准的一种“特性”。由于Pytho的浮点数类型保存的数据采用的是原始表示形式，因此如果编写的代码用到了float实例，那就无法避这样的误差。

如果期望得到更高的精度（并愿意为此牺牲掉一些性能），可以使用decimal模块：

```
>>> from decimal import Decimal
>>> a = Decimal('4.2')
>>> b = Decimal('2.1')
>>> a + b
Decimal('6.3')
>>> print(a + b)
6.3
>>> (a + b) == Decimal('6.3')
True
>>>
```

这么做初看起来似乎有点怪异（将数字以字符串的形式来指定）。但是，Decimal对象能以任何期望方式来工作（支持所有常见的数学操作）。如果要将它们打印出来或是在字符串格式化函数中使用，们看起来就和普通的数字一样。

decimal模块的主要功能是允许控制计算过程中的各个方面，这包括数字的位数和四舍五入。要做到些，需要创建一个本地的上下文环境然后修改其设定。示例如下：

```
>>> from decimal import localcontext
>>> a = Decimal('1.3')
>>> b = Decimal('1.7')
>>> print(a / b)
0.76470588235294117647058823529
>>> with localcontext() as ctx:
...     ctx.prec = 3
...     print(a / b)
...
0.765
>>> with localcontext() as ctx:
...     ctx.prec = 50
```

```
...     print(a / b)
...
0.76470588235294117647058823529411764705882352941176
>>>
```

讨论

decimal模块实现了IBM的通用十进制算术规范（General Decimal Arithmetic Specification）。不说，这里面有着数量庞大的配置选项，这些都超出了本书的范围。

Python新手可能会倾向于利用decimal模块来规避处理float数据类型所固有的精度问题。但是，正确解你的应用领域是至关重要的。如果我们处理的是科学或工程类的问题，像计算机图形学或者大部分有科学性质的问题，那么更常见的做法是直接使用普通的浮点类型。首先，在真实世界中极少有什么西需要计算到小数点后17位（float提供17位的精度）。因此，在计算中引入的微小误差根本就不足齿。其次，原生的浮点数运算性能要快上许多——如果要执行大量的计算，那性能问题就显得很重要。

也就是说我们无法完全忽略误差。数学家花费了大量的时间来研究各种算法，其中一些算法的误差处能力优于其他的算法。我们同样还需要对类似相减抵消（subtractive cancellation）以及把大数和小加在一起时的情况多加小心。示例如下：

```
>>> nums = [1.23e+18, 1, -1.23e+18]
>>> sum(nums)     # Notice how 1 disappears
0.0
>>>
```

上面这个例子可以通过使用math.fsum()以更加精确的实现来解决：

```
>>> import math
>>> math.fsum(nums)
1.0
>>>
```

但是对于其他的算法，需要研究算法本身，并理解其误差传播（error propagation）的性质。

综上所述，decimal模块主要用在涉及像金融这一类业务的程序中。在这样的程序里，计算中如果出微小的误差是相当令人生厌的。因此，decimal模块提供了一种规避误差的方式。当用Python作数据的接口时也会常常会遇到Decimal对象——尤其是当访问金融数据时更是如此。