3.3 数字的格式化输出¶

问题¶

你需要将数字格式化后输出，并控制数字的位数、对齐、千位分隔符和其他的细节。

解决方案¶

格式化输出单个数字的时候，可以使用内置的 `format()` 函数，比如：

```
>>> x = 1234.56789

>>> # Two decimal places of accuracy
>>> format(x, '0.2f')
'1234.57'

>>> # Right justified in 10 chars, one-digit accuracy
>>> format(x, '>10.1f')
'    1234.6'

>>> # Left justified
>>> format(x, '<10.1f')
'1234.6    '

>>> # Centered
>>> format(x, '^10.1f')
'  1234.6  '

>>> # Inclusion of thousands separator
>>> format(x, ',')
'1,234.56789'
>>> format(x, '0,.1f')
'1,234.6'
>>>
```

如果你想使用指数记法，将f改成e或者E(取决于指数输出的大小写形式)。比如：

```
>>> format(x, 'e')
'1.234568e+03'
>>> format(x, '0.2E')
'1.23E+03'
>>>
```

同时指定宽度和精度的一般形式是 `'[<>^]?width[,]?(.digits)?'`， 其中 `width` 和 `digits` 为整数，？代表可选部分。 同样的格式也被用在字符串的 `format()` 方法中。比如：

```
>>> 'The value is {:0,.2f}'.format(x)
'The value is 1,234.57'
>>>
```

讨论¶

数字格式化输出通常是比较简单的。上面演示的技术同时适用于浮点数和 `decimal` 模块中的 `Decimal` 数字对象。

当指定数字的位数后，结果值会根据 `round()` 函数同样的规则进行四舍五入后返回。比如：

```
>>> x
1234.56789
>>> format(x, '0.1f')
'1234.6'
>>> format(-x, '0.1f')
'-1234.6'
>>>
```

包含千位符的格式化跟本地化没有关系。 如果你需要根据地区来显示千位符，你需要自己去调查下 `locale` 模块中的函数了。 你同样也可以使用字符串的 `translate()` 方法来交换千位符。比如：

```
>>> swap_separators = { ord('.'):',', ord(','):'.' }
>>> format(x, ',').translate(swap_separators)
'1.234,56789'
>>>
```

在很多Python代码中会看到使用%来格式化数字的，比如：

```
>>> '%0.2f' % x
'1234.57'
>>> '%10.1f' % x
'    1234.6'
>>> '%-10.1f' % x
'1234.6    '
>>>
```

这种格式化方法也是可行的，不过比更加先进的 `format()` 要差一点。 比如，在使用%操作符格式化数字的时候，一些特性(添加千位符)并不能被支持。