8.2 自定义字符串的格式化¶

问题¶

你想通过 `format()` 函数和字符串方法使得一个对象能支持自定义的格式化。

解决方案¶

为了自定义字符串的格式化，我们需要在类上面定义 `__format__()` 方法。例如：

```
_formats = {
   'ymd' : '{d.year}-{d.month}-{d.day}',
   'mdy' : '{d.month}/{d.day}/{d.year}',
   'dmy' : '{d.day}/{d.month}/{d.year}'
   }

class Date:
   def __init__(self, year, month, day):
      self.year = year
      self.month = month
      self.day = day

   def __format__(self, code):
      if code == '':
         code = 'ymd'
      fmt = _formats[code]
      return fmt.format(d=self)
```

现在 `Date` 类的实例可以支持格式化操作了，如同下面这样：

```
>>> d = Date(2012, 12, 21)
>>> format(d)
'2012-12-21'
>>> format(d, 'mdy')
'12/21/2012'
>>> 'The date is {:ymd}'.format(d)
'The date is 2012-12-21'
>>> 'The date is {:mdy}'.format(d)
'The date is 12/21/2012'
>>>
```

讨论¶

`__format__()` 方法给Python的字符串格式化功能提供了一个钩子。 这里需要着重强调的是格式化代码的解析工作完全由类自己决定。因此，格式化代码可以是任何值。 例如，参考下面来自 `datetime` 模块中的代码：

```
>>> from datetime import date
>>> d = date(2012, 12, 21)
>>> format(d)
'2012-12-21'
>>> format(d,'%A, %B %d, %Y')
'Friday, December 21, 2012'
>>> 'The end is {:%d %b %Y}. Goodbye'.format(d)
'The end is 21 Dec 2012. Goodbye'
```

```
>>>
```

对于内置类型的格式化有一些标准的约定。 可以参考 string模块文档 说明。