动手试一试

16-1 旧金山：旧金山的气温更接近于锡特卡还是死亡谷呢？请绘制一个显示旧金山最高气温和最低气温的图表，并进行比较。可从 http://www.wunderground.com/history/ 下载几乎任何地方的天气数据。为此，请输入相应的地方和日期范围，滚动到页面底部，找到名为 Comma-Delimited File 的链接，再单击该链接，将数据存储为 CSV 文件。

16-2 比较锡特卡和死亡谷的气温：在有关锡特卡和死亡谷的图表中，气温刻度反映了数据范围的不同。为准确地比较锡特卡和死亡谷的气温范围，需要在 y 轴上使用相同的刻度。为此，请修改图 16-5 和图 16-6 所示图表的 y 轴设置，对锡特卡和死亡谷的气温范围进行直接比较（你也可以对任何两个地方的气温范围进行比较）。你还可以尝试在一个图表中呈现这两个数据集。

16-3 降雨量：选择你感兴趣的任何地方，通过可视化将其降雨量呈现出来。为此，可先只涵盖一个月的数据，确定代码正确无误后，再使用一整年的数据来运行它。

16-4 探索：生成一些图表，对你好奇的任何地方的其他天气数据进行研究。

16.2　制作交易收盘价走势图：JSON 格式

在本节中[①]，你将下载JSON格式的交易收盘价数据，并使用模块json来处理它们。Pygal提供了一个适合初学者使用的绘图工具，你将使用它来对收盘价数据进行可视化，以探索价格变化的周期性。

16.2.1　下载收盘价数据

收盘价数据文件位于https://raw.githubusercontent.com/muxuezi/btc/master/btc_close_2017.json。可以直接将文件btc_close_2017.json下载到本章程序所在的文件夹中，也可以用Python 2.*x*标准库中模块urllib2（Python 3.*x*版本使用的是urllib）的函数urlopen来做，还可以通过Python的第三方模块requests（将在第17章介绍）下载数据。

首先，我们直接下载btc_close_2017.json，看看如何着手处理这个文件中的数据：

btc_close_2017.json

```
[
  {
    "date": "2017-01-01",
    "month": "01",
    "week": "52",
```

① 本节为陶俊杰根据原作编写。源代码请见本书主页ituring.cn/book/1861，代码为Python 3版本，尽量支持Python 2版本，可以在binder容器中执行jupyter notebook。——编者注

```
    "weekday": "Sunday",
    "close": "6928.6492"
  },
  --snip--
  {
    "date": "2017-12-12",
    "month": "12",
    "week": "50",
    "weekday": "Tuesday",
    "close": "113732.6745"
  }
]
```

这个文件实际上就是一个很长的Python列表，其中每个元素都是一个包含五个键的字典：统计日期、月份、周数、周几以及收盘价。由于2017年1月1日是周日，作为2017年的第一周实在太短，因此将其计入2016年的第52周。于是2017年的第一周是从2017年1月2日（周一）开始的。如果用函数urlopen来下载数据，可以使用下面的代码：

btc_close_2017.py

```
from __future__ import (absolute_import, division,
                        print_function, unicode_literals)
❶ try:
    # Python 2.x 版本
    from urllib2 import urlopen
except ImportError:
    # Python 3.x 版本
    from urllib.request import urlopen
import json

json_url = 'https://raw.githubusercontent.com/muxuezi/btc/master/btc_close_2017.json'
❷ response = urlopen(json_url)
# 读取数据
req = response.read()
# 将数据写入文件
❸ with open('btc_close_2017_urllib.json','wb') as f:
    f.write(req)
# 加载json格式
❹ file_urllib = json.loads(req)
print(file_urllib)
```

首先导入下载文件使用的模块。这里用try/except语句（见❶）实现兼容Python 2.*x*和Python 3.*x*代码。ImportError可以作为判断Python 2.*x*和Python 3.*x*的方式：如果用Python 2.*x*版本运行代码，from urllib2 import urlopen代码就会执行；如果用Python 3.*x*版本运行代码，由于没有urllib2模块，解释器就会触发ImportError，因此from urllib.request import urlopen代码就会执行。条条大路通罗马，最终都会导入urlopen函数。

然后导入模块json，以便之后能够正确地加载文件中的数据。我们的btc_close_2017.json文件放在GitHub网站上，urlopen(json_url)是将json_url网址传入urlopen函数（见❷）。这行代码执行后，Python就会向GitHub的服务器发送请求。GitHub的服务器响应请求后把btc_close_2017.json文件发送给Python，之后用response.read()就可以读取文件数据。这时，可以将文件数据保存到文件夹中（见❸），btc_close_2017_urllib.json与btc_close_2017.json的内容是一样的。最后，我们用函数json.load()（见❹）将文件内容转换成Python能够处理的格式，与前面直接下载的文件内容一致。

函数urlopen的代码稍微复杂一些，第三方模块requests封装了许多常用的方法，让数据下载和读取方式变得非常简单：

```
import requests

json_url = 'https://raw.githubusercontent.com/muxuezi/btc/master/btc_close_2017.json'
❶ req = requests.get(json_url)
# 将数据写入文件
with open('btc_close_2017_request.json','w') as f:
❷     f.write(req.text)
❸ file_requests = req.json()
```

输出结果为：

```
{'date': '2017-01-01', 'month': '01', 'week': '52', 'weekday': 'Sunday', 'close': '6928.6492'},
{'date': '2017-01-02', 'month': '01', 'week': '1', 'weekday': 'Monday', 'close': '7070.2554'},
--snip--
{'date': '2017-12-11', 'month': '12', 'week': '50', 'weekday': 'Monday', 'close': '110642.88'},
{'date': '2017-12-12', 'month': '12', 'week': '50', 'weekday': 'Tuesday', 'close': '113732.6745'}
```

requests通过get方法（见❶）向GitHub服务器发送请求。GitHub服务器响应请求后，返回的结果存储在req变量中。req.text属性可以直接读取文件数据，返回格式是字符串（见❷），可以像之前一样保存为文件btc_close_2017_request.json，其内容与btc_close_2017_urllib.json是一样的。另外，直接用req.json()（见❸）就可以将btc_close_2017.json文件的数据转换成Python列表file_requests，与之前的file_urllib内容相同。

```
print(file_urllib == file_requests)
```

输出结果为：

```
True
```

16.2.2　提取相关的数据

下面编写一个小程序来提取btc_close_2017.json文件中的相关信息：

btc_close_2017.py

```
import json

# 将数据加载到一个列表中
filename = 'btc_close_2017.json'
with open(filename) as f:
❶    btc_data = json.load(f)
# 打印每一天的信息
❷ for btc_dict in btc_data:
❸    date = btc_dict['date']
    month = btc_dict['month']
    week = btc_dict['week']
    weekday = btc_dict['weekday']
    close = btc_dict['close']
    print("{} is month {} week {}, {}, the close price is {} RMB".format(date, month, week, weekday, close))
```

首先导入模块json，然后将数据存储在btc_data中（见❶）。在❷处，我们遍历了btc_data中的每个元素。每个元素都是一个字典，包含五个键-值对，btc_dict就用来存储字典中的每个键-值对。之后就可以取出所有键的值（见❸），并将日期、月份、周数、周几和收盘价相关联的值分别存储到date、month、week、weekday与close中。接下来，打印每一天的日期、月份、周数、周几和收盘价。输出结果如下：

```
2017-01-01 is month 01 week 52, Sunday, the close price is 6928.6492 RMB
2017-01-02 is month 01 week 1, Monday, the close price is 7070.2554 RMB
2017-01-03 is month 01 week 1, Tuesday, the close price is 7175.1082 RMB
--snip--
2017-12-10 is month 12 week 49, Sunday, the close price is 99525.1027 RMB
2017-12-11 is month 12 week 50, Monday, the close price is 110642.88 RMB
2017-12-12 is month 12 week 50, Tuesday, the close price is 113732.6745 RMB
```

现在，我们已经掌握了json读取数据的方法。下面，让我们将数据转换为Pygal能够处理的格式。

16.2.3 将字符串转换为数字值

btc_close_2017.json中的每个键和值都是字符串。为了能在后面的内容中对交易数据进行计算，需要先将表示周数和收盘价的字符串转换为数值。因此我们使用函数int()：

btc_close_2017.py

```
--snip--

# 打印每一天的信息
for btc_dict in btc_data:
    date = btc_dict['date']
    month = int(btc_dict['month'])
❶    week = int(btc_dict['week'])
    weekday = btc_dict['weekday']
```

```
❷    close = int(btc_dict['close'])
     print("{} is month {} week {}, {}, the close price is {} RMB".format(date, month, week, weekday, close))
```

在❶处，我们将周数的数值都转换为整数格式。将收盘价close转换为整数时，出现了ValueError异常，如下所示：

```
--snip--
❶       5     week = int(btc_dict['week'])
        6     weekday = btc_dict['weekday']
❷ ----> 7     close = int(btc_dict['close'])
        8     print("{} is month {} week {}, {}, the close price is {} RMB".format(date, month, week,
weekday, close))

ValueError: invalid literal for int() with base 10: '6928.6492'
```

在实际工作中，原始数据的格式经常是不统一的，此类数值类型转换造成的ValueError异常十分普遍。这里的原因在于，Python不能直接将包含小数点的字符串'6928.6492'转换为整数。为了消除这种错误，需要先将字符串转换为浮点数（float），再将浮点数转换为整数（int）：

btc_close_2017.py

```
--snip--

# 打印每一天的信息
for btc_dict in btc_data:
    date = btc_dict['date']
    month = int(btc_dict['month'])
    week = int(btc_dict['week'])
    weekday = btc_dict['weekday']
❶   close = int(float(btc_dict['close']))
    print("{} is month {} week {}, {}, the close price is {} RMB".format(date, month, week, weekday, close))
```

这里首先用函数float()将字符串转换为小数（见❶），然后再用函数int()去掉小数部分（截尾取整），返回整数部分。现在，再次运行代码，就不会出现异常了。打印出的收盘价信息如下：

```
2017-01-01 is month 1 week 52, Sunday, the close price is 6928 RMB
2017-01-02 is month 1 week 1, Monday, the close price is 7070 RMB
2017-01-03 is month 1 week 1, Tuesday, the close price is 7175 RMB
--snip--
2017-12-10 is month 12 week 49, Sunday, the close price is 99525 RMB
2017-12-11 is month 12 week 50, Monday, the close price is 110642 RMB
2017-12-12 is month 12 week 50, Tuesday, the close price is 113732 RMB
```

现在，收盘价都已经成功地先从字符串转换成浮点数，再从浮点数转换成了整数。另外，我们还发现，月份中原来1~9月前面的数字0在转换成整数之后都消失了，周数的数据也根据我们的需求转换成了整数。有了这些数据之后，可以结合Pygal的可视化功能来探索一些有趣的信息。

16.2.4 绘制收盘价折线图

第15章已经介绍了用Pygal绘制条形图（bar chart）的方法，也介绍了用matplotlib绘制折线图（line chart）的方法。下面用Pygal来实现收盘价的折线图。

绘制折线图之前，需要获取x轴与y轴数据，因此我们创建了几个列表来存储数据。遍历btc_data，将转换为适当格式的数据存储到对应的列表中。对前面的代码做一些简单的调整：

btc_close_2017.py

```
--snip--

# 创建5个列表，分别存储日期和收盘价
dates = []
months = []
weeks = []
weekdays = []
close = []
# 每一天的信息
for btc_dict in btc_data:
    dates.append(btc_dict['date'])
    months.append(int(btc_dict['month']))
    weeks.append(int(btc_dict['week']))
    weekdays.append(btc_dict['weekday'])
    close.append(int(float(btc_dict['close'])))
```

有了x轴与y轴的数据，就可以绘制折线图了。由于数据点比较多，x轴要显示346个日期，在有限的屏幕上会显得十分拥挤。因此我们需要利用Pygal的配置参数，对图形进行适当的调整。代码如下：

btc_close_2017.py

```
--snip--

import pygal

❶ line_chart = pygal.Line(x_label_rotation=20, show_minor_x_labels=False)
line_chart.title = '收盘价（¥）'
line_chart.x_labels = dates
N = 20 # x轴坐标每隔20天显示一次
❷ line_chart.x_labels_major = dates[::N]
line_chart.add('收盘价', close)
line_chart.render_to_file('收盘价折线图（¥）.svg')
```

首先导入模块pygal，然后在创建Line实例时，分别设置了x_label_rotation与show_minor_x_labels作为初始化参数（见❶）。x_label_rotation=20让x轴上的日期标签顺时针旋转20°，show_minor_x_labels=False则告诉图形不用显示所有的x轴标签。设置了图形的标题和x轴标签之后，我们配置x_labels_major属性，让x轴坐标每隔20天显示一次（见❷），这样x轴就不会显得非常拥挤了。最终效果如图16-7所示。

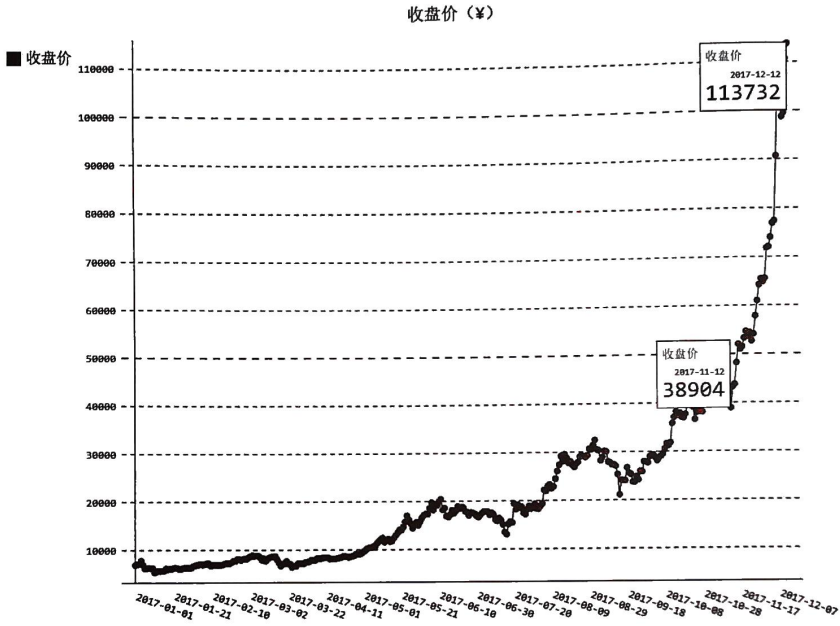


图16-7　收盘价折线图（¥）

从图中可以看出，价格从2017年11月12日到2017年12月12日快速增长，平均每天增值约2500元人民币，图中折线简直就像火箭发射一般，垂直升空。下面对价格做一些简单的探索。

16.2.5　时间序列特征初探

进行时间序列分析总是期望发现趋势（trend）、周期性（seasonality）和噪声（noise），从而能够描述事实、预测未来、做出决策。从收盘价的折线图可以看出，2017年的总体趋势是非线性的，而且增长幅度不断增大，似乎呈指数分布。但是，我们还发现，在每个季度末（3月、6月和9月）似乎有一些相似的波动。尽管这些波动被增长的趋势掩盖了，不过其中也许有周期性。为了验证周期性的假设，需要首先将非线性的趋势消除。对数变换（log transformation）是常用的处理方法之一。让我们用Python标准库的数学模块math来解决这个问题。math里有许多常用的数学函数，这里用以10为底的对数函数math.log10计算收盘价，日期仍然保持不变。这种方式称为半对数（semi-logarithmic）变换。代码如下：

btc_close_2017.py

```
--snip--

import pygal
import math

line_chart = pygal.Line(x_label_rotation=20, show_minor_x_labels=False)
line_chart.title = '收盘价对数变换（¥）'
line_chart.x_labels = dates
N = 20 # x轴坐标每隔20天显示一次
line_chart.x_labels_major = dates[::N]
❶ close_log = [math.log10(_) for _ in close]
line_chart.add('log收盘价', close_log)
line_chart.render_to_file('收盘价对数变换折线图（¥）.svg')
```

现在，用对数变换剔除非线性趋势之后，整体上涨的趋势更接近线性增长。从图16-8中可以清晰地看出，收盘价在每个季度末似乎有显著的周期性——3月、6月和9月都出现了剧烈的波动。那么，12月是不是会再现这一场景呢？下面再看看收盘价的月日均值与周日均值的表现。

图16-8 收盘价对数变换折线图（¥）

16.2.6　收盘价均值

下面再利用btc_close_2017.json文件中的数据，绘制2017年前11个月的日均值、前49周（2017-01-02~2017-12-10）的日均值，以及每周中各天（Monday~Sunday）的日均值。虽然这些日均值的数值不同，但都是一段时间的均值，计算方法是一样的。因此，可以将之前的绘图代码封装成函数，以便重复使用。

btc_close_2017.py

```
--snip--

❶ from itertools import groupby

def draw_line(x_data, y_data, title, y_legend):
    xy_map = []
❷   for x, y in groupby(sorted(zip(x_data, y_data)), key=lambda _: _[0]):
        y_list = [v for _, v in y]
❸       xy_map.append([x, sum(y_list) / len(y_list)])
❹   x_unique, y_mean = [*zip(*xy_map)]
    line_chart = pygal.Line()
    line_chart.title = title
    line_chart.x_labels = x_unique
    line_chart.add(y_legend, y_mean)
    line_chart.render_to_file(title+'.svg')
    return line_chart
```

由于需要将数据按月份、周数、周几分组，再计算每组的均值，因此我们导入Python标准库中模块itertools的函数groupby（见❶）。然后将x轴与y轴的数据合并、排序，再用函数groupby分组（见❷）。分组之后，求出每组的均值，存储到xy_map变量中（见❸）。最后，将xy_map中存储的x轴与y轴数据分离（见❹），就可以像之前那样用Pygal画图了。下面我们画出收盘价月日均值。由于2017年12月的数据并不完整，我们只取2017年1月到11月的数据。通过dates查找2017-12-01的索引位置，确定周数和收盘价的取数范围。代码如下所示：

```
idx_month = dates.index('2017-12-01')
line_chart_month = draw_line(months[:idx_month], close[:idx_month], '收盘价月日均值 (¥) ',
'月日均值')
line_chart_month
```

收盘价月日均值如图16-9所示。

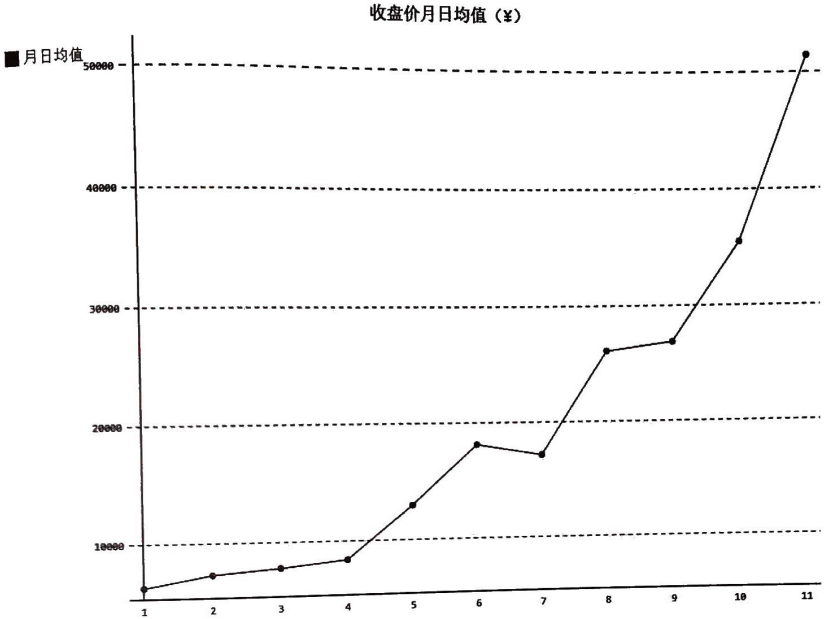


图16-9 收盘价月日均值（¥）

从图16-9中可以看出，除了7月相比上个月有所下降，其他各月都是增长的。11月相比10月的增幅非常惊人，月日均增长了45%。

下面再来绘制前49周（2017-01-02~2017-12-10）的日均值。2017年1月1日是周日，归属为2016年第52周，因此2017年的第一周从2017年1月2日开始，取数时需要将第一天去掉。另外，2017年第49周周日是2017年12月10日，因此我们通过dates查找2017-12-11的索引位置，确定周数和收盘价的取数范围。代码如下所示：

```
idx_week = dates.index('2017-12-11')
line_chart_week = draw_line(weeks[1:idx_week], close[1:idx_week], '收盘价周日均值（¥）', '周日均值
')
line_chart_week
```

收盘价周日均值如图16-10所示。

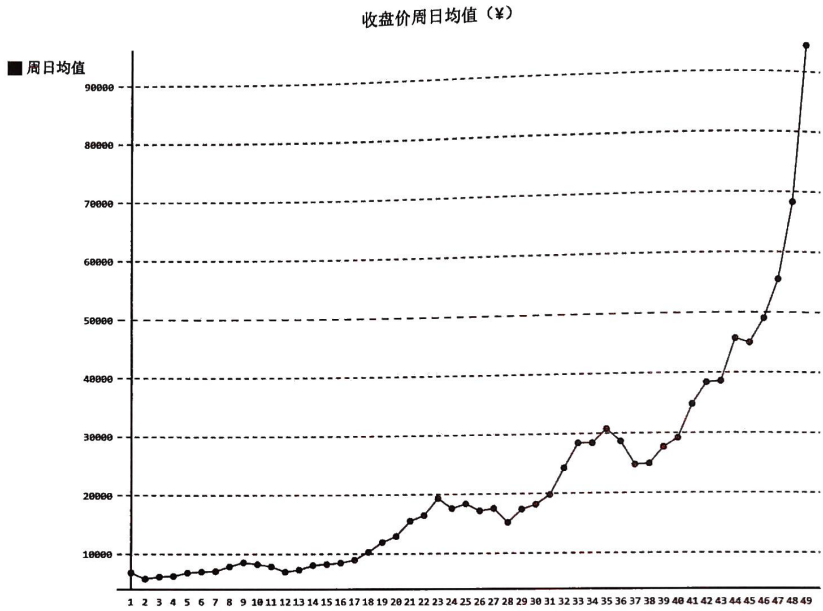


图16-10　收盘价周日均值（¥）

从图16-10中可以看出，价格与节假日无关。在2017年的各个节假日都没有出现价格低点，包括春节（第4周）、清明节（第14周）、劳动节（第18周）、端午节（第22周）、国庆节（第40周）。

最后，绘制每周中各天的均值。为了使用完整的时间段，还像前面那样取前49周（2017-01-02~2017-12-10）的数据，同样通过dates查找2017-12-11的索引位置，确定周数和收盘价的取数范围。但是，由于这里的周几是字符串，按周一到周日的顺序排列，而不是单词首字母的顺序，绘图时x轴标签的顺序会有问题。另外，原来的周几都是英文单词，还可以将其调整为中文。因此，需要对前面的程序做一些特殊处理，代码如下所示：

```
idx_week = dates.index('2017-12-11')
❶ wd = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday','Sunday']
weekdays_int = [wd.index(w) + 1 for w in weekdays[1:idx_week]]
line_chart_weekday = draw_line(weekdays_int, close[1:idx_week], '收盘价星期均值（¥）',
    '星期均值')
❷ line_chart_weekday.x_labels = ['周一','周二','周三','周四','周五','周六','周日']
    line_chart_weekday.render_to_file('收盘价星期均值（¥）.svg')
```

首先，我们列出一周七天的英文单词，然后将weekdays的内容替换成1~7的整数（见❶）。这样函数draw_line在处理数据时按周几的顺序排列，就会将周一放在列表的第一位，周日放在列表的第七位。图形生成之后，再将图形的x轴标签替换为中文（见❷），最终输出的图形如图16-11所示。

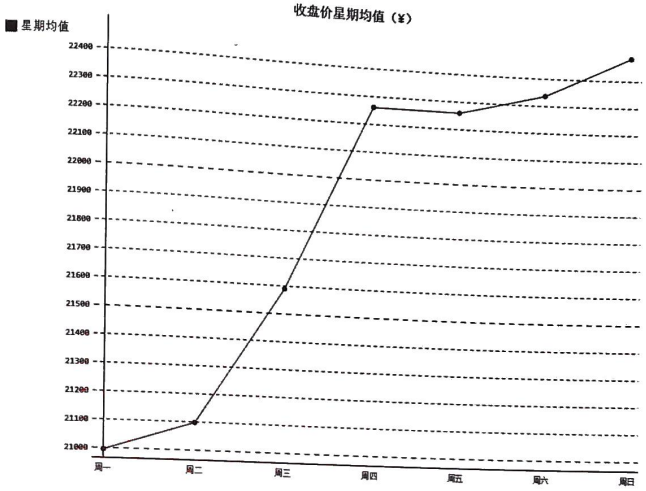


图16-11 收盘价星期均值（¥）

从图16-11中可以看出，比特币收盘价在周一最低，周日最高。周一到周四快速拉升，周四是拐点，周五和周四基本持平（其实略低于周四），之后增速放慢。

16.2.7 收盘价数据仪表盘

前面已经为交易收盘价绘制了五幅图，分别是收盘价对数变换、收盘价月日均值、收盘价周日均值、收盘价星期均值。每个SVG文件打开之后都是独立的页面。如果能够将它们整合在一起，就可以很方便地进行长期管理、监测和分析。另外，新的图表也可以十分方便地加入进来，这样就形成了一个数据仪表盘（dashboard）。下面将前面绘制的图整合起来，做一个收盘价数据仪表盘。代码如下：

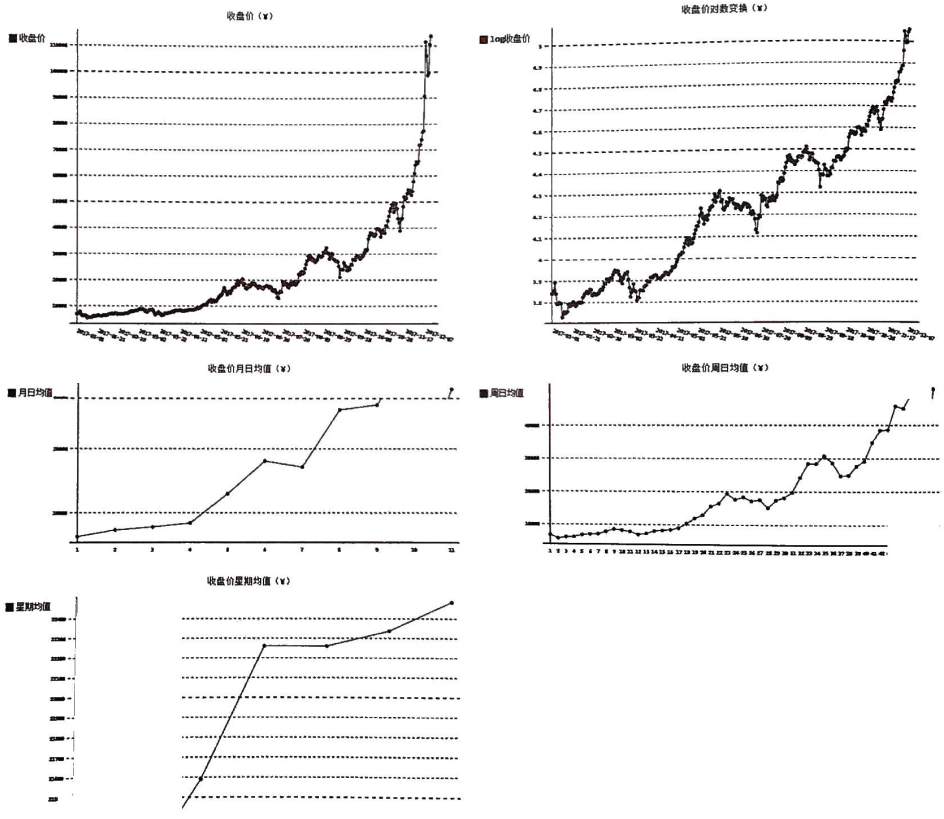
btc_close_2017.py

```
--snip--

❶ with open('收盘价Dashboard.html', 'w', encoding='utf8') as html_file:
    html_file.write('<html><head><title>收盘价Dashboard</title><meta
charset="utf-8"></head><body>\n')
    for svg in [
            '收盘价折线图（¥）.svg', '收盘价对数变换折线图（¥）.svg', '收盘价月日均值（¥）.svg',
            '收盘价周日均值（¥）.svg', '收盘价星期均值（¥）.svg'
    ]:
        html_file.write('    <object type="image/svg+xml" data="{0}"
height=500></object>\n'.format(svg))
    html_file.write('</body></html>')
```

和常见网络应用的数据仪表盘一样，我们的数据仪表盘也是一个完整的网页（HTML文件）。首先，需要创建一个名为收盘价Dashboard.html的网页文件，然后将每幅图都添加到页面中（见❶）。这里设置SVG图形的默认高度为500像素，由于SVG是矢量图，可以任意缩放且不失真，因此可以通过放大或缩小网页来调整视觉效果。最终效果如图16-12所示，每一幅图都是前面演示过的内容。相信聪明的你一定有更有趣的想法，赶紧动手往数据仪表盘里添加一些新图形吧。

收盘价Dashboard



关于交易收盘价的分析就介绍到这里，内容非常粗糙、极不严谨，仅作为Python处理JSON文件格式的示例使用。数据背后的故事往往是非常复杂的，即使精通数据分析的技巧，也未必能预见未来。“世事洞明皆学问，人情练达即文章”，在数据分析过程中，这些修炼都是必不可少的，也是非常艰难的挑战。

动手试一试

16-5 分析更完整的数据：本节仅使用了交易收盘价在2017年的部分数据。如果要全面地分析价格走势，还是应该收集更加完整的数据；目前最早的交易时间数据可以追溯到2012年。如果你感兴趣，可以收集早期的数据进行分析。

16-6 选择你感兴趣的数据：免费的JSON格式数据非常丰富，许多著名的国际组织都在积极分享有价值的数据。例如，Open Knowledge International（https://okfn.org/）上就有许多有趣的JSON数据。你也可以用本节的方法获取它们，开启自己的分析项目。

16-7 测试函数 draw_line：我们编写函数draw_line时，没有使用测试方法检验它能否正确工作。请利用你在第11章学到的知识，为这个函数编写合适的测试程序。

16-8 尝试 Python 数据科学工具：虽然Python标准库对数据分析的支持相对有限，但是Python具有非常完善的数据科学生态系统，有许多易学易用、高效便捷的第三方开源数据分析工具。除了前面介绍的matplotlib，还有科学计算工具包Numpy（http://www.numpy.org/）/Scipy（https://www.scipy.org）、快速数据分析工具Pandas（https://pandas.pydata.org/）、机器学习工具Scikit-learn（http://scikit-learn.org），以及让深度学习开发更简单的Keras（https://keras.io/，它支持TensorFlow、CNTK和Theano）。如果感兴趣，可以用Pandas直接读取JSON文件数据、并进行格式转换、数据聚合、时间序列分析，结合Scikit-learn可以对收盘价进行回归分析与预测。

16.3 小结

在本章中，你学习了：如何使用网上的数据集；如何处理CSV和JSON文件，以及如何提取你感兴趣的数据；如何使用matplotlib来处理以往的天气数据，包括如何使用模块datetime，以及如何在同一个图表中绘制多个数据系列；如何使用模块json来访问以JSON格式存储的交易收盘价数据，并使用Pygal绘制图形以探索价格变化的周期性，以及如何将Pygal图形组合成数据仪表盘。

有了使用CSV和JSON文件的经验后，你将能够处理几乎任何要分析的数据。大多数在线数据集都可以以这两种格式中的一种或两种下载。学习使用这两种格式为学习使用其他格式的数据做好了准备。

在下一章，你将编写自动从网上采集数据并对其进行可视化的程序。如果你只是将编程作为业余爱好，学会这些技能可以增加乐趣；如果你有志于成为专业程序员，就必须掌握这些技能。