Как обращаться с данными 2

Прежде чем перейти к собственно визуальной части работы с данными, вам необходимо их получить.Именно данные делают визуализацию интересной. Если у вас нет интересных сведений, вы в конечном итоге получите лишь незапоминающуюся диаграмму или красивую, но бесполезную картинку. Где можно найти хорошие данные? Как до них добраться? Вот какие вопросы стоят перед вами на этом этапе.

Когда же данные будут у вас на руках, вам понадобится отформатировать их, чтобы загрузить в компьютерную программу. Возможно, вы получили данные в виде текстового файла с разделителями-запятыми или в виде таблицы Excel, и вам нужно конвертировать их во что-то типа XML. Или наоборот. Или, возможно, необходимые вам данные доступны лишь в разрозненном виде через интернет-приложение, а вы хотите получить таблицу целиком.

Вам нужно научиться добывать и перерабатывать данные, и тогда ваши способности в области визуализации существенно возрастут.

Сбор данных

Данные представляют собой основу любой визуализации. К счастью, есть множество мест, откуда их добывают. Вы можете получить их от экспертов в интересующей вас области, почерпнуть из целого ряда онлайн-приложений или собрать собственноручно.

Данные, предоставленные другими людьми

Это проторенная дорога, особенно если вы дизайнер-фрилансер или работаете в отделе графики крупной организации. В большинстве случаев это очень хорошо — когда кто-то другой выполняет за вас работу по сбору данных. Но вам все равно необходимо быть внимательным. Прежде чем красиво отформатированная таблица попадет к вам в руки, в нее может закрасться масса ошибок.

Если таблицу, с которой вам предстоит работать, вы получили от других людей, знайте, что самые распространенные ошибки, на наличие которых вам нужно будет проверить данные, — это опечатки. Нет ли, например, каких-нибудь потерянных нулей? Может, ваш клиент или поставщик данных хотел вставить их шесть, а не пять. На определенном этапе данные обычно считываются из одного источника и вносятся в другой, скажем, в Excel или в какую-нибудь иную программу табличных вычислений (за исключением тех случаев, когда целиком импортируется текстовый файл с разделителями), и в такие моменты какая-нибудь невинная ошибка легко может преодолеть «барьеры» этапа контроля и попасть в массив данных.

А еще вам необходимо изучить контекст. Не нужно становиться экспертом в той области, из которой были почерпнуты данные, однако вам следует знать, из какого именно первоисточника они взяты и о чем говорят. Это поможет вам создать более качественную графику и поведать через нее более цельную историю. Допустим, к примеру, что перед вами лежат результаты некоего опроса. Когда именно он состоялся? Кто его проводил? Кто отвечал на вопросы? Очевидно, что результаты опроса, проведенного в 1970 году, будут нести несколько иной смысл, нежели результаты сегодняшнего дня.

Самостоятельный поиск данных

Если вам не были предоставлены данные, то пойти и найти их — ваша обязанность, что, конечно, прибавит вам работы, и это плохая новость. Однако есть и хорошая новость: в наши дни становится все проще и проще находить релевантные машиночитаемые данные, которые легко можно загрузить в программу. И вот с чего вам следует начинать поиски.

ПОИСКОВЫЕ СИСТЕМЫ

Как люди сегодня находят что-либо в Сети? Они «гуглят» — ищут с помощью Google. Для этого большого ума не надо, но вы будете удивлены, если я скажу вам, как часто люди шлют мне

письма, спрашивая, не знаю ли я, где можно найти данные по той или иной тематике, — и это в то время, когда даже поверхностный поиск дает вполне удовлетворительный результат. Лично я в подобных ситуациях обычно обращаюсь к Google, а время от времени и к Wolfram|Alpha, поисковой машине с вычислительными возможностями.

▶ Вы можете познакомиться с Wolfram|Alpha на `http://wolframalpha.com`. Эта поисковая машина может оказаться особенно полезной при поиске базовых статистических данных по разным темам.

НАПРЯМУЮ ИЗ ПЕРВОИСТОЧНИКА

Если запрос данных в поисковой машине не дал ничего полезного, попробуйте найти ученых, которые специализируются в той области, сведения из которой вы пытаетесь найти. Иногда они размещают внушительные массивы данных на своих персональных сайтах. Если и это не поможет, просканируйте их публикации и труды — вполне возможно, что найдете там хорошие зацепки. А еще вы можете попробовать связаться с ними по электронной почте, но сначала убедитесь, что они действительно занимались изучением интересующего вас вопроса. Иначе вы просто впустую потратите и свое, и их время.

Различные источники данных вы можете также подсмотреть в диаграммах и графиках, публикуемых в средствах массовой информации, таких как New York Times. Обычно источник данных указан мелким шрифтом где-то на самом графическом объекте. Если его там нет, тогда упоминание о нем должно содержаться в связанной с изображением статье. Такое направление поиска бывает особенно полезным тогда, когда вам на глаза попадается газета или онлайн-издание, использующая данные, в изучении которых вы в настоящий момент заинтересованы. Поищите сайты с подходящей информацией, откуда можно почерпнуть необходимые данные.

Это будет работать не всегда. Когда в письме указано, что вы — репортер такой-то газеты, находить общий язык с собеседниками оказывается немного проще, но в любом случае этот способ стоит попробовать.

УНИВЕРСИТЕТЫ

Когда я стал магистрантом, я часто пользовался академическими ресурсами, которыми располагал, а именно библиотекой. Многие библиотеки существенно продвинулись в своем технологическом оснащении и к настоящему времени располагают огромными архивами данных. Различные факультеты и кафедры также поддерживают массу файлов данных, значительная часть которых находится в открытом доступе. Правда, многие из этих доступных массивов данных предназначены в первую очередь для использования при написании курсовых работ и выполнении домашних заданий. Я предлагаю вам зайти и посмотреть следующие ресурсы.

- Data and Story Library (DASL) (`http://lib.stat.cmu.edu/DASL`) — онлайн-библиотека файлов с данными и историями, иллюстрирующими применение основных статистических методов; относится к Университету Карнеги-Меллон.
- Berkeley Data Lab (`http://sunsite3.berkeley.edu/wikis/datalab`) — часть библиотечной системы Калифорнийского университета в Беркли.

- Массивы статистических данных Калифорнийского университета в Лос-Анджелесе (`http://www.stat.ucla.edu/data`) — часть данных, которые кафедра статистики университета использует при выполнении лабораторных работ и заданий.

ИСТОЧНИКИ ДАННЫХ ОБЩЕГО ХАРАКТЕРА

Сегодня появляется все больше приложений, из которых можно почерпнуть различные данные. Некоторые приложения предлагают большие файлы данных, которые можно скачать безвозмездно или за определенную плату. Часть из них изначально создавалась с мыслью о разработчиках, и доступ к их данным осуществляется через интерфейсы программирования приложений (API). Это позволяет вам использовать данные различных сервисов, таких как, например, Twitter, и интегрировать их в свои собственные приложения. Далее представлены некоторые рекомендуемые ресурсы.

- Freebase (`http://www.freebase.com`) — база данных, появившаяся в результате усилий большого сообщества и предоставляющая сведения главным образом о людях, местах и товарах. Немного напоминает «Википедию», но информация в ней более структурирована. Можете пользоваться ею для скачивания информации или в качестве базы данных типа back-end* для вашего приложения.
- Infochimps (`http://infochimps.org`) — рынок с платными и бесплатными массивами данных. Доступ к некоторым из них можно получить через их API.
- Numbrary (`http://numbrary.com`) — онлайн-каталог данных, по большей части связанных с национальной статистикой.
- AggData (`http://aggdata.com`) — еще одно хранилище платных массивов данных, сфокусированных в основном на составлении исчерпывающих перечней торговых объектов.
- Amazon Public Data Sets (`http://aws.amazon.com/publicdatasets`) — особого роста здесь не наблюдается, но тем не менее ресурс содержит весьма солидные подборки научных данных.
- Wikipedia (`http://wikipedia.org`) — в этой энциклопедии, поддерживаемой на общественных началах, можно найти множество небольших подборок данных в виде HTML-таблиц.

ТЕМАТИЧЕСКИЕ ДАННЫЕ

Помимо поставщиков данных общего характера существует также множество сайтов по конкретным тематикам, предлагающих уйму бесплатных данных. Далее следует лишь маленькая выборка того, что доступно по тем или иным направлениям.

---

* Back-end — база данных, к которой пользователи обращаются не напрямую или путем низкоуровневых манипуляций (например SQL-запросов), а через специально разработанное приложение. *Прим. пер.*

География

- У вас есть программа для локализации объектов, но нет географических данных? Вам повезло. В вашем распоряжении огромное количество географических файлов, в том числе в векторном формате (шейп-файлов).
- TIGER (`http://www.census.gov/geo/www/tiger`) — сайт Бюро переписи населения США предоставляет, возможно, самые дорогие, но и самые детальные базы данных об автострадах, железных дорогах, реках и почтовых индексах, которые только можно найти.
- OpenStreetMap (`http://www.openstreetmap.org`) — один из лучших примеров баз данных, созданных на общественных началах.
- Geocommons (`http://www.geocommons.com`) — это одновременно и база данных, и картографическое приложение.
- Flickr Shapefiles (`http://www.flickr.com/services/api`) — географические границы в том виде, в каком они представлены пользователями Flickr.

Спорт

Люди любят спортивную статистику, и вы можете получить доступ к данным этого типа за несколько последних десятилетий. Найти их можно на сайте журнала Sports Illustrated и на сайтах соответствующих спортивных команд и организаций, а также на сайтах, посвященных сугубо спортивной статистике.

- Basketball Reference (`http://www.basketball-reference.com`) — источник исчерпывающих данных и сведений вплоть до прямых репортажей с игр Национальной баскетбольной ассоциации.
- Baseball DataBank (`http://baseball-databank.org`) — сверхосновательный сайт, с которого можно скачать исчерпывающие наборы данных о бейсболе.
- DatabaseFootball (`http://www.databasefootball.com`) — здесь можно посмотреть данные об играх Национальной футбольной лиги США по командам, по игрокам и по сезонам.

Мир в целом

Несколько заслуживающих внимания международных организаций также поддерживают базы данных о мире в целом, главным образом по таким показателям, как здравоохранение и уровень развития. И все же, работая с ними, вам придется кое-что отсеивать, так как многие массивы данных не очень хорошо структурированы. Нелегко получить стандартизированные данные, собирая их из разных стран разными методами.

- Global Health Facts (`http://www.globalhealthfacts.org`) — данные из области здравоохранения в разных странах мира.
- UNdata (`http://data.un.org`) — агрегатор данных из разных источников со всего мира.

- World Health Organization (`http://www.who.int/research/en`) — снова обилие данных из области здравоохранения, таких как уровень смертности и ожидаемая продолжительность жизни.
- OECD Statistics (`http://stats.oecd.org`) — один из крупнейших ресурсов, посвященных экономическим показателям.
- World Bank (`http://data.worldbank.org`) — данные по сотням показателей, причем представленные в очень удобном для разработчиков виде.

Государственные и политические организации

В последние годы все больше говорится о доступности информации и прозрачности, а потому многие ведомства регулярно предоставляют сведения о своей работе, а организации, такие как Sunlight Foundation, стимулируют разработчиков и дизайнеров использовать эти данные. На протяжении некоторого времени государственные организации сами публиковали информацию о себе, но с момента запуска Data.gov большинство этих данных уже доступны в одном месте. А еще вы можете найти массу неправительственных сайтов, чья цель — заставить политиков отчитываться перед гражданами.

- Бюро переписи населения (`http://www.census.gov`) — богатый источник демографических данных.
- Data.gov (`http://data.gov`) — каталог данных, поставляемых органами правительства. Хотя и относительно новый, но пополняется из большого количества источников.
- Data.gov.uk (`http://data.gov.uk`) — британский эквивалент data.gov.
- DataSF (`http://datasf.org`) — данные, связанные конкретно с Сан-Франциско.
- NYC DataMine (`http://nyc.gov/data`) — ресурс, аналогичный предыдущему, только о Нью-Йорке.
- Follow the Money (`http://www.followthemoney.org`) — большой набор инструментов и массивов данных для отслеживания денежных потоков в политической жизни страны.
- OpenSecrets (`http://www.opensecrets.org`) — еще один ресурс, поставляющий сведения о государственных расходах и лоббировании.

Извлечение данных

Очень часто бывает так, что необходимые данные найти можно, но есть одна загвоздка. Они находятся не в одном месте и не в одном файле, а разбросаны по разным HTML-страницам или даже по разным сайтам. Как вам быть?

Самый лобовой, но и самый затратный по времени метод — это посетить по очереди все страницы и вручную ввести интересующие вас крупицы данных в свою таблицу. Если страниц всего несколько, то, конечно, проблем с этим не возникнет.

Ну а что делать, если страниц тысячи? Такая работа займет слишком много времени. Даже если их не тысячи, а сотни — процесс уже становится утомительным. Было бы гораздо лучше, если б вы

могли автоматизировать процесс. Для этого и применяется *скрейпинг (анализ и извлечение) данных*. Вы создаете некий код для автоматического посещения определенного перечня страниц, получения конкретного контента с этих страниц и сохранения его в базе данных или в текстовом файле.

ПРИМЕР: АНАЛИЗ САЙТА

Лучший способ научиться извлекать необходимые данные — это сразу же перейти к примерам. Скажем, вы хотели скачать данные по температуре за прошедший год, но у вас не получается найти источник, который предоставил бы вам все сведения за нужный отрезок времени или по нужному городу. Вы можете пойти на любой сайт о погоде, но самое большее, что вы там найдете, — подробный прогноз на ближайшие десять дней. Это совсем не то, что вам нужно. Вам нужны реальные показатели температуры из прошлого, а не прогнозы на будущее.

К счастью, сайт Weather Underground предоставляет исторические данные о погоде. И плохая новость: на одной странице сведения можно получить только за один день.

Чтобы разговор стал конкретнее, посмотрите температуру в городе Буффало, США. Зайдите на сайт Weather Underground и введите BUF в окно поиска. После этого вы попадете на страницу о погоде в районе расположенного в Буффало аэропорта Ниагара Интернешнл (рис. 2.1).

ПРИМЕЧАНИЕ

Хотя самый гибкий метод извлечения необходимых вам данных — это программирование, вы можете попробовать еще и такие инструменты, как Needlebase и PDF-конвертер Able2Extract. Используйте их по назначению — они способны существенно сэкономить вам время.

▶ Посетите сайт Weather Underground: `http://wunderground.com`.

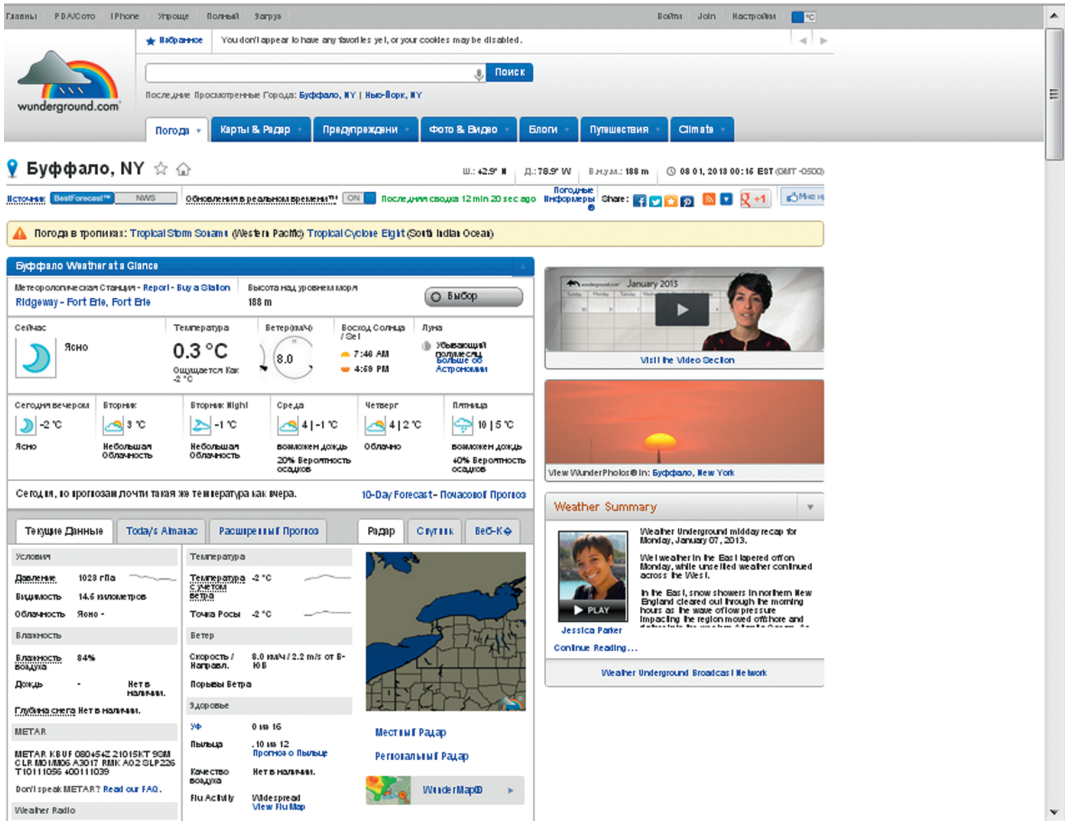


Рис. 2.1. Температура в Буффало по данным Weather Underground

Рис. 2.2. Выпадающее меню для получения исторических сведений по конкретной дате

Вверху страницы дается информация о температуре в настоящий момент, пятидневный прогноз погоды и некоторые другие детали о сегодняшнем дне. Прокрутите страницу вниз, поближе к середине, и найдите панель «История и альманах» («History & Almanac»), как показано на рис. 2.2. Обратите внимание на выпадающее меню, в котором вы можете выбрать определенную дату.

Настройте меню на показ данных за 8 января 2013 года и нажмите кнопку «Просмотр» (View). Откроется другая страница, где вам будут представлены подробные данные о погоде в выбранный вами день (рис. 2.3).

Там будет и температура, и градусо-день, и влажность, и осадки, и множество других данных, но вас интересует только максимальная температура в тот день. Ее вы можете найти во втором столбце, во второй строчке сверху вниз. Восьмого января 2013 года максимальная температура в Буффало составляла 0 °C.

Получить этот один показатель вышло довольно легко. Но как можно добыть сведения о максимальной температуре за каждый день на протяжении всего 2012 года? Самый простой, лобовой метод — это поочередно менять дату в выпадающем меню. Повторите процедуру 365 раз — и готово.

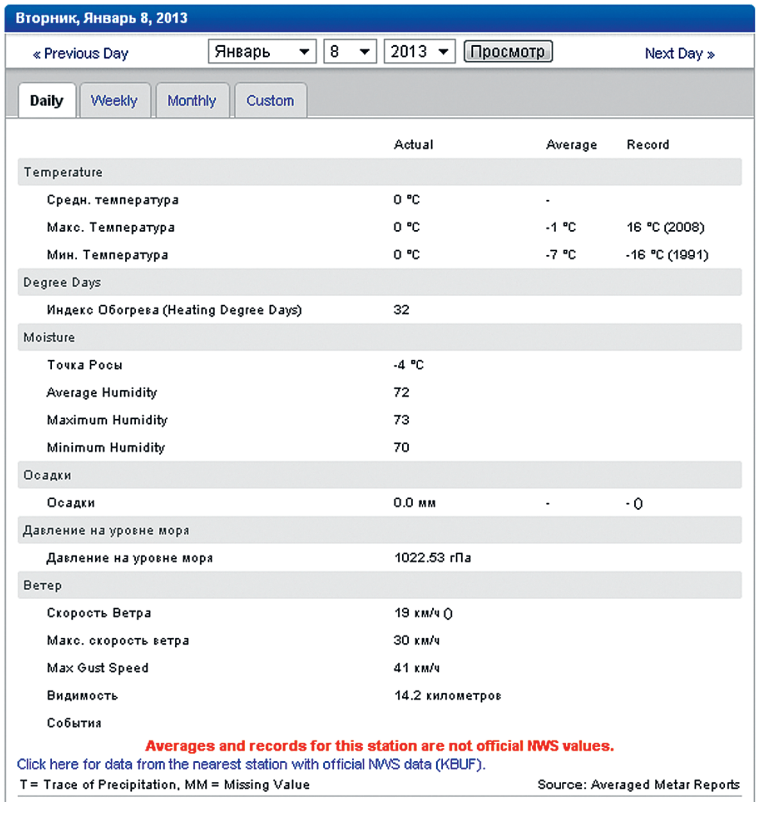


Рис. 2.3. Температурные данные за один конкретный день

Разве это не кажется вам увлекательным? Нет? Тогда вы можете ускорить процесс с помощью небольшого кода и кое-какого ноу-хау. Для этого обратитесь к языку программирования Python и к библиотеке под названием Beautiful Soup, созданной Леонардом Ричардсоном (Leonard Richardson).

В течение следующих нескольких абзацев вы впервые в жизни ощутите вкус создания кодов. Если у вас есть опыт в программировании, следующий этап вы пройдете относительно быстро. Однако если у вас такого опыта нет, не беспокойтесь — я проведу вас через всю процедуру шаг за шагом. Многие люди предпочитают оставаться в рамках интерфейсов, где все делается одним кликом, но доверьтесь мне. Достаточно освоить самую малость программистских умений, и вы сможете открыть для себя целый мир новых возможностей в работе с данными. Готовы? Поехали.

Во-первых, вам необходимо убедиться, что у вас в компьютере установлено подходящее программное обеспечение. Если вы работаете под Mac OS X, тогда Python у вас уже должен быть инсталлирован. Чтобы начать, откройте приложение Terminal и введите: **python** (рис. 2.4).

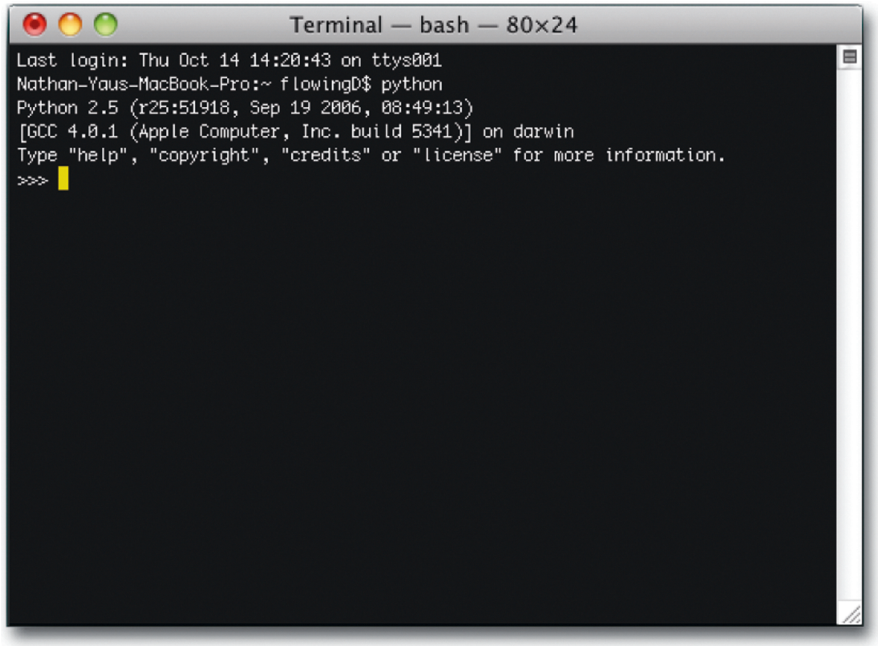


Рис. 2.4. Запуск Python в Mac OS X

▶ Для скачивания и установки Python зайдите на `http://python.org`. Не беспокойтесь, это не так уж трудно.

Если вы работаете под Windows, вы можете зайти на соответствующий сайт и, следуя инструкциям, скачать и установить Python у себя.

Затем вам необходимо скачать Beautiful Soup, библиотеку, которая поможет вам считывать веб-страницы легко и быстро. Поместите файл Beautiful Soup Python (он имеет расширение .py) в директории, в которой вы собираетесь сохранить и свой код. Если вы знаете путь до Python, можете поместить Beautiful Soup также у себя в библиотеке — на результате это никак не скажется.

▶ Чтобы скачать Beautiful Soup, зайдите на `http://www.crummy.com/software/BeautifulSoup`. Скачайте ту версию, которая подходит к используемой вами версии Python.

После того как вы установите Python и скачаете Beautiful Soup, создайте файл в вашем любимом текстовом редакторе или в редакторе кодов и сохраните его как `get-weather-data.py`. Теперь вы можете заняться написанием кода.

Первое, что вам необходимо сделать, — это загрузить страницу, на которой представлены исторические сведения о погоде. URL у страницы с информаций о погоде в Буффало за 1 октября 2010 выглядит так:

```
www.wunderground.com/history/airport/KBUF/2010/10/1/DailyHistory.html?req_city=NA&req_
state=NA&req_statename=NA
```

Если в этом адресе вы удалите все, что следует за .html, страница будет по-прежнему загружаться, так что избавьтесь от этих лишних символов. На данный момент они вас не интересуют. Должно получиться:

```
www.wunderground.com/history/airport/KBUF/2010/10/1/DailyHistory.html
```

В URL дата указана как /2010/10/1. Используя выпадающее меню, замените ее на новую: на 1 января 2009 года, так как вы собираетесь извлекать данные по температуре за весь 2009 год. После этого URL должен выглядеть так:

```
www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.html
```

В адресе все осталось по-прежнему за исключением той части, в которой указана дата. Теперь она записана так: /2009/1/1/. Интересно. А как можно загрузить страницу со сведениями о 2 января 2009 года без использования выпадающего меню? Просто измените параметр даты так, чтобы URL выглядел следующим образом:

```
www.wunderground.com/history/airport/KBUF/2009/1/2/DailyHistory.html
```

Загрузите новый URL в ваш веб-браузер, и вы получите историческую информацию о 2 января 2009 года. Итак, все, что вам необходимо сделать, чтобы узнать погоду за определенную дату, — модифицировать URL страницы Weather Underground. Запомните это — мы еще вернемся к данному вопросу.

Теперь загрузите одну-единственную страницу с помощью Python, используя библиотеку urllib2, для чего импортируйте ее, введя следующую строчку кода:

```
import urllib2
```

Чтобы загрузить страницу с данными о погоде за 1 января с помощью Python, используйте функцию `urlopen`.

```
page = urllib2.urlopen("www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.
html")
```

Таким образом вы загрузите весь HTML-файл, на который указывает URL в переменной страницы. Следующим шагом будет извлечение интересующего вас значения максимальной температуры из этого HTML. Beautiful Soup сделает выполнение данной задачи намного проще. Вслед за `urllib2` импортируйте Beautiful Soup следующим образом:

```
from BeautifulSoup import BeautifulSoup
```

В конце вашего файла используйте Beautiful Soup, чтобы прочитать страницу, то есть произвести ее анализ.

```
soup = BeautifulSoup(page)
```

Не вдаваясь во все детали, скажу, что эта строчка кода прочитывает HTML, который представляет по сути одну длинную строку, а затем сохраняет элементы страницы, такие как заголовок или изображения, в удобном для работы виде.
Например, если вы хотите найти все изображения на странице, вы можете использовать вот что:

```
images = soup.findAll('img')
```

В результате вы получите перечень всех изображений на странице Weather Underground, отображаемых с помощью HTML-тега `<img />`. Вам нужно первое изображение на странице? Сделайте так:

```
first_image = images[0]
```

Хотите второе изображение? Измените ноль на единицу. Если вам нужно значение `src` в первом теге `<img />`, тогда используйте:

```
src = first_image['src']
```

Ладно, вас ведь не интересуют изображения. Вы просто хотите одно-единственное значение: максимальную температуру в городе Буффало, штат Нью-Йорк, в день 1 января 2009 года. Тогда она составляла 26 °F (или –3 °C). Чтобы найти в «супе» данное значение, нужно будет потрудиться чуть подольше, чем когда вы искали изображения, но метод применяется все тот же. Вам всего лишь нужно выяснить, что именно следует вставить в `findAll()`, а потому просмотрите исходный HTML.

Это легко делается в любом из популярных веб-браузеров. В Firefox, например, необходимо открыть меню «Вид» (View) и выбрать «Исходный код страницы» (Page Source). Появится окно с HTML-кодом текущей страницы, как это показано на рис. 2.5.

ПРИМЕЧАНИЕ

Beautiful Soup снабжена хорошей документацией и наглядными примерами, поэтому если что-то из описываемого здесь вас смущает, я горячо рекомендую вам поискать дополнительную информацию на том же сайте Beautiful Soup, с которого вы скачали библиотеку.

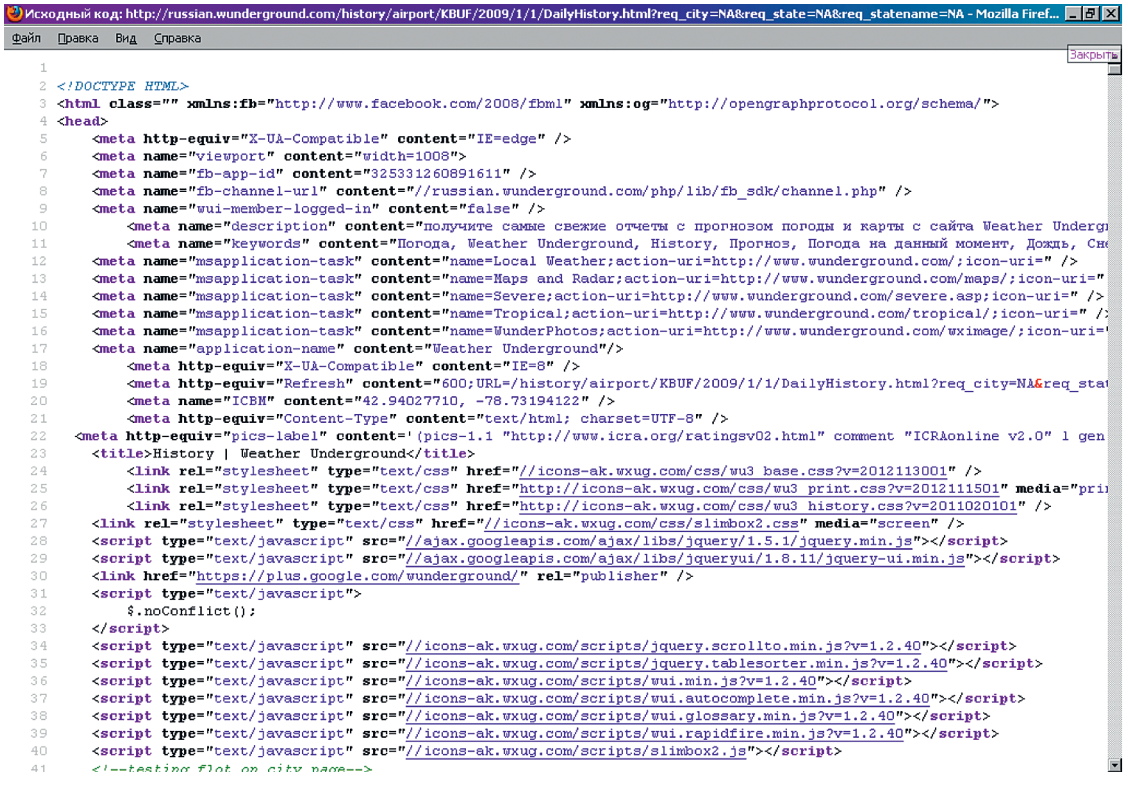


Рис. 2.5. Исходный HTML-код страницы Weather Underground

Прокрутите вниз до того места, где показана средняя температура, или просто воспользуйтесь функцией поиска в браузере, что будет быстрее. Отметьте 26 (–3, если по Цельсию). Это именно то, что вы хотите извлечь.

Строка замыкается тегом с классом nobr. Это и есть ваш ключ. Вы теперь можете найти все элементы на странице с классом nobr.

```
nobrs = soup.findAll(attrs={"class":"nobr"})
```

Как и в предыдущем примере, это дает вам в руки перечень всех случаев употребления nobr. Вас интересует шестой из них, который вы найдете с помощью следующей строчки:

```
print nobrs[5]
```

Это выдаст вам весь элемент, но вас интересует только 26 (–3). Внутри тега с классом nobr есть другой тег , а за ним идет 26 (–3). Вот то, что вам необходимо использовать.

```
dayTemp = nobrs[5].span.string
print dayTemp
```

Вуаля! Вы впервые в жизни извлекли нужное вам значение из HTML-кода веб-страницы. Следующий шаг — проанализировать все страницы за 2009 год и извлечь из них необходимые данные. Для этого вернитесь к первоначальному URL.

```
www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.html
```

Помните, как вы вручную изменили адрес, чтобы получить сведения о той дате, которая вас интересовала? Приведенный выше код относится к 1 января 2009 года. Если вам нужна страница за 2 января 2009 года, просто измените ту часть URL, в которой указана дата, на соответствующую. Чтобы получить данные за все дни 2009 года, загрузите все месяцы (с 1-го по 12-й), а затем загрузите каждый день каждого месяца. Ниже представлен весь скрипт с комментариями. Сохраните его в вашем файле get-weather-data.py.

```
import urllib2
from BeautifulSoup import BeautifulSoup

# Create/open a file called wunder.txt (which will be a comma-delimited file)
f = open('wunder-data.txt', 'w')

# Iterate through months and day
for m in range(1, 13):
   for d in range(1, 32):

      # Check if already gone through month
      if (m == 2 and d > 28):
         break
      elif (m in [4, 6, 9, 11] and d > 30):
         break
```

```
      # Open wunderground.com url
      timestamp = '2009' + str(m) + str(d)
      print "Getting data for " + timestamp
      url = "http://www.wunderground.com/history/airport/KBUF/2009/" +
str(m) + "/" + str(d) + "/DailyHistory.html"
      page = urllib2.urlopen(url)

      # Get temperature from page
      soup = BeautifulSoup(page)
      # dayTemp = soup.body.nobr.b.string
      dayTemp = soup.findAll(attrs={"class":"nobr"})[5].span.string

      # Format month for timestamp
      if len(str(m)) < 2:
         mStamp = '0' + str(m)
      else:
         mStamp = str(m)

      # Format day for timestamp
      if len(str(d)) < 2:
         dStamp = '0' + str(d)
      else:
         dStamp = str(d)

      # Build timestamp
      timestamp = '2009' + mStamp + dStamp

      # Write timestamp and temperature to file
      f.write(timestamp + ',' + dayTemp + '\n')

# Done getting data! Close file.
f.close()
```

Вы наверняка узнали первые две строчки кода, с помощью которых вы импортировали необходимые библиотеки, — urllib2 и BeautifulSoup.

```
import urllib2
from BeautifulSoup import BeautifulSoup
```

Далее создается текстовый файл под названием `wunder-data-txt` с правами на запись, используя метод `open()`. Все данные, которые вы извлечете, будут сохраняться в этом текстовом файле, расположенном в той же директории, куда вы сохранили свой скрипт.

```
# Create/open a file called wunder.txt (which will be a comma-delimited file)
f = open('wunder-data.txt', 'w')
```

С помощью следующей строчки кода, используя цикл for, компьютеру отдается команда просмотреть каждый месяц. Число, обозначающее месяц, указывается в переменной m. Следующий цикл сообщает компьютеру, что надо посетить каждый день каждого месяца. Каждый отдельный день месяца указывается в переменной d.

```
# Iterate through months and day
for m in range(1, 13):
   for d in range(1, 32):
```

▶ За дополнительной информацией о том, как работают циклы и происходит итерация, обратитесь к документации Python, расположенной по адресу http://docs.python.org/reference/compound_stmts.html

Обратите внимание на то, что для повторения операции по дням используется range (1, 32). Это означает, что повтор будет производиться для каждого числа, начиная с 1 и по 31. Однако не в каждом месяце в году есть 31 день. В феврале их 28; в апреле, июне, сентябре и ноябре — по 30. Температурных показателей за 31 апреля нет, потому что такого дня не существует. Обратите внимание на то, о каком месяце идет речь, и действуйте сообразно этому. Если текущий месяц — февраль, и число больше 28, прервитесь и переходите к следующему месяцу. Если вы хотите извлечь данные по нескольким годам, тогда вам необходимо использовать дополнительный оператор if, чтобы справиться с високосными годами.

Точно так же, если речь идет не о феврале, а об апреле, июне, сентябре или ноябре, и если значение текущего дня больше 30, нужно перейти к следующему месяцу.

```
# Check if already gone through month
if (m == 2 and d > 28):
   break
elif (m in [4, 6, 9, 11] and d > 30):
   break
```

И снова следующие строчки кода должны показаться вам знакомыми. Вы использовали их для извлечения данных из одной конкретной страницы сайта Weather Underground. Отличие состоит в переменной месяца и дня в URL. Ее просто нужно менять для каждого дня, а не оставлять статичной — все прочее тут без изменений. Загрузите страницу, используя библиотеку urllib2, произведите анализ контента с помощью Beautiful Soup, а затем извлеките максимальную температуру, но ищите шестое появление класса nobr.

```
# Open wunderground.com url
timestamp = '2009' + str(m) + str(d)
print "Getting data for " + timestamp
url = "http://www.wunderground.com/history/airport/KBUF/2009/" +
str(m) + "/" + str(d) + "/DailyHistory.html"
page = urllib2.urlopen(url)

# Get temperature from page
soup = BeautifulSoup(page)
# dayTemp = soup.body.nobr.b.string
dayTemp = soup.findAll(attrs={"class":"nobr"})[5].span.string
```

Предпоследний кусок кода отвечает за составление метки времени исходя из года, месяца и дня. Метка времени дается в формате «ггггммдд». Здесь вы можете задать любой формат, но на данном этапе чем проще — тем лучше.

```
# Format day for timestamp
if len(str(d)) < 2:
   dStamp = '0' + str(d)
else:
   dStamp = str(d)

# Build timestamp
timestamp = '2009' + mStamp + dStamp
```

И наконец, в файл `'wunder-data.txt'` с помощью метода `write()` добавляются температура и временна́я отметка.

```
# Write timestamp and temperature to file
f.write(timestamp + ',' + dayTemp + '\n')
```

По окончании работы со всеми месяцами и днями применяется `close()`.

```
# Done getting data! Close file.
f.close()
```

Осталось лишь запустить код, что вы и сделаете в своем терминале с помощью вот такой команды:

```
$ python get-weather-data.py
```

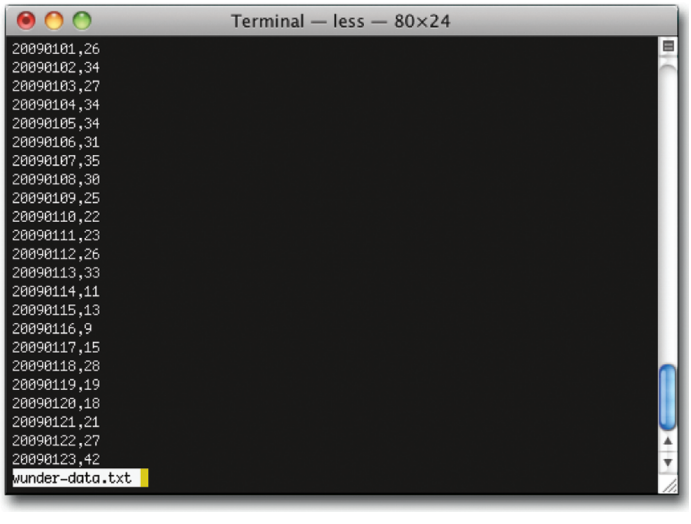


Рис. 2.6. Извлеченные данные о температуре за год

Прогон займет некоторое время, так что наберитесь терпения. По сути, в процессе выполнения программы ваш компьютер загрузит по очереди 365 страниц — по одной на каждый день 2009 года. Когда выполнение скрипта завершится, у вас в рабочем каталоге появится файл под названием `wunder-data.txt`. Откройте его, и там вы найдете нужные вам данные в формате с разделителями-запятыми. В первой колонке вы увидите отметку о времени, во второй колонке — температуру. Выглядеть все будет примерно так, как показано на рис. 2.6.

ГЕНЕРАЛИЗАЦИЯ ПРИМЕРА

Хотя вы всего лишь извлекли данные о погоде с сайта Weather Underground, этот процесс можно генерализировать для использования в работе с другими источниками. Извлечение данных, как правило, состоит из трех этапов:

1. Выявление паттернов.
2. Произведение итерации.
3. Сохранение данных.

В рассмотренном выше примере вы, чтобы получить данные по температуре, должны были найти два паттерна. Первый из них содержался в URL, а второй — в загружаемой веб-странице. Чтобы загрузить страницу о другом дне в 2009 году, вы изменяли часть URL с указанием дня и месяца. Значение температуры содержалось в шестом случае употребления класса `nobr` на странице. Если в URL нет очевидных паттернов, попробуйте сообразить, как можно получить URL всех страниц, которые вы хотите подвергнуть анализу, чтобы извлечь необходимые данные. Возможно, у сайта есть карта страниц, или вы можете пробежаться по индексу с помощью поиска. Так или иначе, в конечном итоге в вашем распоряжении должны оказаться URL всех страниц с данными.

После того как вы найдете паттерны, приступайте к итерации. Иными словами, вам нужно будет посетить — программным способом — все страницы по очереди, загрузить их и проанализировать. В предыдущем примере вы сделали это с помощью библиотеки Beautiful Soup, которая делает парсинг* XML и HTML в Python довольно легким. Если вы выберете другой язык программирования, для него, вероятно, найдется похожая библиотека.

И наконец, вам понадобится где-то сохранить полученные данные. Самое простое решение — это записать их в текстовый файл, в котором значения разделены запятыми. Но если у вас уже создана некая база данных, вы можете записать полученные значения в нее.

Задача становится несколько сложнее, когда приходится иметь дело с веб-страницами, использующими для загрузки данных JavaScript, но по своей сути процесс остается таким же.

Форматирование данных

Разные инструменты для визуализации используют различные форматы данных. Структура зависит от того, какую историю вы хотите рассказать. Поэтому чем больше гибкости вы проявите в работе со структурой ваших данных, тем больше у вас будет возможностей. Используйте приложения для форматирования данных, добавьте к этому немного ноу-хау в области

* Парсинг — процесс сбора контента (обычно с сайта) и его синтаксического анализа с целью перевода данных в более удобную для обработки структуру; осуществляется с помощью специальной программы — парсера. *Прим. пер.*

программирования, и у вас появится возможность отобразить данные в любом желаемом вами формате так, чтобы они отвечали вашим нуждам в конкретный момент.

Самый легкий путь — это найти программиста, который сможет отформатировать все ваши данные и произвести их анализ, но в таком случае вы всегда будете зависеть от кого-то другого. Подобная зависимость становится особенно очевидной на ранних этапах любого проекта, когда итерация и изучение данных имеют ключевое значение для создания дельной визуализации. Честно говоря, если бы я занимался подбором кадров, я предпочел бы нанять кого-то, кто знает, как работать с данными, а не того, кому каждый раз на начальном этапе проекта будет нужна помощь со стороны.

ЧТО Я УЗНАЛ О ФОРМАТИРОВАНИИ

Когда я только начал изучать статистику в средней школе, данные нам всегда предоставлялись в аккуратном табличном формате. Все, что от меня требовалось сделать, — это вставить несколько чисел в таблицу Excel или в мой потрясающий графический калькулятор (что было предпочтительнее, так как позволяло играть в «Тетрис», делая вид, будто я выполняю классное задание). На протяжении всех лет учебы в бакалавриате ситуация оставалась практически без изменений. Поскольку я занимался изучением методов и теорем анализа, мои преподаватели не тратили лишнего времени на работу с сырыми, неподготовленными данными. Создавалось впечатление, будто данные всегда существуют в удобном для работы виде.

Это и вполне понятно, учитывая ограничения во времени и т. п. Но в магистратуре я осознал, что в реальном мире данные почти никогда не встречаются в нужном вам формате. Какие-то значения отсутствуют вовсе, другие появляются без какого-либо контекста или сопровождаются нелогичными, противоречивыми подписями и опечатками. Очень часто необходимые данные разбросаны по разным таблицам, а вам нужно, чтобы они все оказались в одной-единственной, причем выстроенными по одному какому-то показателю, такому как имя или уникальный идентификатор.

С этой же проблемой я столкнулся и тогда, когда начал заниматься визуализацией. И проблема становилась тем острее, чем сильнее мне хотелось добиться большего с помощью имеющихся в наличии данных. Сегодня для меня уже не является чем-то необычайным то, что на получение данных в нужном мне формате я трачу столько же времени, сколько собственно на выполнение работы по их визуализации. Иногда на наведение порядка в данных у меня уходит даже больше времени. Поначалу это может показаться странным, но вы убедитесь, что создание графики становится намного легче, когда данные у вас организованы так четко, как это было во времена вводного курса в статистику в средней школе.

Далее вы познакомитесь с различными форматами данных, с доступными инструментами, с помощью которых вы сможете работать с этими форматами, и, наконец, обретете некоторые знания в области программирования, применяя те же принципы, что и при извлечении данных в предыдущем примере.

Форматы данных

Большинство людей привыкли работать с данными в Excel. И в этом нет ничего плохого, если вы будете делать в данной программе все — от анализа до визуализации. Но если вам понадобится выйти за эти рамки, тогда вам придется ознакомиться и с другими форматами данных. Смысл этих форматов в том, чтобы сделать ваши данные машиночитаемыми. Иными словами, структурировать ваши данные таким образом, чтобы их мог понять компьютер. Какой именно формат данных использовать, будет зависеть от инструмента и целей визуализации, но в большинстве случаев ваша база данных окажется в каком-то из следующих трех форматов: текст с разделителями, объектная нотация JavaScript или расширяемый язык разметки.

ТЕКСТ С РАЗДЕЛИТЕЛЯМИ

Текст с разделителями знаком большинству людей. Вы только что, в примере с извлечением данных, создали текстовый файл с разделителями-запятыми. Если вы привыкли думать о наборах данных в виде строчек и столбцов, то в текстовом файле с разделителями позиции будут разграничены именно разделителем. В файле с разделителями-запятыми это запятые. Помимо них в роли разделителя может выступать символ табуляции, а также пробелы, точки с запятыми, двоеточия, слеши — все что угодно, хотя чаще всего применяются запятые и табуляция.

Текст с разделителями-запятыми имеет очень широкое применение и читается большинством программ с таблицами, таких как Excel или Google Documents. Электронные таблицы вы также можете экспортировать в виде текста с разделителями. Если у вас в книге Excel много таблиц, тогда у вас и много файлов с разделителями, если вами не задано что-то иное.

Этот формат удобен также для обмена данными с другими людьми, так как он не зависит от какой-либо конкретной программы.

▶ Чтобы ознакомиться с полной спецификацией JSON, посетите `http://json.org`. Вам не нужно штудировать все детали, но данный ресурс может оказаться полезен, если вдруг вы не сумеете разобраться в данных, представленных в этом формате.

ОБЪЕКТНАЯ НОТАЦИЯ JAVASCRIPT (JSON)

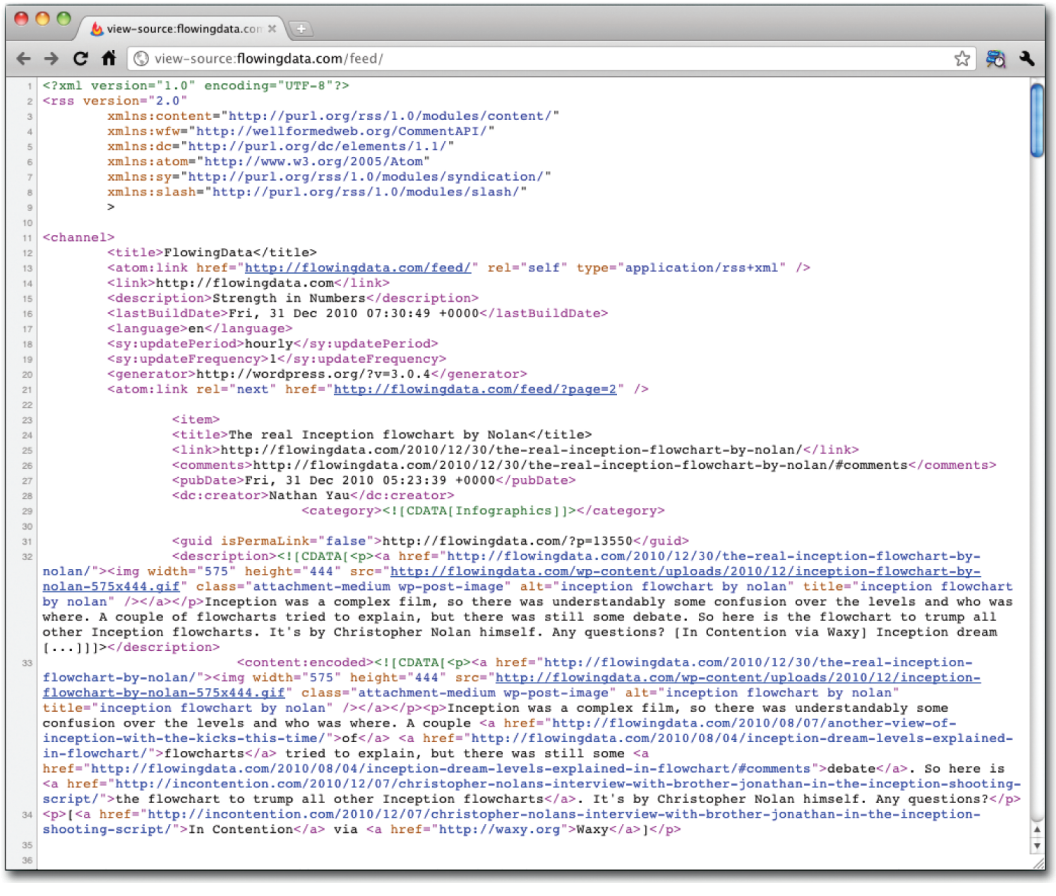
Это один из самых распространенных форматов, который предлагают веб-API. Он разработан таким образом, чтобы его могли читать и машины, и люди, хотя если вам придется работать с таким файлом и внимательно всматриваться в него, вы вскоре окосеете. Он основан на нотации JavaScript, но является языконезависимым. Для JSON существует множество спецификаций, но для начала работы вам хватит понимания азов.

JSON работает с ключевыми словами и значениями и относится к ним как к объектам. Если вы конвертируете данные в формате JSON в значения, разделенные запятыми (CVS), каждый объект станет строчкой.

Далее в этой книге вы найдете множество приложений, языков и библиотек, принимающих JSON в качестве формата вводимых данных. Если вы собираетесь создавать диаграммы и графики для Сети, вы наверняка столкнетесь с этим форматом.

РАСШИРЯЕМЫЙ ЯЗЫК РАЗМЕТКИ (XML)

Другой распространенный в сети формат — это XML. Его часто применяют для передачи данных через различные API. Существует множество разных типов XML и спецификаций, но на самом базовом уровне его можно охарактеризовать как формат текстового документа со значениями, заключенными в теги. Например, фид RSS (Really Simple Syndication) — который люди используют, подписываясь на блоги вроде FlowingData, — по сути, представляет собой XML-файл (рис. 2.7).




```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0"
        xmlns:content="http://purl.org/rss/1.0/modules/content/"
        xmlns:wfw="http://wellformedweb.org/CommentAPI/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:atom="http://www.w3.org/2005/Atom"
        xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
        xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
        >

<channel>
        <title>FlowingData</title>
        <atom:link href="http://flowingdata.com/feed/" rel="self" type="application/rss+xml" />
        <link>http://flowingdata.com</link>
        <description>Strength in Numbers</description>
        <lastBuildDate>Fri, 31 Dec 2010 07:30:49 +0000</lastBuildDate>
        <language>en</language>
        <sy:updatePeriod>hourly</sy:updatePeriod>
        <sy:updateFrequency>1</sy:updateFrequency>
        <generator>http://wordpress.org/?v=3.0.4</generator>
        <atom:link rel="next" href="http://flowingdata.com/feed/?page=2" />

                <item>
                <title>The real Inception flowchart by Nolan</title>
                <link>http://flowingdata.com/2010/12/30/the-real-inception-flowchart-by-nolan/</link>
                <comments>http://flowingdata.com/2010/12/30/the-real-inception-flowchart-by-nolan/#comments</comments>
                <pubDate>Fri, 31 Dec 2010 05:23:39 +0000</pubDate>
                <dc:creator>Nathan Yau</dc:creator>
                                <category><![CDATA[Infographics]]></category>

                <guid isPermaLink="false">http://flowingdata.com/?p=13550</guid>
                <description><![CDATA[<p><a href="http://flowingdata.com/2010/12/30/the-real-inception-flowchart-by-
nolan/"><img width="575" height="444" src="http://flowingdata.com/wp-content/uploads/2010/12/inception-flowchart-by-
nolan-575x444.gif" class="attachment-medium wp-post-image" alt="inception flowchart by nolan" title="inception flowchart
by nolan" /></a></p>Inception was a complex film, so there was understandably some confusion over the levels and who was
where. A couple of flowcharts tried to explain, but there was still some debate. So here is the flowchart to trump all
other Inception flowcharts. It's by Christopher Nolan himself. Any questions? [In Contention via Waxy] Inception dream
[...]]]></description>
                        <content:encoded><![CDATA[<p><a href="http://flowingdata.com/2010/12/30/the-real-inception-
flowchart-by-nolan/"><img width="575" height="444" src="http://flowingdata.com/wp-content/uploads/2010/12/inception-
flowchart-by-nolan-575x444.gif" class="attachment-medium wp-post-image" alt="inception flowchart by nolan"
title="inception flowchart by nolan" /></a></p><p>Inception was a complex film, so there was understandably some
confusion over the levels and who was where. A couple <a href="http://flowingdata.com/2010/08/07/another-view-of-
inception-with-the-kicks-this-time/">of</a> <a href="http://flowingdata.com/2010/08/04/inception-dream-levels-explained-
in-flowchart/">flowcharts</a> tried to explain, but there was still some <a
href="http://flowingdata.com/2010/08/04/inception-dream-levels-explained-in-flowchart/#comments">debate</a>. So here is
<a href="http://incontention.com/2010/12/07/christopher-nolans-interview-with-brother-jonathan-in-the-inception-shooting-
script/">the flowchart to trump all other Inception flowcharts</a>. It's by Christopher Nolan himself. Any questions?</p>
<p>[<a href="http://incontention.com/2010/12/07/christopher-nolans-interview-with-brother-jonathan-in-the-inception-
shooting-script/">In Contention</a> via <a href="http://waxy.org">Waxy</a>]</p>
```

Рис. 2.7. Фрагмент RSS-фида FlowingData

В RSS перечислены недавно опубликованные статьи. Они заключены в теги `<item></item>`, и по каждой из них дается заголовок, описание, автор, дата публикации и еще кое-какие другие детали.

Производить парсинг XML относительно несложно, если использовать такие библиотеки, как Beautiful Soup для Python. После прочтения следующих разделов книги вы начнете лучше разбираться в XML, а также в CSV и JSON.

Инструменты форматирования

Еще пару лет назад для форматирования данных всегда приходилось писать скрипты. Стоит создать несколько скриптов, как начинаешь подмечать определенные паттерны в логике, и чем дальше, тем становится проще, однако во всех случаях на это уходит время. К счастью, вместе с ростом объемов данных стали появляться и некоторые инструменты для выполнения этой типовой процедуры.

GOOGLE REFINE

Google Refine — это результат эволюции Freebase Gridworks. Gridworks разрабатывался поначалу для открытой платформы данных Freebase в качестве собственного инструментального средства. Однако позже Freebase была приобретена компанией Google, отсюда и новое наименование. Google Refine представляет собой по сути Gridworks 2.0, но с более удобным в употреблении интерфейсом (см. рис. 2.8) и с бóльшим набором возможностей.

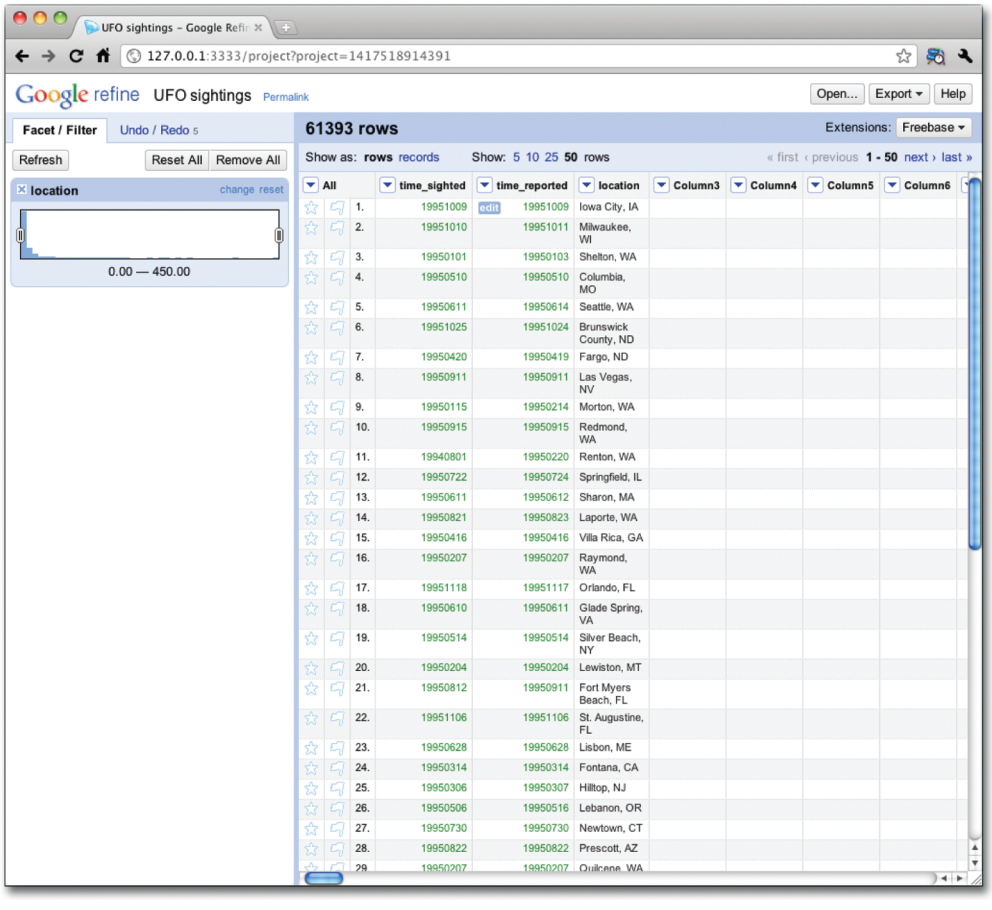


Рис. 2.8. Пользовательский интерфейс Google Refine

Программа запускается на вашем компьютере (только через браузер), что само по себе очень удобно, так как вам не придется беспокоиться из-за необходимости загружать закрытую информацию на сервера Google. Вся обработка данных происходит у вас на компьютере. У Refine по-прежнему открытый исходный код, так что если вами овладеют амбиции, вы сможете приспособить инструмент к вашим собственным «расширенным» потребностям.

Когда вы откроете Refine, вы увидите знакомый табличный интерфейс с вашими строчками и столбцами. В нем вы легко можете сортировать данные по полям и искать определенные величины. А еще вы можете относительно быстро находить ошибки и проводить операции объединения данных.

Допустим, по какой-то причине у вас на руках оказалась инвентарная опись вашей кухни. Вы можете загрузить данные в Refine и быстро найти нестыковки, такие как опечатки или разнящиеся классификации. Может, в слове «вилки» допущена ошибка и в перечне значатся «илки». Или вы хотите изменить классификацию и перевести все вилки, ложки и ножи в категорию «утварь». С Refine вы легко найдете все эти пункты и внесете необходимые правки. Если в результате изменения вам не понравятся или вы допустите ошибку, вы сможете вернуться к предыдущему варианту простой командой «Отменить» (Undo).

Когда вы перейдете к более сложным процедурам, то сможете также объединять различные источники данных, скажем, ваши собственные данные с массивами из Freebase, и создавать расширенные данные.

В любом случае Google Refine — это хороший инструмент, который всегда стоит иметь под рукой. Он мощный, и скачать его можно бесплатно, так что я горячо рекомендую вам хотя бы опробовать его.

▶ Скачайте Google Refine с `http://code.google.com/p/google-refine` и просмотрите обучающее руководство, чтобы узнать, как извлечь из этого инструмента максимум пользы.

MR. DATA CONVERTER

Часто бывает и так, что у вас есть возможность получить все данные в Excel, но затем необходимо конвертировать их в другой формат, который в большей степени соответствует вашим потребностям. Подобное происходит почти всегда в случае создания графики для Сети. Вы уже умеете экспортировать таблицы Excel в формате CSV, но как быть, если вам нужно сделать что-то еще? Вам поможет Mr. Data Converter.

Mr. Data Converter — это простой и бесплатный инструмент. Его создал Шэн Картер (Shan Carter), редактор графики в New York Times. Бóльшую часть своего рабочего времени Картер проводит в создании интерактивной графики для онлайн-версии газеты. Ему часто приходится конвертировать данные, чтобы они подходили для использования в программах, с которыми он работает, а потому неудивительно, что он создал инструмент, упрощающий этот процесс.

Пользоваться Mr. Data Converter легко. Как вы можете убедиться, взглянув на рис. 2.9, интерфейс совершенно простой. Все, что

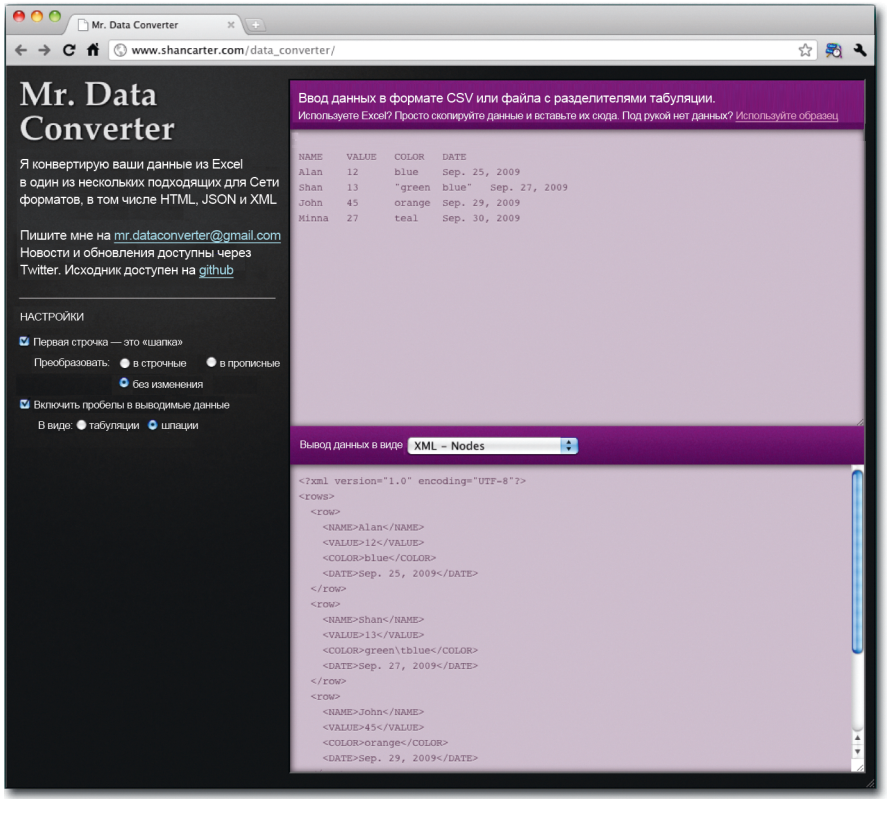


Рис. 2.9. Mr. Data Converter существенно упрощает перевод данных из одного формата в другой

▶ Опробуйте Mr. Data Converter на сайте `http://www.shancarter.com/data_converter` или скачайте исходник с GitHub по адресу at `https://github.com/shancarter/Mr-Data-Converter`, чтобы конвертировать свои таблицы Excel в формат, удобный для размещения в Сети.

вам необходимо сделать, — это скопировать данные из Excel и вставить их в область ввода в верхней части экрана, а внизу указать, в каком формате вы хотите получить их. На выбор у вас XML, JSON и ряд других форматов.

Исходный код Mr. Data Converter также доступен — на тот случай, если вы захотите сделать свой собственный вариант или дополнить существующий.

MR. PEOPLE

Вдохновленный примером Картера с его Mr. Data Converter, заместитель руководителя отдела графики New York Times Мэтью Эриксон (Matthew Ericson) создал свой инструмент — Mr. People. Как и Mr. Data Converter, Mr. People дает возможность копировать и вставлять данные в текстовое поле, а затем анализирует и извлекает их за вас. Однако, как вы могли догадаться, Mr. People предназначен специально для выполнения парсинга имен.

Может, у вас есть длинный перечень имен, не отформатированных ни по какому принципу, и вы хотите установить для каждого включенного в список человека его имя, фамилию, а заодно и инициалы второго имени, титул, если таковой есть, и пр. А может, множество людей перечислены друг за другом в один ряд. Вот здесь-то на сцену и выходит Mr. People. Скопируйте имена и вставьте их так, как показано на рис. 2.10. Вы получите красивую аккуратную таблицу, которую вы сможете использовать в вашей любимой программе для работы с таблицами (рис. 2.11).

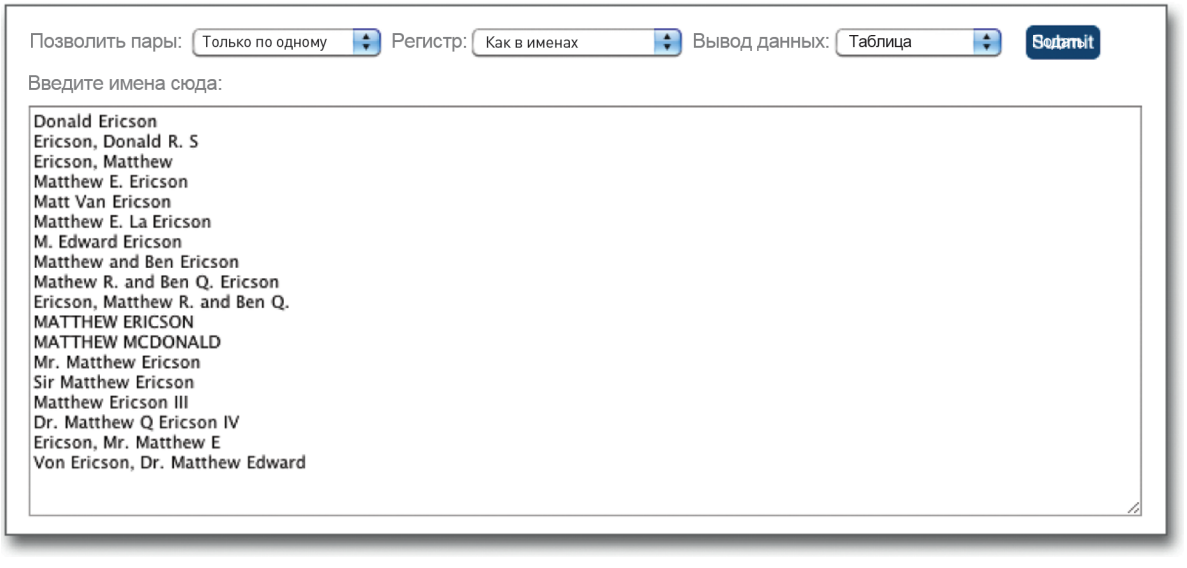
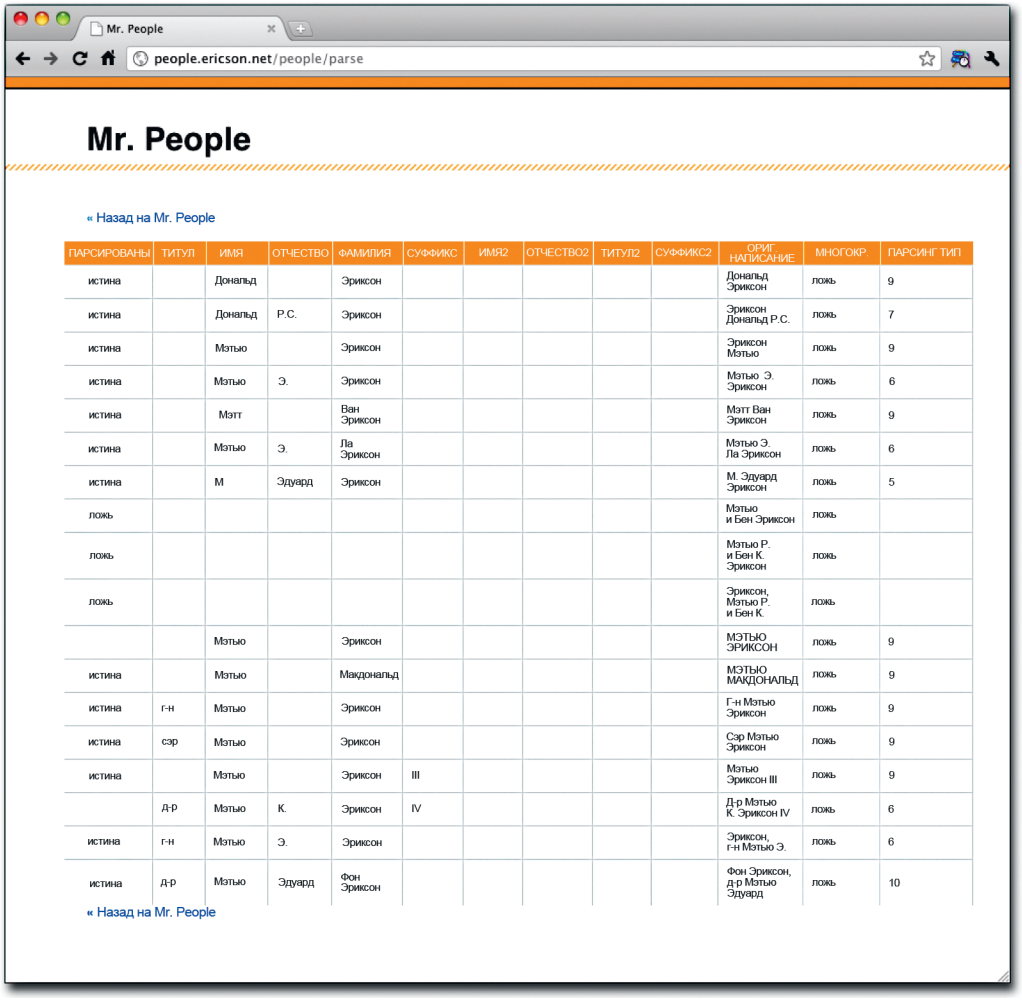


Рис. 2.10. Страница ввода имен в Mr. People



ПАРСИРОВАНЫ	ТИТУЛ	ИМЯ	ОТЧЕСТВО	ФАМИЛИЯ	СУФФИКС	ИМЯ2	ОТЧЕСТВО2	ТИТУЛ2	СУФФИКС2	ОРИГ. НАПИСАНИЕ	МНОГОКР.	ПАРСИНГ ТИП
истина		Дональд		Эриксон						Дональд Эриксон	ложь	9
истина		Дональд	Р.С.	Эриксон						Эриксон Дональд Р.С.	ложь	7
истина		Мэтью		Эриксон						Эриксон Мэтью	ложь	9
истина		Мэтью	Э.	Эриксон						Мэтью Э. Эриксон	ложь	6
истина		Мэтт		Ван Эриксон						Мэтт Ван Эриксон	ложь	9
истина		Мэтью	Э.	Ла Эриксон						Мэтью Э. Ла Эриксон	ложь	6
истина		М	Эдуард	Эриксон						М. Эдуард Эриксон	ложь	5
ложь										Мэтью и Бен Эриксон	ложь	
ложь										Мэтью Р. и Бен К. Эриксон	ложь	
ложь										Эриксон, Мэтью Р. и Бен К.	ложь	
		Мэтью		Эриксон						МЭТЬЮ ЭРИКСОН	ложь	9
истина		Мэтью		Макдональд						МЭТЬЮ МАКДОНАЛЬД	ложь	9
истина	г-н	Мэтью		Эриксон						Г-н Мэтью Эриксон	ложь	9
истина	сэр	Мэтью		Эриксон						Сэр Мэтью Эриксон	ложь	9
истина		Мэтью		Эриксон	III					Мэтью Эриксон III	ложь	9
	д-р	Мэтью	К.	Эриксон	IV					Д-р Мэтью К. Эриксон IV	ложь	6
истина	г-н	Мэтью	Э.	Эриксон						Эриксон, г-н Мэтью Э.	ложь	6
истина	д-р	Мэтью	Эдуард	Фон Эриксон						Фон Эриксон, д-р Мэтью Эдуард	ложь	10

Рис. 2.11. Имена в формате таблицы после обработки в Mr. People

▶ Используйте Mr. People на `http://people.ericson.net` или скачайте исходный код на языке Ruby с GitHub, чтобы использовать парсер имен в своих собственных скриптах. Для этого зайдите на `http://github.com/mericson/people`.

Подобно Mr. Data Converter, Mr. People также является программой с открытым исходным кодом, и ее можно скачать с GitHub.

ПРОГРАММЫ ТАБЛИЧНЫХ ВЫЧИСЛЕНИЙ

Конечно, если вам нужно просто рассортировать некие данные или внести небольшие изменения в отдельные пункты, вы всегда сможете воспользоваться вашей любимой табличной программой. Если вам предстоят небольшие правки, идите проторенной дорогой. Если же нет, тогда попробуйте сначала то, о чем говорилось выше (особенно если у вас огромный массив данных), или подумайте о создании кода, специально предназначенного для конкретной задачи.

Форматирование с помощью кода

Хотя программы, работающие по методу «указал и щелкнул», могут быть полезны, иногда они выдают не совсем то, что вам нужно, особенно если вы работаете с данными достаточно длительное время. Некоторые программы не очень хорошо справляются с объемными файлами данных, их работа замедляется или они и вовсе выходят из строя.

Что делать в подобной ситуации? Вы можете поднять белый флаг и сдаться, хотя это будет непродуктивное решение. Но вы также можете написать небольшую программу и выполнить работу с ее помощью. Располагая кодом, вы станете намного гибче и сумеете подогнать свой скрипт под имеющиеся данные.

Так что давайте, с ходу смотрите пример того, как можно легко переключаться с одного формата данных на другой с помощью всего нескольких строчек кода.

ПРИМЕР: ПЕРЕКЛЮЧЕНИЕ МЕЖДУ РАЗЛИЧНЫМИ ФОРМАТАМИ ДАННЫХ

В этом примере применяется Python, но вы, конечно, можете использовать тот язык, которому отдаете предпочтение. Логика остается такой же, только синтаксис будет несколько иным. (Я люблю разрабатывать приложения в Python, а потому производственный процесс у меня хорошо согласуется с обработкой исходных данных с помощью Python.)

Вернитесь к предыдущему примеру с анализом данных и используйте полученный файл `wunder-data.txt`, в котором содержатся даты и показания о температуре в Буффало за 2009 год. Первые строчки выглядят так:

```
20090101,26
20090102,34
20090103,27
20090104,34
20090105,34
20090106,31
20090107,35
20090108,30
20090109,25
...
```

Это CSV-файл, но, допустим, вы хотите, чтобы данные были в XML, вот в таком формате:

```
<weather_data>
   <observation>
      <date>20090101</date>
```

```
      <max_temperature>26</max_temperature>
   </observation>
   <observation>
      <date>20090102</date>
      <max_temperature>34</max_temperature>
   </observation>
   <observation>
      <date>20090103</date>
      <max_temperature>27</max_temperature>
   </observation>
   <observation>
      <date>20090104</date>
      <max_temperature>34</max_temperature>
   </observation>
   …
</weather_data>
```

Температура за каждый день содержится в тегах <observation>, включающих теги <date> и <max_temperature>.

Чтобы конвертировать CSV в формат XML, как это сделано выше, вы можете использовать следующий фрагмент кода:

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
print '<weather_data>'

for row in reader:
   print '<observation>'
   print '<date>' + row[0] + '</date>'
   print '<max_temperature>' + row[1] + '</max_temperature>'
   print '</observation>'

print '</weather_data>'
```

Необходимые модули вы импортируете, как и прежде. В данном случае, чтобы прочитать `wunder-data.txt`, вам нужен только CSV-модуль.

```
import csv
```

Вторая строчка кода открывает для чтения `wunder-data.txt`, используя `open()`, а затем загружает его при помощи метода `csv.reader()`.

```
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
```

Обратите внимание на то, что в качестве разделителя здесь стоит запятая. Если в файле для этой цели используется табуляция, тогда вам следует указать разделитель в виде `'\t'`.

Затем в строке 3 вы вводите в XML-файл первый тег.

```
print '<weather_data>'
```

В основном фрагменте цикл повторяется для каждой строчки данных, и они отображаются в том формате, в каком вам нужно. В настоящем примере каждая строчка в заголовке CSV эквивалентна каждому наблюдению (observation) в XML.

```
for row in reader:
   print '<observation>'
   print '<date>' + row[0] + '</date>'
   print '<max_temperature>' + row[1] + '</max_temperature>'
   print '</observation>'
```

В каждой строчке есть два значения: даты и максимальной температуры.
Завершите перевод в XML закрывающим тегом:

```
print '</weather_data>'
```

Главную роль здесь играют два основных момента. Вы вводите данные и затем производите итерацию по ним, меняя каждую строчку определенным образом. Если бы вам предстояло конвертировать полученный XML обратно в CSV, логика действий была бы точно такой же. Как видно из следующего фрагмента, отличие состоит лишь в том, что вы используете для парсинга XML-файла другой модуль.

```
from BeautifulSoup import BeautifulStoneSoup

f = open('wunder-data.xml', 'r')
xml = f.read()

soup = BeautifulStoneSoup(xml)
observations = soup.findAll('observation')
for o in observations:
   print o.date.string + "," + o.max_temperature.string
```

Код выглядит по-другому, но, по сути, вы делаете то же самое. Вместо CSV-модуля вы импортируете BeautifulStoneSoup с BeautifulSoup. Вспомните, что вы уже использовали BeautifulSoup для анализа HTML-сайта Weather Underground. BeautifulStoneSoup используется для парсинга более распространенного XML.

Вы можете открыть XML-файл для чтения, используя `open()`, а затем загрузить контент в переменную `xml`. На данном этапе контент сохраняется в виде строки. Для начала парсинга передайте `xml`-строку в `BeautifulStoneSoup` для итерации через каждый `<observation>` в XML-файле. Используйте `findAll()` для того, чтобы извлечь все наблюдения и, наконец, как и при

переводе CSV в XML, прогнать цикл по каждому наблюдению и отобразить значения в нужном вам формате.

Это приведет вас туда, откуда вы начинали:

```
20090101,26
20090102,34
20090103,27
20090104,34
…
```

Для полноты картины вот вам еще и код, с помощью которого можно конвертировать CSV в формат JSON.

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=»,»)

print "{ observations: ["
rows_so_far = 0
for row in reader:

   rows_so_far += 1

   print '{'
   print '"date": ' + '"' + row[0] + '", '
   print '"temperature": ' + row[1]

   if rows_so_far < 365:
      print " },"
   else:
      print " }"

print "] }"
```

Пробегитесь глазами по строчкам, чтобы понять, что здесь происходит. Мы снова имеем дело с той же логикой, только с иным результатом. Если вы запустите приведенный выше код, то получите свои данные в формате JSON. Вот как они будут выглядеть:

```
{
   "observations": [
      {
         "date": "20090101",
         "temperature": 26
      },
      {
         "date": "20090102",
```

```
            "temperature": 34
        },
        …
    ]
}
```

Это все те же данные с датой и температурой, но в еще одном формате. Компьютеры просто обожают разнообразие.

Внесите в цикл логику

Если вы посмотрите на код для конвертирования CSV-файла в JSON, вы наверняка заметите оператор `if-else` в цикле `for` после трех строчек `print`. Его назначение — проверить, не является ли текущая итерация по рядам с данными последней. Если нет, тогда в конце записи точка не нужна. В противном случае поставьте ее. Это часть спецификации JSON. И здесь у вас больше возможностей.

Вы можете проверить, например, не поднималась ли максимальная температура выше определенного уровня, и создать новое поле со значением 1, если температура в конкретный день оказывалась выше некоего заданного порога, или 0, если нет. Вы можете также создать категории или пометить дни, за которые нет данных.

На самом деле проверка данных на предмет надпороговых значений — это не единственный доступный вариант. Вы можете подсчитать скользящее среднее или разницу между текущим и предыдущим днями. В рамках цикла вы можете много чего сделать, чтобы обогатить сырые данные. Здесь упомянуто далеко не все, поскольку вам позволено делать что угодно — от поверхностных правок до глубокого анализа. Посмотрите на следующий простой пример.

Вернитесь к вашему первоначальному файлу `wunder-data.txt` и создайте третью колонку, показывающую, превышает ли дневная температура точку замерзания. Так, 0 будет значить, что температура превысила заданный порог, а 1 — что она оказалась на уровне точки замерзания или ниже.

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
for row in reader:
    if int(row[1]) <= 32:
        is_freezing = '1'
    else:
        is_freezing = '0'

    print row[0] + "," + row[1] + "," + is_freezing
```

Как и прежде, данные считываются из CSV-файла в Python, после чего производится итерация по каждой строчке. Все дни проверяются и помечаются соответствующим образом.

Пример, конечно, простой, но он наглядно демонстрирует, как вы можете развить эти принципы для форматирования или обогащения данных как вашей душе угодно. Запомните эти три шага: загрузка, цикл, обработка — и действуйте свободно.

Закругляясь

В настоящей главе обсуждалось, как находить необходимые данные и что с ними делать после нахождения. В процессе визуализации это очень важный этап, чтобы не сказать — самый важный. Информационная графика интересна настолько, насколько интересны лежащие в ее основе данные. Вы можете украсить свою диаграмму или график как хотите, но именно данные (или результаты вашего анализа этих данных) будут оставаться основой основ. И теперь, когда вы знаете, откуда и как доставать данные, вы готовы двигаться дальше.

Вы также впервые попробовали свои силы в программировании. Вы извлекли данные с сайта, а затем отформатировали и перегруппировали их. Этот прием еще сослужит вам добрую службу в следующих главах. Однако самое важное в коде — логика. Вы использовали Python, но с таким же успехом вы могли бы обратиться и к Ruby, и к Perl, и к PHP. Каким бы языком вы ни пользовались, принцип действия остается неизменным. Когда вы освоите один из языков программирования — а если вы уже состоявшийся программист, вы подтвердите мои слова, — вам будет гораздо легче затем освоить еще один.

Не всегда нужно прибегать к кодам. В некоторых случаях приложения, действующие по принципу перетаскивания (drag and drop), способны существенно облегчить вашу работу, и в таких случаях вам стоит этим пользоваться. В конечном счете, чем больше инструментов будет в вашем инструментарии, тем меньше опасность, что вы увязнете в процессе работы.

Итак, данные у вас уже есть. Теперь пора приступить к визуализации.