Райан Митчелл. Скрапинг вебсайтов с помощью Python

Я не программист. Правда, иногда пишу небольшой код на VBA, когда стандартными средствами Excel не могу решить стоящую передо мной задачу. Некоторое время тому назад я начал читать книгу Нейтана Яу. Искусство визуализации в бизнесе. Я думал еще более усовершенствовать свои знания в области визуализации с помощью Excel, но оказалось, что автор продвигает методы, основанные на программировании. Почти сразу же я столкнулся с небольшой программой, написанной на языке Python, извлекающей данные из Интернета. Я установил на своем ПК свободно распространяющуюся версию программы, но код не заработал. Коллега подсказал, что код был написан в 2009 г., так что современная версия Python 3.5.1 его не поддерживает… Я решил приобрести начальные знания по программе и прочитал книгу Майка МакГрата Программирование на Python для начинающих. Полученных знаний для запуска непослушного кода не хватило. Обратился к сообществу, но начинающий программист никого не заинтересовал своими проблемами… Тем временем сдаваться не хотелось. И, удача – нахожу книгу издательства ДМК Пресс точно по моей теме. Любопытно, что книга вышла 30 апреля 2016 г.

Райан Митчелл. Скрапинг веб-сайтов с помощью Python. – М.: ДМК Пресс, 2016. – 280 с.

Купить книгу в издательстве ДМК Пресс или Лабиринте

Примеры, используемые в книге, можно скачать с сайта издательства. Скачать программу можно с python.org. Я установил версию Python 3.5.1 с помощью Windows x86-64 web-based installer. Вместе с программой устанавливается среда разработки IDLE. Эта книга не подойдет в качестве первого знакомства с Python.

Веб-скрапинг – это сбор данных с помощью любых средств, кроме программ, использующих API (или человека, использующего веб-браузер). Чаще всего веб-скрапинг осуществляется с помощью программы, которая автоматически запрашивает веб-сервер, получает данные (HTML и другие файлы, которые размещены на веб-страницах), а затем выполняет парсинг этих данных, чтобы извлечь необходимую информацию.

ЧАСТЬ I. ПОСТРОЕНИЕ СКРАПЕРОВ

Глава 1. Ваш первый скрапер

```
from urllib.request import urlopen
html = urlopen("http://pythonscraping.com/pages/page1.html")
print(html.read())
```

Эта программа выведет полный HTML-код указанной страницы (рис. 1). В Python 3.x библиотека urlib2 была переименована в urllib и разбита на несколько подмодулей (здесь мне открылась первая проблема неработающего кода: в первой строке скрипт пытался загрузить библиотеку urllib2, которой в Python 3.5 просто не существует. – *Прим. Багузина*).

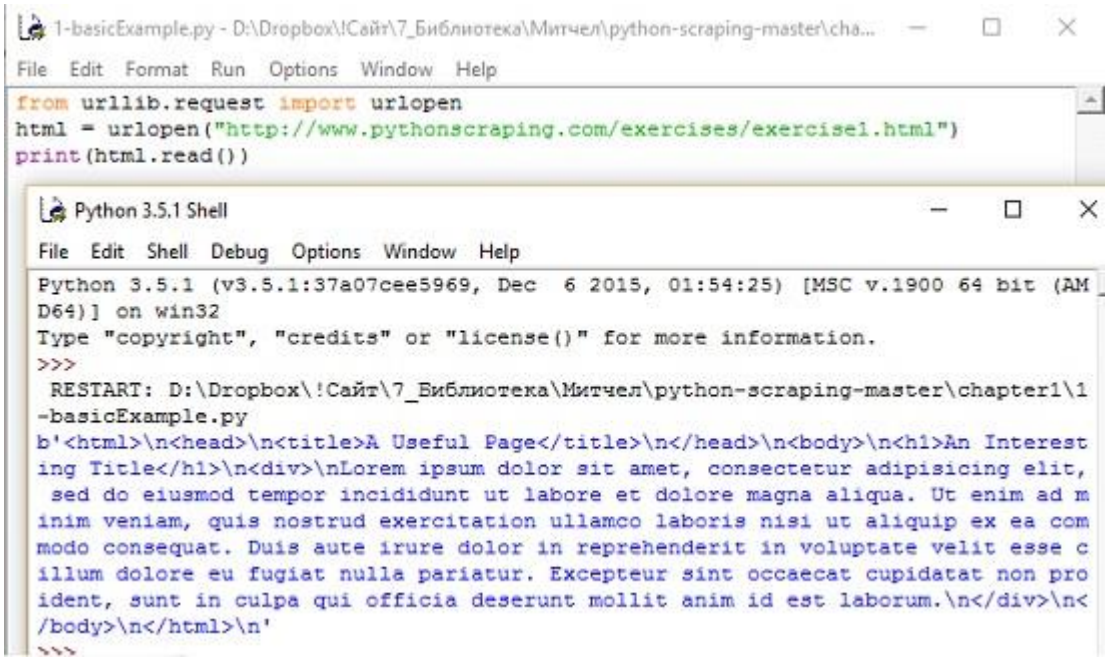

Рис. 1. Ваш первый скрапер

urllib – стандартная библиотека Python (то есть вам не нужно устанавливать ничего дополнительно, чтобы запустить этот пример) и содержит функции для запроса данных в сети, обработки cookies и даже изменения метаданных (заголовков и пользовательского агента). На протяжении всей книги мы будем использовать urllib довольно часто, поэтому рекомендуем вам прочитать документацию Python по этой библиотеке.

Библиотека *BeautifulSoup* была названа в честь одноименного стихотворения, входящего в сказку Льюиса Кэролла «Приключения Алисы в Стране чудес». Библиотека BeautifulSoup пытается придать смысл бессмыслице, она помогает отформатировать и систематизировать грязные интернет-данные, исправляя плохо размеченные HTML страницы и выводя их в виде легко обрабатываемых объектов Python, представляющих собой XML-структуры. Поскольку библиотека BeautifulSoup не является библиотекой Python по умолчанию, ее нужно установить. На протяжении всей книги мы будем использовать библиотеку BeautifulSoup 4 (также известную как BS4). Чтобы скачать Beautiful Soup, зайдите на crummy.com (я скачал самую свежую версию 4.4.1). Распакуйте архив, зайдите в распакованную папку и в командной строке выполните:

> python setup.py install

BeautifulSoup теперь используется в качестве библиотеки Python на вашем компьютере (это дало мне подсказку по второй проблеме неработающего кода: во второй строке скрипт пытался загрузить библиотеку BeautifulSoup, которая написана для Python 2. Чтобы сконвертировать модуль BeautifulSoup для Python 3 нужно выполнить команду *setup.py install – Прим. Багузина*). Проверьте что библиотека BeautifulSoup стала доступна, открыв терминал Python и импортировав библиотеку:

> from bs4 import BeautifulSoup

Подробнее о библиотеке BeautifulSoup см. документацию на русском языке. Наиболее часто в библиотеке BeautifulSoup используется объект BeautifulSoup. Дополним пример, приведенный выше:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen("http://www.pythonscraping.com/pages/page1.html")
bsObj = BeautifulSoup(html.read());
print(bsObj.h1)
```

Результат работы кода:

<h1>An Interesting Title</h1>

Поскольку web не идеален, предусмотрите в коде скрапинга обработку ошибок (рис. 2).

```
3-exceptionHandling.py - D:\Dropbox\!Сайт\7_Библиотека\Митчел\python-scraping-maste...
File  Edit  Format  Run  Options  Window  Help
from urllib.request import urlopen
from urllib.error import HTTPError
from bs4 import BeautifulSoup
import sys


def getTitle(url):
    try:
        html = urlopen(url)
    except HTTPError as e:
        print(e)
        return None
    try:
        bsObj = BeautifulSoup(html.read())
        title = bsObj.body.h1
    except AttributeError as e:
        return None
    return title

title = getTitle("http://www.pythonscraping.com/exercises/exercise1.html")
if title == None:
    print("Title could not be found")
else:
    print(title)
```

Рис. 2. Обработка ошибок

Глава 2. Продвинутый парсинг HTML

Стилевое оформление (CSS) предназначено не только для браузера, оно упрощает работу веб-скраперов. CSS основан на дифференциации HTML-элементов, которые могут иметь точно такую же разметку, но отличаться по стилю. То есть некоторые теги могут выглядеть следующим образом:

тогда как другие теги выглядят так:

Веб-скраперы могут дифференцировать эти теги в зависимости от их класса (дочитав до этого места, я вернулся к своему коду, и мне, наконец, удалось его запустить; см. Извлечение данных с web-страниц с помощью кода на языке Python).

Следующий код позволит извлечь с тестовой страницы все имена персонажей, выделенные зеленым цветом (рис. 3).

```
1-selectByClass.py - D:\Dropbox\!Сайт\7_Библиотека\Митчел\python-scraping-master\cha...
File  Edit  Format  Run  Options  Window  Help
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.pythonscraping.com/pages/warandpeace.html")
bsObj = BeautifulSoup(html, "html.parser")
nameList = bsObj.findAll("span", {"class":"green"})
for name in nameList:
    print(name.get_text())

Python 3.5.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Anna
Pavlovna Scherer
Empress Marya
Fedorovna
Prince Vasili Kuragin
Anna Pavlovna
St. Petersburg
```

Рис. 3. Извлечение всех имен, выделенных зеленым цветом

С помощью объекта bsObj мы можем вызвать функцию findAll, чтобы извлечь список имен собственных. Их находят путем отбора текста, находящегося внутри тегов . В общем виде запрос выглядит следующим образом: bsObj.findAll(tagName, tagAttributes). Он позволяет получить список всех тегов с указанным именем и атрибутами. Функция findAll возвращает теги с их содержимым. Наконец функция name.get_text() отделяет контент от тегов, и печатает только имена.

На самом деле функция findAll имеет больше аргументов:

findAll(tag, attributes, recursive, text, limit, keywords)

А аргументами tag и attributes мы только что познакомились. Аргумент recursive является булевым значением. По умолчанию он равен True и функция findAll исследует элементы на всю глубину вложенности тегов. Если установлено значение False, то будут исследованы только теги верхнего (первого) уровня. Аргумент text ищет совпадения, руководствуясь текстовым содержимым тегов, а не свойствами самих тегов. Например, код:

nameList = bsObj.findAll(text="the prince")

print(len(nameList))

вернет число вхождений фразы "the prince" в текст (7).

Аргумент limit позволяет ограничить извлечение первыми x элементами со страницы. Аргумент keyword позволяет вам выбрать теги, содержащие конкретный атрибут. Например,

allText = bsObj.findAll(id="text")

print(allText[0].get_text())

Помимо двух основных объектов библиотеки BeautifulSoup (объекты BeautifulSoup, например bsObj и объекты Tag, возвращаемые функцией findAll) существуют объекты NavigableString: используются для представления текста внутри тегов, а не самих тегов, и объект Comment: используются для поиска HTML-комментариев в тегах комментариев <!-как, например, этот-->.

Навигация по дереву синтаксического разбора. Функция findAll осуществляет поиск тегов по их имени и атрибуту. Но что, если вам нужно найти тег, основываясь на его месторасположении в документе? Навигация вглубь осуществляется просто:

bsObj.tag.subTag.anotherSubTag

В библиотеке BeautifulSoup существует различие между дочерними элементами и элементами-потомками: дочерние теги всегда на один уровень ниже родительского тега, в то время как теги-потомки могут быть ниже родительского тега, располагаясь на любом уровне. Например, bsObj.body.h1 выбирает первый тег h1, который является потомком тега body. Она не найдет теги, расположенные вне body. Аналогично bsObj.div.findAll("img") найдет первый тег div в документе, а затем извлечет все теги img, которые являются потомками тега div.

Если вы хотите извлечь только дочерние теги, используйте тег .children:

```
from urilib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.pythonscraping.com/pages/page3.html")
bsObj = BeautifulSoup(html, "html.parser")

for child in bsObj.find("table",{"id":"giftList"}).children:
        print(child)
```

Этот код выводит список строк с названиями продуктов таблицы giftList. Если бы вы написали его, используя функцию descendants() вместо функции children(), в таблице было бы найдено и напечатано около двух десятков тегов, в том числе теги img, теги span и отдельные теги td.

Функции next_siblings(), previous_ siblings упрощают сбор данных из таблиц, а для работы с родительскими тегами служит функция parents().

Регулярные выражения – это быстрый способ представления наборов правил. Например,

1. Запишите букву «a», по крайней мере, один раз.
2. Добавьте к этой букве букву «b» ровно пять раз.
3. Добавьте к этой букве букву «c» любое четное количество раз.
4. Опционально запишите букву «d» в конце.

Строки, которые соответствуют этим правилам: «aaaabbbbbccccd», «aabbbbbcc» и т.д. Регулярное выражение для этого набора правил:

aa*bbbbb(cc)*(dl),

где

a – записывает букву a,

a* – записывает букву a любое количество раз, в том числе и ноль,

bbbbb – просто пять букв b,

(cc)* – любое количество элементов можно сгруппировать, заключив их в скобки; если мы считаем ноль четным, то подойдет это выражение; если ноль не относим к четным, то нужно записать c(cc)*,

(dl) – вертикальная черта в середине двух выражений означает возможность выбора («этот символ или тот»; в этом случае мы задаем правило «добавить букву d с последующим пробелом или просто добавить пробел без буквы d». Таким образом, мы можем гарантировать, что у нас будет максимум одна буква d с последующим пробелом, завершающим строку.

Один из классических примеров применения регулярных выражений – поиск адресов электронной почты: [A-Za-z0-9\._+]+@[A-Za-z]+\.(com|org|edu|net|ru).

Регулярное выражение можно вставить в качестве аргумента в выражение BeautifulSoup (рис. 4). Данный код печатает только относительные пути к файлам изображений, которые начинаются с ../mg/gifts/img и заканчиваются на .jpg.

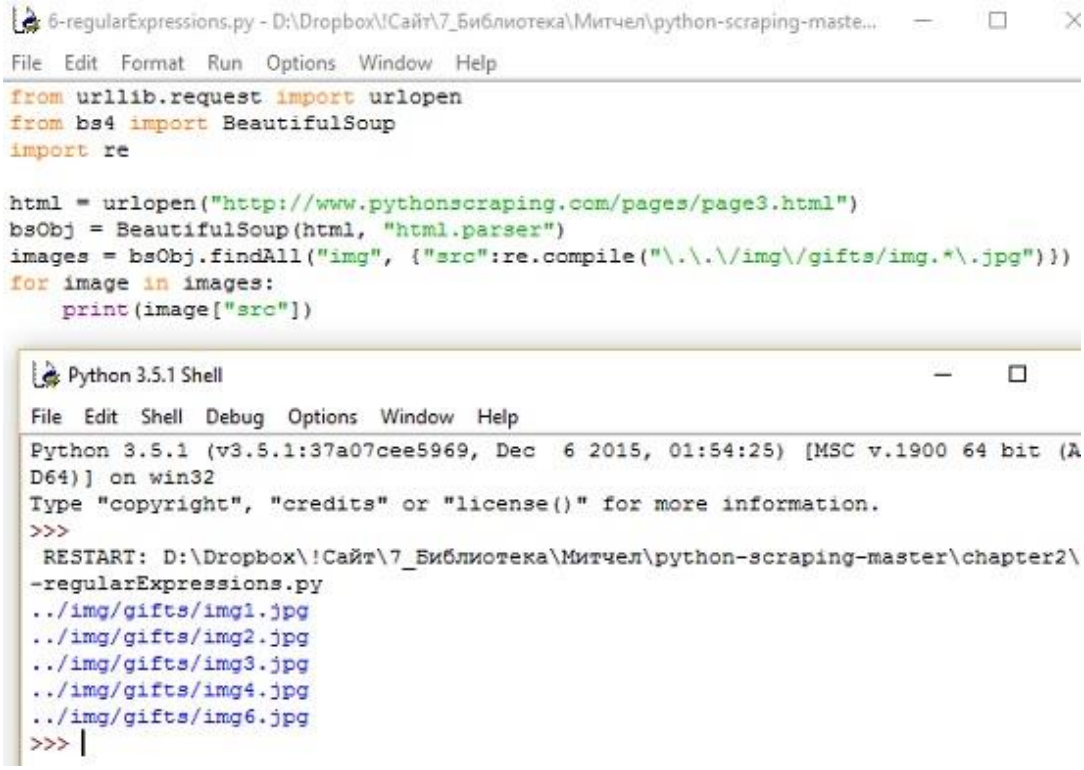

Рис. 4. Регулярное выражение, как аргумент функции BeautifulSoup

Часто в ходе веб-скрапинга надо найти не контент, заключенный внутри тега, а атрибуты тега. Особенно это важно для тега <a>, который с помощью атрибута href устанавливает ссылку на сайт, его отдельную страницу или файл (в качестве ссылки используется URLадрес), или тега <img>, который с помощью атрибута src задает адрес файла изображения. Получить доступ к списку атрибутов можно, вызвав: myTag.attrs. Адрес файла изображения можно найти с помощью следующей строки: myImgTag.attrs['src'].

Лямбда-выражения – это функция, которая передается в другую функцию в качестве переменной. То есть вместо определения функции как f(x, y) вы можете задать функцию как f(g(x), y).

BeautifulSoup позволяет передать определенные типы функций в качестве параметров в функцию findAll. Единственное ограничение в том, что эти функции должны принять теговый объект в качестве аргумента и возвратить булево значение. Каждый теговый объект, обрабатываемый BeautifulSoup, оценивается в этой функции, и теги, которые оцениваются как «истинные», возвращаются, в то время как остальные отбрасываются.

Например, следующая строга извлекает все теги, у которых ровно два атрибута:

soup.findAll(lambda tag: len(tag.attrs) == 2)

То есть она найдет следующие теги:

Глава 3. Запуск краулера

Веб-краулеры, или веб-пауки, называются так потому, что они ползают по вебу. В основе их работы рекурсивный обход. Они должны извлечь содержимое страницы по указанному URL-адресу, исследовать эту страницу на наличие другого URL-адреса, извлечь страницу по найденному URL-адресу и т.д. Следующий код позволит случайным образом обходить страницы английского сегмента wiki, начиная со страницы Moscow (рис. 5).

Рис. 5. Краулер обходит 10 случайных страниц Wiki, начиная с https://en.wikipedia.org/wiki/Moscow

Первое, что делает программа после импорта необходимых библиотек, устанавливает начальное значение генератора случайных чисел, используя текущее системное время. Это практически гарантирует уникальный и случайный маршрут перемещения по статьям Википедии при каждом запуске программы. Затем она задает функцию getLinks, которая берет URL-адрес статьи в формате /wiki/..., добавляет доменное имя Википедии http://en.wikipedia.org и извлекает объект BeautifulSoup из HTML-страницы в этом домене. Потом она извлекает список тегов, содержащих ссылки только на статьи Wiki, используя регулярные выражения, и возвращает их. Далее случайным образом выбирает одну из ссылок, переходит по ней и продолжает так 10 раз.

Глава 4. Использование API

Интерфейсы прикладного программирования (application programming interfaces, или API) – средства для обмена информацией между несколькими различными приложениями. Не имеет значения, что приложения написаны разными программистами, с использованием различных архитектур или даже на разных языках, API выступают в качестве общепринятого языка для различных программ, которые должны обмениваться информацией друг с другом. Как правило, программист выполняет запрос к API с помощью HTTP, чтобы получить некоторые типы данных, а API возвращает эти данные в виде XML или JSON.

Например, следующий API-запрос http://freegeoip.net/json/50.78.253.58 вернет (этот API определяет географическое местоположение по IP-адресу):

```
{"ip":"50.78.253.58","country_code":"US","country_name":"United
States","region_code":"CT","region_name":"Connecticut","city":"Milford","zip_cod
e":"06460","time_zone":"America/New_York","latitude":41.2241,"longitude":-
73.0517,"metro_code":533}
```

Есть четыре способа отправить запрос к веб-серверу с помощью HTTP: GET, POST, PUT, DELETE.

GET используется, когда вы посещаете вебсайт с помощью адресной строки в браузере (как в примере выше). POST используется, когда вы заполняете регистрационную форму или отправляете информацию внутреннему скрипту сервера. PUT-запрос нужен для изменения объекта (ранее созданного POST-запросом). DELETE используется для удаления объекта (ранее созданного POST-запросом).

Современные API перед использованием часто требуют аутентификации. Методы аутентификации построены на использовании определенного токена, который передается на веб-сервер с каждым вызовом API. Токен можно передать в URL-адресе запроса напрямую, или с помощью cookie в заголовке запроса.

API возвращают хорошо структурированные ответы в формате

- Расширяемый язык разметки или Extensible Markup Language (XML),
- или Представление объектов JavaScript или JavaScript Object Notation (JSON).

При извлечении данных с помощью GET-запроса URL-путь описывает интересующие нас данные, а параметры запроса выступают в качестве фильтров. Например, вы можете отправить запрос, чтобы извлечь все посты пользователя с ID 1234, написанные в течение августа 2014 года:

http://socialmediasite.com/users/1234/posts?from=08012014&to=08312014

Twitter защищает свои API. Те не менее, вы можете создать учетную запись в Twitter, а затем зарегистрировать ваше «приложение» на сайте Twitter Developers. Для работы с Twitter существует библиотеки для Python 3.x. Код, приведенный ниже, подключается к Twitter API и выводит список твитов в формате JSON, содержащий хэштег #python. Не забудьте изменить строки в OAuth в соответствии с вашими реальными учетными данными:

```
from twitter import Twitter
t = Twitter(auth=OAuth(<Access Token>,<Access Token Secret>,<Consumer Key>,<Consumer Secret>))
pythonTweets = t.search.tweets(q = "#python")
print(pythonTweets)
```

Вывод этого скрипта может быть огромен, но вы получаете развернутую информацию о твите: дата и время публикации твита, подробная информация о ретвитах или избранных твитах, сведения об аккаунте пользователя, изображении профиля и др.

На сегодняшний день **Google** предлагает наиболее универсальные и простые в использовании API в Интернете. Для многих своих популярных приложений (Gmail, YouTube, Blogger) Google разработал API. Если у вас уже есть аккаунт Google, можно просмотреть список доступных API и получить API-ключ, используя Google Developers Console (рис.

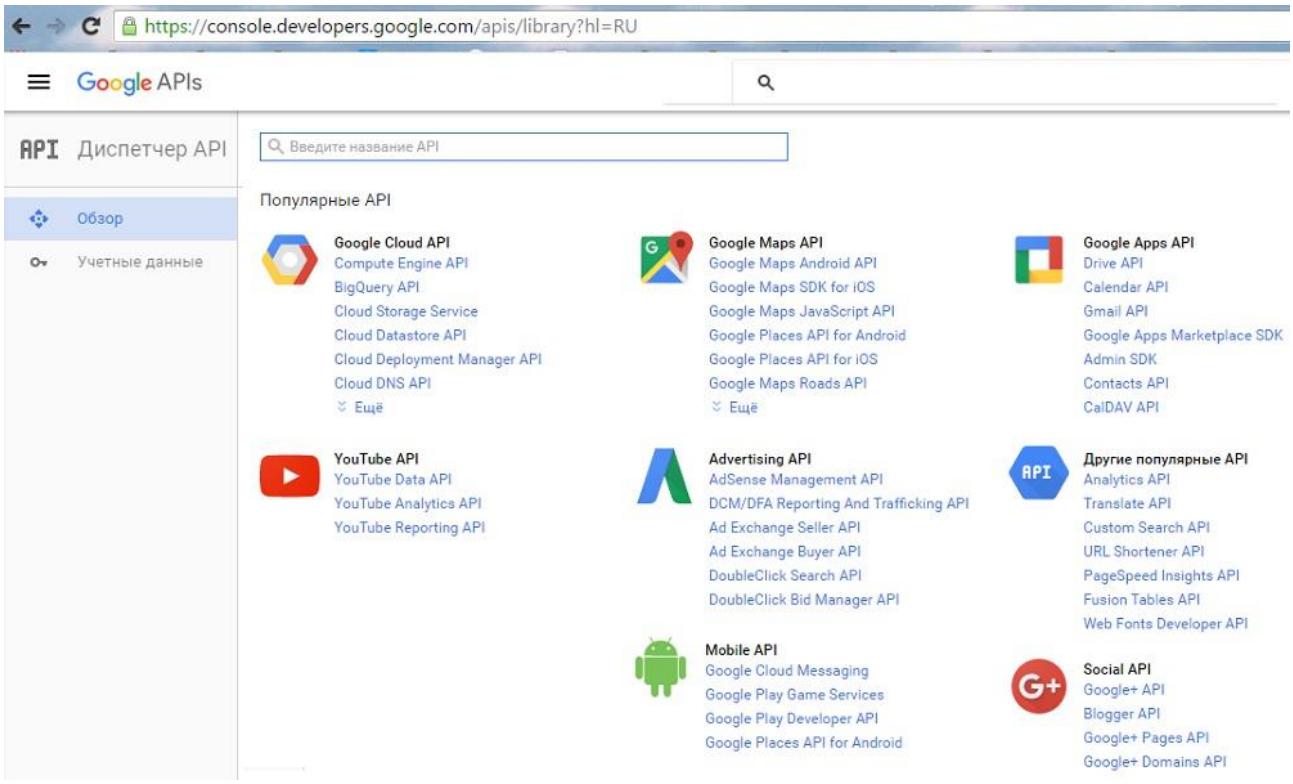

Рис. 6. Диспетчер API Google

Получив ответ на API-запрос, вы можете выполнить парсинг JSON-данных. Например, ответ на запрос, приведенный в начале главы, может быть обработан следующим образом:

```
import json

from urllib.request import urlopen

def getCountry(ipAddress):
    response = urlopen("http://freegeoip.net/json/"+ipAddress).read().decode('utf8')
    responseJson = json.loads(response)
    return responseJson.get("country_code")

print(getCountry("50.78.253.58"))
```

Этот код выводит код страны для IPадреса: 50.78.253.58. Библиотека, используемая для парсинга JSON-данных, является частью стандартной библиотеки языка Python.

Глава 5. Хранение данных

Несмотря на то что вывод результатов в терминале довольно забавен, это невероятно неудобно, когда дело касается агрегации и анализа данных. Чтобы веб-скраперы были полезны, вы должны уметь сохранить собранную ими информацию.

Есть два основных способа хранить мультимедийные файлы: в виде ссылки или загрузив сам файл. Оба способа имеют, как преимущества, так и недостатки. В Python 3.x для скачивания файлов с любого удаленного URL-адреса можно использовать urllib.request.urlretrieve. Следующий код скачивает логотип из http://pythonscraping.com и сохраняет его в виде файла logo.jpg в том же самом каталоге, из которого запущен скрипт:

```
from urilib.request import urlretrieve

from urilib.request import urlopen
```

```
from bs4 import BeautifulSoup

html = urlopen("http://www.pythonscraping.com")

bsObj = BeautifulSoup(html)

imageLocation = bsObj.find("a", {"id": "logo"}).find("img")["src"]

urlretrieve (imageLocation, "logo.jpg")
```

CSV (Comma Separated Values – значения, разделенные запятыми) является одним из самых популярных файловых форматов, в котором хранятся данные электронных таблиц. Он поддерживается Microsoft Excel и многими другими приложениями из-за своей простоты. Изменить CSV-файл или создать его с нуля можно воспользовавшись питоновской библиотекой csv:

```
import csv

csvFile = open("../files/test.csv", 'wt')

try:
        writer = csv.writer(csvFile)
        writer.writerow(('number', 'number plus 2', 'number times 2'))
        for i in range(10):
                writer.writerow( (i, i+2, i*2))
finally:
        csvFile.close()
```

Если ../files/test.csv не существует, Python создаст каталог и файл автоматически. Если он уже существует, Python перезапишет test.csv новыми данными.

На сегодняшний день MySQL является самой популярной системой управления базами данных с открытым исходным кодом. «Реляционные данные» – это данные, которые содержат взаимосвязи. Например, «Пользователь A идет учиться в учреждение B», где пользователя A можно найти в таблице «пользователей» базы данных, а учреждение B можно найти в таблице «учреждений». MySQL предлагает удобный способ взаимодействия с данными с помощью интерфейса командной строки. Например, команда, приведенная ниже, возвращает список всех пользователей в вашей базе данных с именем «Ryan»:

```
SELECT * FROM users WHERE firstname = "Ryan"
```

Для установки MySQL под Windows для начала вам нужно создать учетную запись Oracle. Далее воспользуйтесь установщиком. Используйте настройки по умолчанию, за одним исключением: на странице Setup Type (Тип установки) я рекомендую вам выбрать Server Only (Только сервер), чтобы избежать установки дополнительного программного обеспечения и библиотек Microsoft (см. также Ручная установка MySQL на Windows).

К сожалению, поддержка Python в MySQL не предусмотрена. Однако существует большое количество библиотек с открытым исходным кодом как для Python 2.x, так и для Python 3.x, которые можно использовать для работы с базой данных MySQL. Например, PyMySQL. После установки модуля вы сможете выполнить следующий скрипт (не забудьте добавить root-пароль для вашей базы данных):

```
import pymysql

conn = pymysqi.connect(host='127.0.0.1', unix_socket='/tmp/mysqi.sock', user='root', passwd=None, db='mysqi')

cur = conn.cursor()

cur.execute("USE scraping")

cur.execute("SELECT * FROM pages WHERE id=1")

print(cur.fetchone())
```

```
cur.close()

conn.close()
```

Здесь используются два новых типа объектов: объект Connection (conn) и объект Cursor (cur). Модель подключение/курсор широко используется в программировании баз данных. Подключение отвечает за подключение к базе данных. Курсор предназначен для работы с базой данных. Подключение может иметь много курсоров. Самое частое, что вы, вероятно, хотите сделать, начав работу с MySQL, – это сохранить результаты скрапинга в базе данных.

Глава 6. Чтение документов

Интернет не является хранилищем HTML-файлов. Это хранилище информации, где HTML-файлы часто используются в качестве средства ее визуального представления. Не имея возможности прочитать различные типы документов, включая текст, PDF, изображения, видео, электронные письма и т.д., мы теряем огромную часть имеющихся данных.

На базовом уровне все документы кодируются нулевыми и единичными битами. Помимо того, есть алгоритмы кодирования, которые задают такие параметры, как «количество бит на символ» или «количество бит, выделенных для записи цвета одного пикселя» (при работе с графическими файлами). Кроме того, вы можете использовать алгоритм сжатия, как в случае с PNG-файлами.

Наиболее распространены следующие типы файлов: TXT, PDF, PNG, GIF.

Обычно, когда мы извлекаем страницу с помощью urlopen, мы превращаем ее в объект BeautifulSoup, чтобы выполнить парсинг HTML-структуры. В случае с текстовым документом мы можем прочитать страницу напрямую:

```
from urllib.request import urlopen

textPage = urlopen("http://www.pythonscraping.com/pages/warandpeace/chapter1.txt")

print(textPage.read())
```

Краткий обзор типов кодировки. В начале 1990-х некоммерческая организация под названием Unicode Consortium предложила универсальную кодировку текста, которая была названа UTF-8, (Universal Character Set – Transformation Format 8 bit, Универсальный набор символов – Формат преобразования 8 бит). Для хранения символов доступен 21 бит информации, что позволяет отобразить 2^{21} = 2 097 152 возможных символов.

Стандарт кодировки ASCII, применяемый с 1960-х годов, использует 7 битов для кодирования каждого символа, что в общей сложности составляет 2^7, или 128, символов. Этого достаточно для латинского алфавита (как для прописных, так и строчных букв), знаков препинания и всех остальных символов, которые можно встретить на англоязычной клавиатуре.

Проблема стандарта Unicode заключается в том, что любой документ на иностранном языке занимает гораздо больше места, чем должен. Несмотря на то что в вашем языке может использоваться только 100 символов или около того, вам понадобится, по крайней мере, 16 бит для каждого символа, а не просто 8 бит, как в случае с английской версией кодировки ASCII. ISO решает эту проблему путем создания отдельных кодировок для каждого языка. Хотя популярность документов, использующих кодировку ISO, сокращается в последние годы, около 9% сайтов в Интернете по-прежнему предпочитают ее.

Можно явно задать строку в кодировке UTF-8, которая правильно отформатирует вывод кириллических символов:

```
from urllib.request import

urlopen textPage = urlopen("http://www.pythonscraping.com/pages/warandpeace/chapter1ru.txt")

print(str(textPage.read(), 'utf8'))
```

Использование этого принципа в BeautifulSoup и Python 3.x выглядит так:

```
html = urlopen("http://en.wikipedia.org/wiki/Python_(programming_language)")

bsObj = BeautifulSoup(html)
```

```
content = bsObj.find("div", {"id":"mwcontenttext"}).get_text()

content = bytes(content, "UTF8")

content = content.decode("UTF8")
```

Чтобы узнать, какую кодировку использовать, можно воспользоваться информацией, содержащейся в теге, расположенном в разделе <head> сайта. Большинство сайтов, особенно англоязычных, содержат тег:

```
<meta charset="utf8" />
```

Тогда как вебсайт European Computer Manufacturers Association содержит этот тег:

```
<META HTTPEQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
```

ECMA была одним из первых разработчиков кодировки ISO, поэтому неудивительно, что ее веб-сайт использует эту кодировку.

Компания Adobe совершила революцию, разработав в 1993 году формат Portable Document Format. PDF-файлы позволили пользователям различных операционных систем просматривать графические и текстовые документы в едином универсальном формате. Для парсинга PDF можно использовать библиотеку PDFMiner3K.

Примерно до 2008 года продукты Microsoft Office использовали собственный файловый формат .doc. Этот бинарный формат файла было неудобно читать, и он плохо поддерживался другими текстовыми процессорами. Желая идти в ногу со временем и принять стандарт, который использовался бы другими программами, компания Microsoft решила использовать файловый формат Open Office XML, который поддерживается различными программами, в том числе программами с открытым кодом.

К сожалению, в Python поддержка этого файлового формата, используемого Google Docs, Open Office и Microsoft Office, по-прежнему не велика. Существует библиотека pythondocx, но она лишь дает пользователям возможность создавать документы и читать основную информацию о файле, например, размер и название файла, а не фактическое содержание.

До этого момента ваши вебскраперы были относительно бестолковыми. Они не в состоянии извлечь информацию, за исключением случаев, когда она размещена в удобном формате на сервере. Формы, интерактивный дизайн сайта и даже JavaScript могут поставить ваш скрапер в тупик. Короче говоря, они не годятся для извлечения информации, за исключением случаев, когда эта информация уже оптимизирована для извлечения.

Во второй части книги рассмотрен продвинутый скрапинг:

- Очистка данных
- Чтение и запись естественных языков
- Краулинг сайтов, использующих веб-формы
- Скрапинг JavaScript-кода
- Обработка изображений и распознавание текста
- Обход ловушек в ходе скрапинга
- Тестирование вашего сайта с помощью скраперов
- Скрапинг с помощью удаленных серверов

В приложении говорится о правовых и этических аспектах веб-скрапинга (товарных знаках, авторских правах и патентах); описано несколько судебных дел в практике веб-скрапинга (eBay против Bidder’s Edge и посягательство на движимое имущество, США против Орнхаймера и Закон о компьютерном мошенничестве и злоупотреблении, Филд против Google: авторское право и robots.txt).