**Извлечение данных с web-страниц с помощью кода на языке Python**

Если необходимые вам данные разбросаны по разным HTML-страницам для их извлечения применяется *скрапинг*. Вы создаете код для автоматического посещения определенного перечня страниц, получения конкретного контента с этих страниц и сохранения его в базе данных или в текстовом файле.[[1]](#footnote-1)

Скажем, вы хотите 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. К счастью, сайт [Weather Underground](https://www.wunderground.com/) предоставляет исторические данные о погоде. И плохая новость: на одной странице сведения можно получить только за один день (рис. 1).



Рис. 1. Температура в Москве по данным Weather Underground

Проходим по меню *More* –> *Historical Weather* и выбираем определенную дату (рис. 2).



Рис. 2. Окно поиска архивных данных о погоде

Нажмите *Submit*, откроется другая страница, где вам будут представлены подробные данные о погоде в выбранный вами день (рис. 3).



Рис. 3. Подробные сведения о погоде в выбранный день

Допустим вас интересует максимальная температура. Вы могли бы посетить 365 страниц и собрать требуемые данные. Но процесс можно ускорить с помощью кода. Для этого обратитесь к языку программирования Python и к библиотеке под названием BeautifulSoup, созданной Леонардом Ричардсоном.

Если вы работаете под MacOS, тогда Python у вас уже должен быть инсталлирован. Если вы работаете под Windows, зайдите на [python.org](https://www.python.org/) и, следуя инструкциям, скачайте и установить Python (я установил версию Python 3.5.1 с помощью [Windows x86-64 web-based installer](https://www.python.org/ftp/python/3.5.1/python-3.5.1-amd64.exe)). Чтобы скачать Beautiful Soup, зайдите на [crummy.com](http://www.crummy.com/software/) (я скачал самую свежую [версию 4.4.1](https://www.crummy.com/software/BeautifulSoup/bs4/download/4.4/beautifulsoup4-4.4.1.tar.gz)). Распакуйте архив, зайдите в распакованную папку и в командной строке выполните:

> python setup.py install

Создайте файл в текстовом редакторе (например, в блокноте) и сохраните его как get-weather-data.py. Теперь вы можете заняться написанием кода. Загрузите страницу, на которой представлены исторические сведения о погоде (как на рис. 3). URL страницы с информаций о погоде в Москве за 1 января 2015 выглядит так:

https://www.wunderground.com/history/airport/UUEE/2015/1/1/DailyHistory.html?req\_city=Moscow+-+Sheremet%5C%27Ye&req\_state=&req\_statename=Russia&reqdb.zip=00000&reqdb.magic=1&reqdb.wmo=WUUEE

Если в этом адресе вы удалите все, что следует за .html, страница будет по-прежнему загружаться, так что избавьтесь от этих лишних символов:

https://www.wunderground.com/history/airport/UUEE/2015/1/1/DailyHistory.html

В URL дата указана как /2015/1/1. Чтобы загрузить страницу со сведениями о 2 января 2015 г. без использования выпадающего меню, просто измените параметр даты так, чтобы URL выглядел следующим образом:

https://www.wunderground.com/history/airport/UUEE/2015/1/2/DailyHistory.html

Теперь загрузите страницу с помощью Python, используя функцию *urlopen,* хранящуюся в модуле *urllib.request* библиотеки *urllib*. Для начала импортируйте эту функцию в программу:

from urllib.request import urlopen

Чтобы загрузить страницу с данными о погоде за 1 января, введите код:

page = urllib.urlopen("www.wunderground.com/history/airport/UUEE/2015/1/2/DailyHistory.html")

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

from bs4 import BeautifulSoup

Используйте функцию BeautifuLSoup, чтобы прочитать страницу, и произвести ее анализ.

soup = BeautifulSoup(page)

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 код:

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? Введите:

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 извлечь максимальную температуру в Москве 1 января 2015 года. Чтобы найти ее, нужно потрудиться чуть подольше, но метод применяется все тот же. Вам нужно выяснить, что вставить в findAll(), а потому просмотрите исходный HTML. Это легко сделать в веб-браузере. В Google Chrome, например, кликните на странице правой кнопкой мыши и выберите *Просмотреть код*. Появится окно с HTML-кодом (рис. 4).



Рис. 4. Исходный HTML-код страницы Weather Underground

Прокрутите вниз до того места, где показана максимальная температура, или воспользуйтесь функцией поиска в браузере. Строка замыкается тегом <span> с классом wx-value. Это и есть ваш ключ. Вы теперь можете найти все элементы на странице с классом wx-value.

wx-values = soup.findAll(attrs={"class":"wx-value"})

Как и в предыдущем примере, это дает вам в руки перечень всех случаев употребления wx-value. Вас интересует второй из них, который вы найдете с помощью строчки:

print wx-values[1]

Вуаля! Вы впервые в жизни извлекли нужное вам значение из HTML-кода веб-страницы. Следующий шаг — проанализировать все страницы за 2015 год и извлечь из них необходимые данные. Для этого вернитесь к первоначальному URL:

https://www.wunderground.com/history/airport/UUEE/2015/1/1/DailyHistory.html

Помните, как вы вручную изменили адрес, чтобы получить сведения о той дате, которая вас интересовала? Приведенный выше код относится к 1 января 2015 года. Если вам нужна страница за 2 января 2015 года, просто измените ту часть URL, в которой указана дата, на соответствующую. Чтобы получить данные за все дни 2015 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-full.py.

from urllib.request import urlopen

from bs4 import BeautifulSoup

# Create/open a file called wunder.txt (which will be a comma-delimited file)

f = open('wunder-data.txt', 'w')

# Iterate through year, month, and day

for y in range(2015,2016):

 for m in range(1, 13):

 for d in range(1, 32):

 # Check if leap year

 if y%400 == 0:

 leap = True

 elif y%100 == 0:

 leap = False

 elif y%4 == 0:

 leap = True

 else:

 leap = False

 # Check if already gone through month

 if (m == 2 and leap and d > 29):

 continue

 elif (m == 2 and d > 28):

 continue

 elif (m in [4, 6, 9, 10] and d > 30):

 continue

 # Open wunderground.com url

 url = "http://www.wunderground.com/history/airport/UUEE/"+str(y)+ "/" + str(m) + "/" + str(d) + "/DailyHistory.html"

 page = urlopen(url)

 # Get temperature from page

 soup = BeautifulSoup(page.read(), "html.parser")

 # dayTemp = soup.body.wx-values

 dayTemp = soup.findAll(attrs={"class":"wx-value"})[1].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 = dStamp + '.' + mStamp + '.' + str(y)

 # Write timestamp and temperature to file

 f.write(timestamp + '\tab' + dayTemp + '\n')

# Done getting data! Close file.

f.close()

Вы наверняка узнали первые две строчки кода, с помощью которых вы импортировали необходимые библиотеки, — urllib и 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 папке, куда вы сохранили свой скрипт.

С помощью следующего блока, используя циклы for, компьютеру отдается команда просмотреть каждый год, месяц и день. Функция range() генерирует последовательность чисел, используемых для итераций цикла. Эта функция генерирует последовательность, начинающуюся с нуля и кончающуюся числом в скобках, *не включая его*. Поэтому, если вы укажитец

for y in range(2015):

то получите последовательность 0, 1, …, 2014. Вы можете указать в качестве параметров функции range() два числа, разделенных занятой, — начальную и конечную величину. Например, range (1,13) генерирует последовательность 1, 2, ..., 12.

Внешний цикл for проверяет, является ли год високосным. 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.

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 (последнее значение не включается). 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.

Следующие строчки кода вы использовали 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 urllib, произведите анализ контента с помощью Beautiful Soup, а затем извлеките максимальную температуру, с помощью второго появления класса wx-values.

Предпоследний кусок кода отвечает за составление метки времени исходя из года, месяца и дня. Метка времени дается в формате «ДДММГГГГ». Здесь вы можете задать любой формат, но на данном этапе чем проще — тем лучше. И наконец, в файл 'wunder-data.txt' с помощью метода write() добавляются дата и температура. По окончании работы со всеми месяцами и днями файл закрывается. Осталось лишь запустить программу.

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 2015 года. Когда выполнение скрипта завершится, у вас в рабочем каталоге появится файл под названием wunder-data.txt (рис. 5). Откройте его, и там вы найдете нужные вам данные в формате с разделителями табуляцией. В первой колонке вы увидите дату, во второй – температуру.



Рис. 5. Извлеченные данные в файле с разделителем табуляцией

Обратите также внимание, что, если вы будете запрашивать данные по будущему периоду (дата еще не наступила), то получите некорректный ответ. Запрос найдет второе вхождение тега с классом wx-value, но оно не будет соответствовать максимальной температуре выбранного дня (рис. 6).



Рис. 6. Для будущего периода второй тег класса wx-value не соответствует максимальной температуре выбранного дня

\* \* \*

Любопытна история написания этой заметки. Мой друг несколько лет назад подарил мне книгу Нейтана Яу Искусство визуализации в бизнесе. Поскольку меня интересует обработка и представление информации, книга мне показалась любопытной, и я начал ее читать. Но… буквально с первых страниц автор уводит с проторенных дорожек. Критикуя Excel за его недостаточную мощь и гибкость, автор предлагает использовать программную обработку на основе языка Python. Тогда мне это показалось слишком сложным.

Но около месяца назад я вновь вернулся к книге Нейтана Яу. И теперь использование Python меня не отпугнуло. Я решил написать конспект книги, а, чтобы не перегружать его, специфические темы опубликовать отдельно. Первая из таких заметок перед вами. Надо сказать, что далась она непросто.

Для начала, код Нейтана Яу не запустился. Проблемы подстерегали практически на каждом шагу. Выяснилось, что код написан для Python 2, а последняя версия – это Python 3.5. Изучение Интернета подсказало, что язык был существенно переработан при переходе с версии 2 на 3. Мне показалось странным, потчевать читателей устаревшим кодом. Поэтому я установил последнюю из доступных на данный момент версий Python 3.5.1.

Чтобы попытаться самому переписать код, или хотя бы более грамотно задавать вопросы на форумах, я прочитал книгу [Майк МакГрат. Программирование на Python для начинающих](http://baguzin.ru/wp/?p=14338). Книга мне понравилась. Очень хорошо структурирована, сопровождается файлами с примерами. То, что надо! Она мне здорово помогла.

Следующая проблема была с запуском модуля BeautifulSoup. После импорта внутрь кода Python модуль не был виден. Я менял папки, запускал скрипт setup.py, ничего не помогало. Интенсивный поиск в Интернете натолкнул меня на только что вышедшую в издательстве [ДМК Пресс](http://dmkpress.com/catalog/computer/programming/python/978-5-97060-223-2/) книгу Райана Митчелла Скрапинг вебсайтов с помощью Python. Оцените, книга вышла 30 апреля 2016 г.! Здесь я почерпнул знания о библиотеке urllib, и о том, как установить модуль BeautifulSoup под Python 3. Критически важным было выполнить в командной строке

> python setup.py install

Эта команда конвертирует код модуля BeautifulSoup, адаптируя его к версии Python 3.

Далее оказалось, что с момента написания Нейтаном Яу своей книги на сайте Weather Underground произошли изменения в стилевом оформлении CSS. Поменялось имя тега, используемого для представления значения температуры: с *nobr* на *wx-value*.

И наконец, с переходом на версию Python 3, наверное, поменялся синтаксис извлечения значения из тега. У Нейтана Яу было:

dayTemp = soup.findAll(attrs={"class":"nobr"})[5].span.string

У меня:

dayTemp = soup.findAll(attrs={"class":"wx-value"})[1].string

Т.е., [ ].span.string следовало заменить на [ ].string. Я нигде не мог найти подсказки, что же следует изменить в первоначальном коде, чтобы извлечь не весь тег

<span class="wx-value">1</span>,

а только его значение

1.

Обращение на форум [python.su](http://python.su/forum/topic/29971/) не помогло. Я применил метод научного тыка, и после нескольких попыток получил то, что нужно.

Успехов вам в освоении языка Python!

1. Заметка написана на основе материалов книги Нейтан Яу. Искусство визуализации в бизнесе. – М.: Манн, Иванов и Фербер, 2013. – С. 44–53. [↑](#footnote-ref-1)