**Форматирование данных с помощью кода** **на языке Python**

Excel и другие программы, работающие по методу «указал и щелкнул», относительно просты в освоении и покрывают большинство потребностей. Однако, если вы хотите создавать выдающиеся диаграммы и инфографику, вам не обойтись без написания кода для форматирования данных.[[1]](#footnote-1) Посмотрите, как можно легко переключаться с одного формата данных на другой с помощью всего нескольких строчек кода Python.

В качестве исходных данных возьмем файл, полученный путем скрапинга сайта [Weather Underground](https://www.wunderground.com/) (подробнее см. [Извлечение данных с web-страниц с помощью кода на языке Python](http://baguzin.ru/wp/?p=14390)). Файл wunder-data.txt содержит данные о максимальных ежедневных температурах в Москве за 2015 год (рис. 1).



Рис. 1. Максимальные ежедневные температуры в Москве за 2015 год

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

<weather\_data>

<observation>

<date>01.01.2015</date>

<max\_temperature>1</max\_temperature>

</observation>

<observation>

<date>02.01.2015</date>

<max\_temperature>1</max\_temperature>

</observation>

<observation>

<date>03.01.2015</date>

<max\_temperature>2</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 программу csv-to-xml.py (рис. 2; подробнее о языке см. [Майк МакГрат. Программирование на Python для начинающих](http://baguzin.ru/wp/?p=14338)).



Рис. 2. Программа конвертера csv-to-xml

В первой строке вы импортируете стандартный модель Python. Вторая строчка кода открывает для чтения wunder-data.txt, используя функцию open(), а затем загружает его при помощи метода csv.reader(). Обратите внимание на то, что в качестве разделителя здесь стоит знак табуляции. 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. В каждой строчке есть два значения: даты и максимальной температуры. Завершите перевод в XML закрывающим тегом: print('</weather\_data>').

Модернизируйте ваш код, чтобы не просто любоваться на экране работой вашей программы, а записать результаты конвертации в файл \*.xml (рис. 3).



Рис. 3. Код конвертера csv-to-xml-write и файл wunder-data.xml, открытый в Internet Explorer

В третьей строчке вы открываете файл wunder-data.xml на запись. А если такого файла нет, то создаете его. В четвертой строке вы записываете первую строку в файл: <weather\_data>. Далее действуете, как и в первоначальном коде, но вместо печати на экран, последовательно записываете строки в файл. Вы добавили знаки табуляции, чтобы образовать структуру файла. В конце закрываете файл.

Если бы вам предстояло конвертировать полученный XML обратно в CSV, логика действий была бы точно такой же. Отличие состоит лишь в том, что вы используете для парсинга XML-файла другой модуль (рис. 4).



Рис. 4. Обратный конвертер xml-to-csv

Вместо CSV-модуля в первой строке кода вы импортируете BeautifulSoup (подробнее об этом модуле см. [Извлечение данных с web-страниц с помощью кода на языке Python](http://baguzin.ru/wp/?p=14390) или [Райан Митчелл. Скрапинг веб-сайтов с помощью Python](http://baguzin.ru/wp/?p=14420)). Вы открываете XML-файл для чтения, используя функцию open(), а затем загружаете всё содержимое файла в переменную xml. На данном этапе контент сохраняется в виде строки. Для начала парсинга передайте xml-строку в BeautifulSoup. Используйте findAll() для того, чтобы извлечь все теги <observation>. И, наконец, как и при переводе CSV в XML, задайте цикл по каждому тегу <observation> и отобразите в каждой строк сначала значение тега <date>, затем запятую, а затем значение тега <max\_temperature>. Результат работы кода показан в нижней части рис. 4.

Для полноты картины вот вам еще и код, с помощью которого можно конвертировать CSV в формат JSON (рис. 5).



Рис. 5. Конвертер csv-to-json

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.

Обратите внимание на небольшое новшество – оператор if-else. Его назначение — проверить, не является ли текущая итерация по рядам с данными последней. Если нет, тогда печатается запятая после фигурной скобки. Если это последняя итерация, запятая не нужна.

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. В рамках цикла вы можете подсчитать скользящее среднее или разницу между текущим и предыдущим днями.

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 (рис. 6).



Рис. 6. Добавим колонку с признаком положительной температуры (значение 0 в третьей колонке)

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 наглядно демонстрирует, как вы можете развить эти принципы для форматирования или обогащения данных как вашей душе угодно. Запомните эти три шага: загрузка, цикл, обработка!

1. Заметка написана на основе материалов книги Нейтан Яу. Искусство визуализации в бизнесе. – М.: Манн, Иванов и Фербер, 2013. – С. 62–67. [↑](#footnote-ref-1)