Практическое занятие № 5. XPath и web scraping

В этом задании вы будете практиковаться в обработке xml-файлов при помощи XPath. Так же как и для работы с xml-файлами в Python это осуществляется с помощью библиотеки `lxml`. Вы можете ее подключить при помощи команды

```
from lxml import etree
```

1. Простой пример

Скачайте xml-файл `inventory.xml`, который находится по ссылке http://bit.ly/2efjlLM.

Следующий код является примером использования XPath-выражений для анализа xml-файлов:

```
from lxml import etree

doc = etree.parse('inventory.xml')  ## Загружаем xml-файл
xp1 = etree.XPath('//book/author')  ## Функция, которая будет искать узлы согласно XPath-выражению
for r in xp1(doc.getroot()):        ## Проходим по всем узлам, который удовлетворяют XPath-выражению
    # Анализ элемента r: r.tag, r.text, for node in r,...
```

В браузере откройте страничку http://bit.ly/2eh2HcU и поэкспериментируйте с XPath-выражениями, приведенными на этой странице на примере файла `inventory.xml`.

2. Модуль `requests`

Для скачивания данных из интернета средствами Python можно использовать модуль `requests`, который очень хорошо работает в связке с модулем `lxml`:

```
from lxml import etree
import requests
htmlreq = requests.get('http://www.yandex.ru')

parser = etree.HTMLParser()
doc = etree.HTML(htmlreq.text)
```

3. Где сейчас находится автобус?

Исходными данными для этой задачи будут материалы сайта http://www.ot76.ru/mob/.

Напишите программу, которой на вход подается номер автобуса, а она возвращает где сейчас находятся автобусы и когда они приедут на следующую остановку.

4. Полезные факты о модуле `lxml`

1. Загрузить файл и построить DOM-дерево можно при помощи функции `etree.parse(filename)`, где `filename` — имя xml-файла.
2. Как только вы построили DOM-дерево `doc`, получить корневой элемент можно при помощи метода `doc.getroot()`.
3. У элемента дерева `node` есть два полезных атрибута: `node.tag` (xml-тег элемента) и `node.text` (текст, который находится между открывающим и закрывающим тегом, включая пробелы и переводы строки).
4. Рекурсивно обойти дерево начиная с узла `node` можно при помощи такой конструкции:

   ```
   for item in node.iter():
       # обрабатываем узел item
   ```

5. Больше примеров есть в Google.