**Тема «json и requests»**

Одним из самых больших источников данных является Глобальная сеть Интернет, откуда их можно выгрузить, используя, например, Python. Для этого чаще всего используются http-запросы, доступные в модуле requests

**Requests**

**Установка**

Чаще всего модуль requests устанавливается вместо с установкой Python. Однако если этого не произошло, можно установить данный модуль с помощью pip следующим образом.

|  |
| --- |
| pip install requests |

**Основные виды http-методов**

HTTP – это протокол взаимодействия между двумя компьютерами (клиентом и сервером), построенное на базе сообщений, называемых запрос (request) и ответ (response). Клиент инициирует соединение и посылает запрос, а сервер ожидает соединение для получения запроса и, проведя необходимые действия, возвращает ответ с результатом. Каждое сообщение состоит из трех частей: стартовая строка, заголовки и тело. При этом обязательной является только стартовая строка.

Метод HTTP – последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом.

Для разграничения действия с ресурсами на уровне HTTP-методов придуманы следующие варианты:

* GET – используется для запроса содержимого указанного ресурса
* POST – используется для передачи пользовательских данных заданному ресурсу
* PUT – используется для обновления ресурса
* DELETE – используется для удаления ресурса
* И др.

**Метод GET в requests**

GET является одним из самых популярных HTTP-методов, он указывает на то, что происходит попытка извлечь данные из определенного ресурса. Для того, чтобы выполнить запрос GET, используется requests.get()

|  |
| --- |
| **import** **requests**  response = requests.get('https://yandex.ru') |

В результате возвращается объект ответ (response), который содержит ответ на запрос.

**HTTP коды состояний**

Самыми первыми данными, которые получены через Response, являются коды состояния. Коды состояния сообщают о статусе запросе. Например, в указанном выше коде может последовать строчка (или при вызове команды print(response) или print(response.status\_code))

|  |
| --- |
| **import** **requests**  response = requests.get('https://yandex.ru')  <Response [200]> |

В данном случае число 200 является статусом состояние. Статус состояния 200 означает, что запрос успешно выполнен. Существует множество различных кодов состояний (например, 404 – «не найдено» и др.) (<https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>).

**Настройка GET-запроса**

Зачастую необходимо отправить какие-то данные в строке запроса URL. При настройке URL вручную эти данные будут представлены в нем в виде пар ключ/значение после знака вопроса. Например, httpbin.org/get?key=val. Requests позволяет передать эти аргументы в качестве словаря, используя аргумент params. Если вы хотите передать key1=value1 и key2=value2.

|  |
| --- |
| payload = {'key1': 'value1', 'key2': 'value2'}  r = requests.get('http://httpbin.org/get', params=payload)  >>> print(r.url)  http://httpbin.org/get?key2=value2&key1=value1 |

**Получение содержимого response**

**1.response.text**

Requests автоматически декодирует содержимое ответа сервера. Большинство кодировок Юникода легко декодируется. Когда пользователь делает запрос, Requests делает предположение о кодировке ответа основе заголовков HTTP. Кодировка текста, угадываемая запросами, используется при обращении к r.text.

|  |
| --- |
| >>> import requests  >>> r = requests.get('https://api.github.com/events')  >>> r.text  u'[{"repository":{"open\_issues":0,"url":"https://github.com/... |

**2.response.json()**

Данные способ позволяет декодировать ответ в формат JSON, однако декодирование возможно не всегда (в этом случае вернется исключение).

|  |
| --- |
| >>> import requests  >>> r = requests.get('https://api.github.com/events')  >>> r.json()  [{u'repository': {u'open\_issues': 0, u'url': 'https://github.com/... |

**3.response.content**

Данный способ используется для получения к телу ответа в виде байтов для не текстовых ответов

**LXML**

**Установка lxml**

Библиотека lxml позволяет обрабатывать разметку XML и HTML на Python, Кроме того, в ней присутствует возможность разложения элементов документа в дерево. Установка библиотеки производится через pip

|  |
| --- |
| pip install lxml |

**Получение объекта html для обработки из GET-запроса**

Для получения объекта html из GET-запроса существует функция fromstring(). Она преобразует исходное содержимое в элемент с указанной разметкой. Преимуществом данного объекта является возможность навигации по нему через язык запросов xpath

|  |
| --- |
| response = requests.get(url)  **if** response.status\_code == **200**:  html = fromstring(response.text) |

**Использование xpath**

Xpath – это язык запросов к элементам xml и xhtml. Для получения интересующих данных необходимо создать запрос, описывающий эти данные. Для того, как он работает, рассмотрим часть html-кода страницы.

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml" lang="en">  <head>  <meta http-equiv="X-UA-Compatible" content="IE=Edge" />  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Quickstart **&#8212;** Requests 2.22.0 documentation</title>  <link rel="stylesheet" href="../../\_static/custom.css" type="text/css" />  <script type="text/javascript" src="../../\_static/readthedocs-data.js"></script>  <script type="text/javascript">  READTHEDOCS\_DATA['page'] = "user/quickstart"  READTHEDOCS\_DATA['source\_suffix'] = ".rst"  </script>  <script type="text/javascript" src="https://assets.readthedocs.org/static/javascript/readthedocs-analytics.js"></script>  </head><body>  <div class="document">  <div class="documentwrapper">  <div class="bodywrapper"> |

Как видно, html-код страницы состоит из иерархичных заголовков и содержимого. Рассмотрим синтаксис запроса на примере:

|  |
| --- |
| “//p/a” |

// - рекурсивный спуск на ноль и более уровней иерархии

p/a – xpath путь до конкретного элемента. Путь состоит из шагов адресации и условия проверки узлов (p, a, ul, li и т.д.) Шаги разделяются символом ‘/’ (косая черта).

**Xpath в lxml**

Для использования xpath в python осуществляет вызовом соответствующей функции для объекта xml или html.

|  |
| --- |
| response = requests.get(url)  **if** response.status\_code == **200**:  html = fromstring(response.text)  all\_urls = html.xpath("//a")  for element\_url in all\_urls:  url = element\_url.xpath("@href") |

**JSON**

**Описание формата JSON**

JSON представляет собой текстовый формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. JSON – текстовый формат, полностью независимый от языка реализации. JSON основан на двух структурах данных:

* Коллекция пар ключ/значение. В разных языках это концепция может быть реализована по-разному (наиболее подходящий аналог в Python - словарь).
* Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность (в Python - список).

Пример файла данного формата приведен ниже

|  |
| --- |
| [  {  "orderID": **12345**,  "shopperName": "Ваня Иванов",  "contents": [  {  "productID": **34**,  "productName": "Супер товар",  "quantity": **1**  },  {  "productID": **56**,  "productName": "Чудо товар",  "quantity": **3**  }  ],  "orderCompleted": **true**  }] |

**Установка**

Для работы с файлами формата JSON чаще всего используют одноименную библиотеку json, которая может быть установлена через pip.

|  |
| --- |
| pip install json |

**Чтение из JSON**

Чтения данных из файлов формата JSON напоминает чтения из обычных файлов. Для этого используется функция load

|  |
| --- |
| def read\_json(json\_file):  with open(json\_file, **'r'**, encoding=**'utf-8'**) as file:  return json.load(file) |

В результате возвращается объект типа dict, поэтому дальнейшие операции с объектом происходят, как с обычным словарем.

|  |
| --- |
| def boolean\_choice(json\_data, field):  for item in json\_data:  value = item[field]  # TO DO |

**Запись в JSON**

Запись в файлы формата JSON осуществляются чаще всего через словарь и напоминает запись в обычный файл. Пусть переменная data представляет собой некоторый список, каждый из элементов которого является словарем.

|  |
| --- |
| with open(file\_path, **"w"**, encoding=**"utf-8"**) as file:  json.dump(data, file, ensure\_ascii=False, indent=4) |

Параметр ensureascii отвечает за правильное отображение символов ASCII-таблицы, indent отвечает за отступ у вложенных объектов.

**Практика**

Получить статистику по лекарствам, отпускаемых по рецепту и без него.

**Электронные источники**

1. Requests <https://requests.readthedocs.io/en/master/>
2. Resuests <https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests>
3. Xpath <https://ru.wikipedia.org/wiki/XPath>
4. JSON <https://ru.wikipedia.org/wiki/JSON>
5. JSON <https://www.json.org/json-ru>
6. JSON <https://docs.python.org/3/library/json.html>