Язык Python. Часть 2

Лекция 4

Статические ресурсы

- Сами данные меняются нечасто
- Хранятся на сервере как файлы
- Можно считать, что есть какой-то каталог в файловой системе сервера
- В нем эти файлы лежат
- Путь в URL - это путь к файлу, начиная от этого каталога

Статические ресурсы

- http://lib.ru/POEEAST/GOMER/gomer01.txt
- На сервере есть некий каталог
- В нем подкаталог POEEAST
- А в нем - GOMER
- А там - файл gomer01.txt
- Не обязательно все буквально так - но это очень популярный вариант

Статические ресурсы: ограничения

- Хорошо подходит для интернет-библиотек
- Для новостных сайтов - уже хуже
- Но как-то еще можно
- Совсем никак - форумы, соцсети и т.п.

Динамические ресурсы

- На стороне сервера работает код
- Код порождает HTML-документ
- Например, страничку дискуссии
- Или заголовки новостей по данной теме
- На основании содержимого баз данных и т.п.

Динамические ресурсы

- Пример: https://www.google.com/search?q=python
- Путь - это идентификатор программы, порождающей ресурс
- А у программы могут быть параметры
- Они передаются как дополнительная часть URL
- В данном случае - это `?q=python`

Динамические ресурсы

- '?' отделяет путь от области параметров
- Область параметров - последовательность пар
- Первый элемент пары - ключ
- Второй элемент пары - значение
- Пары отделяются символом &
- Интерпретация ключей и значений - дело сервера

Параметры запроса

- Ключ и значение отделяются знаком =
- Пары отделяются символом &
- Если хочется спецсимвол сделать частью ключа или значения
- Нужно использовать '%' и ASCII-код
- Интерпретация ключей и значений - дело сервера

Параметры запроса в requests

- Можно сформировать строку запроса
- И включить в нее параметры
- Но это однотипная рутинная работа
- И еще надо заботиться о спецсимволах
- Есть вариант поудобнее

Используем именованный параметр `params`

```
1 import requests
2
3 response = requests.get('https://google.com',
4                         params={'q': 'Python'})
5 print(response.status_code)
```

Параметры запроса в requests

Параметр с несколькими значениями

```
1 import requests
2
3 response = requests.get('https://python.org',
4                         params={'key': ['value1', 'value2']}
5 print(response.url)
```

Смешанный вариант

```
1 import requests
2
3 response = requests.get('https://some-server.com',
4                         params={'key1': ['value1', 'value2']
5                                 'key2': 'value'})
6 print(response.url)
```

HTML: Hyper-Text Markup Language

- Язык разметки
- Текст перемежается тегами
- Часто хочется очистить текст от тегов
- Или что-то взять из тегов
- Например, ссылки на другие ресурсы

HTML: Hyper-Text Markup Language

- Хочется отделять полезное содержимое
- От баннеров
- От рекламных вставок
- От элементов навигации

```
<span id="library-index"></span><h
<p>While <a class="reference inter
semantics of the Python language,
describes the standard library tha
describes some of the optional com
in Python distributions.</p>
<p>Python’s standard library is ve
facilities as indicated by the lon
library contains built-in modules
system functionality such as file
```

Выглядит не очень сложно

- Можно написать код, который
  - Найдет все теги
  - Вычленит полезный текст без тегов
  - Извлечет знание о том, где начинаются и заканчиваются абзацы
  - Найдет теги со ссылками

Все сложнее, чем может показаться

- Теги бывают вложенные
- Структура тегов бывает нарушенной
- Переводы строк могут идти в произвольном порядке
- Теговые символы могут встречаться в тексте
  - И для этого есть отдельные конструкции

Beautiful Soup

- Есть специальная библиотека для обработки HTML
- Анализирует HTML-текст
- Порождает структуру вложенных элементов
- Дает возможеность удобного поиска

Beautiful Soup

- https://beautiful-soup-4.readthedocs.io/en/latest/"
- https://pypi.org/project/beautifulsoup4
- https://www.crummy.com/software/BeautifulSoup/
- Установка:

```
python3 -m pip install beautifulsoup4
```

Простейший пример

```
1 from bs4 import BeautifulSoup
2 import requests
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 print(soup.prettify())
```

Анализ содержимого

- Но чаще нужна не печать документа
- А анализ его содержимого
- Есть два подхода
- Явный перебор элементов(навигация) и поиск

Навигация

- HTML-документ состоит из тегов
- Внутри тегов могут быть вложенные теги
- На верхнем уровне - возможны два тега
- head - заголовок (метаданные)
- body - тело (основное содержимое)

Напечатаем заголовок

```
1 from bs4 import BeautifulSoup
2 import requests
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 print(soup.head)
```

Напечатаем тело

```
1 from bs4 import BeautifulSoup
2 import requests
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 print(soup.body)
```

Напечатаем абзац текста

```
1 from bs4 import BeautifulSoup
2 import requests
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 print(soup.p)
```

Возникают вопросы

- У нас же нет тега p на верхнем уровне
  - p в BeautifulSoup реализован как свойство класса
  - Его get-метод реализован так, что он сначала смотрит тег на текущем уровне
  - А если не находит, идет на более низкие уровни

Возникают вопросы

- Но на более низких уровнях у нас много тегов p
  - Возвращается первый найденный
  - А если нет ни на каком уровне, то возвращается None

А если нужны все подэлементы

- Или совсем все
- Или все данного типа
- Есть свойство children (прямые дети)
- И есть свойство descendants (потомки)

Пройдемся по детям

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 for e in soup.body.children:
7     print(e.name)
```

Проанализируем

- Дети чередуются
- У одних имя - div
- Что соответствует html-тегу div
- У других имя - None

Посмотрим на типы

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 response = requests.get('https://docs.python.org/3/library/i
5 soup = BeautifulSoup(response.text, 'html.parser')
6 for e in soup.body.children:
7     print(e.name, type(e))
```

Проанализируем

- У половины тип - Tag
- Что соответствует html-тегу div
- У половины - NavigableString
- У кого имя- None, они и есть NavigableString

NavigableString

- Внутри тега могут быть другие теги
- А может быть и содержимое
- Например, в теге p или внутри div
- И здесь оно тоже есть
- Просто оно ничем не заполнено