Курс «Базовая обработка данных на языке Python»

автор: Киреев В.С., к.т.н., доцент

Лабораторная работа № 6

Тема: «Скрапинг и парсинг в Python. Использование библиотек requests, bs4»

Цель работы: изучить основы скрапинга и парсинга в Python.

Теоретическая справка

Библиотека Requests в Python является одной из неотъемлемых частей Python для выполнения HTTP-запросов к указанному URL. Будь то REST API или веб-скрапинг, запросы необходимо изучить для дальнейшего использования этих технологий. Когда кто-то делает запрос к URI, он возвращает ответ. Запросы Python предоставляют встроенные функции для управления как запросом, так и ответом.

Методы библиотеки requests

Модуль запросов Python имеет несколько встроенных методов для выполнения HTTP-запросов к указанному URI с использованием запросов GET, POST, PUT, PATCH или HEAD. HTTP-запрос предназначен либо для извлечения данных из указанного URI, либо для отправки данных на сервер. Он работает как протокол запрос-ответ между клиентом и сервером. Давайте покажем, как выполнить GET-запрос к конечной точке. Метод GET используется для извлечения информации с указанного сервера с использованием указанного URI. Метод GET отправляет закодированную информацию о пользователе, прикрепленную к запросу страницы. Страница и закодированная информация разделяются символом «?». Например:

```
import requests
response = requests.get('https://api.github.com/')
print(response.url)
print(response.status_code)
```

Методы ответа на запрос к сайту

Метод	Описание
response.headers	response.headers возвращает словарь заголовков ответа.

response.encoding	response.encoding возвращает кодировку, используемую для декодирования response.content.
response.elapsed	response.elapsed возвращает объект timedelta со временем, прошедшим с момента отправки запроса до получения ответа.
response.close()	response.close() закрывает соединение с сервером.
response.content	response.content возвращает содержимое ответа в байтах.
response.cookies	response.cookies возвращает объект CookieJar с файлами cookie, отправленными с сервера.
response.history	response.history возвращает список объектов ответа, содержащих историю запроса (url).
response.is_permanent_redirect	response.is_permanent_redirect возвращает True, если ответ является URL-адресом постоянного перенаправления, в противном случае — False.
response.is_redirect	response.is_redirect возвращает True, если ответ был перенаправлен, в противном случае — False.
response.iter_content()	response.iter_content() выполняет итерацию по response.content.
response.json()	response.json() возвращает объект JSON результата (если результат был записан в формате JSON, в противном случае возникает ошибка).
response.url	response.url возвращает URL ответа.
response.text	response.text возвращает содержимое ответа в юникоде.
response.status_code	response.status_code возвращает число, указывающее статус (200 — OK, 404 — Не найдено).
response.request	response.request возвращает объект запроса, запросивший этот ответ.

response.reason	response.reason возвращает текст, соответствующий коду статуса.
response.raise_for_status()	response.raise_for_status() возвращает объект HTTPError, если во время процесса произошла ошибка.
response.ok	response.ok возвращает True, если status_code меньше 200, в противном случае — False.
response.links	response.links возвращает ссылки заголовка.

Аутентификация с использованием Python Requests

Аутентификация означает предоставление пользователю разрешений на доступ к определенному ресурсу. Поскольку всем не может быть разрешен доступ к данным с каждого URL, в первую очередь потребуется аутентификация. Для достижения этой аутентификации обычно предоставляются данные аутентификации через заголовок Authorization или пользовательский заголовок, определенный сервером.

```
import requests
from requests.auth import HTTPBasicAuth
response = requests.get('https://api.github.com / user, ',
            auth = HTTPBasicAuth('user', 'pass'))
print(response)
```

Библиотека bs4

В техническом плане веб-скриптинг — это автоматический метод получения больших объемов данных с веб-сайтов. Большая часть этих данных представляет собой неструктурированные данные в формате HTML, которые затем преобразуются в структурированные данные в электронной таблице или базе данных, чтобы их можно было использовать в различных приложениях. Beautiful Soup(bs4) — это библиотека Python для извлечения данных из файлов HTML и XML.

Объект BeautifulSoup

Объект BeautifulSoup представляет собой проанализированный документ в целом. Таким образом, это полный документ, который мы пытаемся вытащить. Для большинства целей вы можете рассматривать его как объект Tag.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup('''<h1>Geeks for Geeks</h1>''',
                     "html.parser")
print(type(soup))
```

Анализ HTML

Библиотека BeautifulSoup создана на основе библиотек анализа HTML, таких как html5lib, lxml, html.parser и т. д. Поэтому объект BeautifulSoup и указанная библиотека анализатора могут быть созданы одновременно.

```
import requests
from bs4 import BeautifulSoup
r = requests.get('https://www.geeksforgeeks.org/python-programming-
language/')
soup = BeautifulSoup(r.content, 'html.parser')
print(soup.prettify())
```

Самостоятельное задание

1. В разделе https://www.chipdip.ru/catalog-show/glass-fuses скачайте данные по первым 20 страницам, используя библиотеки requests и BeautifulSoup.
2. Распарсите полученные страницы в датафрейм **df** вида (используйте регулярные выражения):

Бренд	**Название**	**Артикул**	**ТОК**	**Номинальное напряжение**	**Размеры корпуса**	**Тип**	**Цена**	**Кол-во**	**Срок поставки**

3. Выделите из названий предохранителей тип – (быстродействующий, медленного срабатывания и обычный) и добавьте соответствующую новую колонку в датафрейм df.
4. Постройте группировку по переменной тип, в качестве агрегации используя - среднюю цену, максимальное количество на складе, минимальное число артикулов в группе.
5. Проведите анализ качества предохранителей:
   5.1. выявите самый лучший предохранитель внутри каждой категории, по техническим характеристикам.
   5.2. Определите индекс качества каждого предохранителя относительно лучшего в категории – разделите значения характеристик на лучшие, и суммируйте полученные числа.
   5.3. Постройте три линейных графика – зависимость цены от индекса качества.
6. Определите ценовой сегмент для каждого предохранителя. Используйте полученные ранее индексы качества.
7. Определите самый дефицитный предохранитель – с самым долгим сроком поставки, и минимальным числом в наличии на складе.
8. Проверьте гипотезу о связи тока и номинального напряжения в предохранителях:
   8.1. Разбейте ток и номинальное напряжение по две группы соответственно
   8.2. Постройте таблицу сопряженности – группированный ток vs группированное напряжение, с частотой предохранителей в ячейках.
   8.3. Проведите по построенной таблице тест хи-квадрат (scipy.stats.chi_contingency)
9. Визуализируйте с помощью тепловой карты зависимость цены от группированного тока и группированного напряжения.
10. Постройте линейную регрессию и определите наиболее значимые характеристики товара, влияющие на рост цены.