Лабораторная работа 2. Парсинг HTML. XPath+Selenium

Цель работы: научиться извлекать данные из HTML-страниц с использованием XPath. Практически применить навыки парсинга для сбора данных с веб-страниц.

Оборудование и ПО:

- Компьютер с доступом в интернет.
- Интерпретатор Python.
- Библиотеки: requests, lxml, BeautifulSoup.
- **Теоретическая часть**

XPath (XML Path Language) — это язык запросов, который используется для выбора узлов из XML-документов. Он также используется для навигации по HTML-страницам.

Пример XPath

Тема: Анализ данных о котировках акций с сайта Yahoo Finance

URL: https://finance.yahoo.com/most-active

Шаги выполнения

1. Импорт необходимых библиотек:

```
import requests
from lxml import html
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Отправка GET-запроса к целевому URL:

```
url = 'https://finance.yahoo.com/most-active'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
response = requests.get(url, headers=headers)
if response.status_code == 200:
  page_content = response.content
else:
  print(f"Ошибка при получении страницы: {response.status_code}")
  exit()
```

3. Парсинг HTML-страницы с использованием lxml:

```
tree = html.fromstring(page_content)
tree
```

4. Использование XPath для извлечения данных.

```
# Пример XPath для извлечения тикеров акций
symbols = tree.xpath('//td[@aria-label="Symbol"]/a/text()')
# Пример XPath для извлечения названий компаний
names = tree.xpath('//td[@aria-label="Name"]/text()')
# Пример XPath для извлечения текущих цен акций
prices = tree.xpath('//td[@aria-label="Price (Intraday)"]/fin-streamer/@value')
# Пример XPath для извлечения изменения цен акций
changes = tree.xpath('//td[@aria-label="Change"]/fin-streamer/@value')
# Пример XPath для извлечения процента изменения цен акций
percent_changes = tree.xpath('//td[@aria-label="% Change"]/fin-streamer/@value')
volumes = tree.xpath('//td[@aria-label="Volume"]/fin-streamer/@value')
```

5. Создание DataFrame.

```
data = {
  'Symbol': symbols,
```

```
    'Name': names,
    'Price': prices,
    'Change': changes,
    'Percent Change': percent_changes,
    'Volume': volumes
}
df = pd.DataFrame(data)
df
```

	Symbol	Name	Price	Change	Percent Change	Volume
0	NVDA	NVIDIA Corporation	118.08	1.9400024	1.6703999	335114546
1	NU	Nu Holdings Ltd.	13.38	0.67	5.27144	69369327
2	TSLA	Tesla, Inc.	201.38	-6.449997	-3.1034966	69508741
3	LUMN	Lumen Technologies, Inc.	5.66	0.65	12.974	55355245
4	INTC	Intel Corporation	19.92	-0.54999924	-2.6868553	56031238
5	PLTR	Palantir Technologies Inc.	31	0.610001	2.00724	49504222

6. Предобработка данных. Преобразуем строковые значения в числовые для возможности проведения математических операций.

```
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
df['Change'] = pd.to_numeric(df['Change'], errors='coerce')
df['Percent Change'] = pd.to_numeric(df['Percent Change'], errors='coerce')
df['Volume'] = pd.to_numeric(df['Volume'], errors='coerce')
```

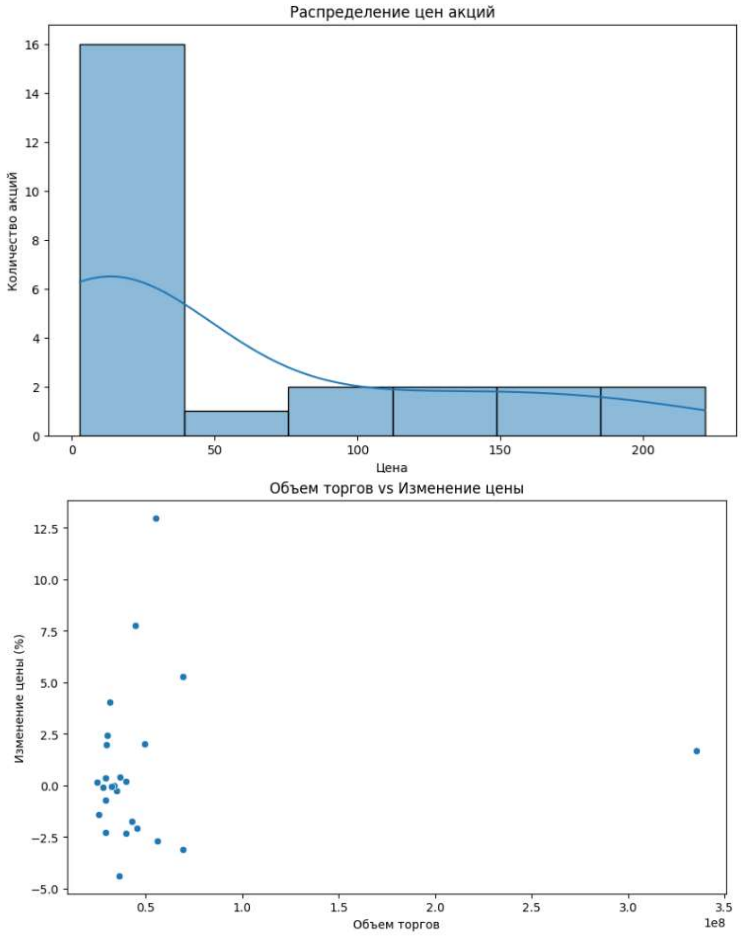
7. Анализ данных.

```
print(df.describe())
print("\nТоп 5 акций по объему торгов:")
print(df.nlargest(5, 'Volume')[['Symbol', 'Name', 'Volume']])
print("\nАкция с наибольшим ростом:")
print(df.loc[df['Percent Change'].idxmax()][['Symbol', 'Name', 'Percent Change']])
print("\nАкция с наибольшим падением:")
print(df.loc[df['Percent Change'].idxmin()][['Symbol', 'Name', 'Percent Change']])
```

```
            Price     Change  Percent Change        Volume
count   25.000000  25.000000       25.000000  2.500000e+01
mean    56.320800  -0.138400        0.725083  5.121253e+07
std     70.542271   2.030154        3.722875  6.044089e+07
min      2.760000  -6.449997       -4.370180  2.490159e+07
25%      6.960000  -0.180000       -1.740820  2.956220e+07
50%     15.140000   0.000000        0.000000  3.656066e+07
75%     93.900000   0.340000        1.954403  4.568690e+07
max    221.720000   5.780000       12.974000  3.351145e+08

Топ 5 акций по объему торгов:
  Symbol                     Name     Volume
0   NVDA        NVIDIA Corporation  335114546
2   TSLA                Tesla, Inc.   69508741
1     NU          Nu Holdings Ltd.   69369327
4   INTC         Intel Corporation   56031238
3   LUMN  Lumen Technologies, Inc.   55355245

Акция с наибольшим ростом:
Symbol                                LUMN
Name              Lumen Technologies, Inc.
Percent Change                      12.974
Name: 3, dtype: object

Акция с наибольшим падением:
Symbol                 NIO
Name              NIO Inc.
Percent Change    -4.37018
Name: 11, dtype: object
```

8. Визуализация данных.

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], kde=True)
plt.title('Распределение цен акций')
plt.xlabel('Цена')
plt.ylabel('Количество акций')
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Volume', y='Percent Change', data=df)
plt.title('Объем торгов vs Изменение цены')
plt.xlabel('Объем торгов')
plt.ylabel('Изменение цены (%)')
plt.show()
```



9. Сохранение DataFrame в CSV

```
df.to_csv('yahoo_finance_data.csv', index=False)
# скачать файл
from google.colab import files
files.download('yahoo_finance_data.csv')
```

Выводы.

В ходе выполнения лабораторной работы научился извлекать данные из HTML-страниц с использованием XPath, обрабатывать и анализировать собранные данные.

Пример SELENIUM

Извлечь данные через BeautifulSoup таблицы на сайте https://www.msci.com/indexes

Задача выполняется в виртуальной машине **selenium_dba_bmstu (логин: dba, пароль: 1)** **https://disk.yandex.ru/d/HYCe2JLCojbSuA**

Проверка наличия таблицы на сайте, а также анализ класса, который описывает искомую таблицу представлен на рис.1.

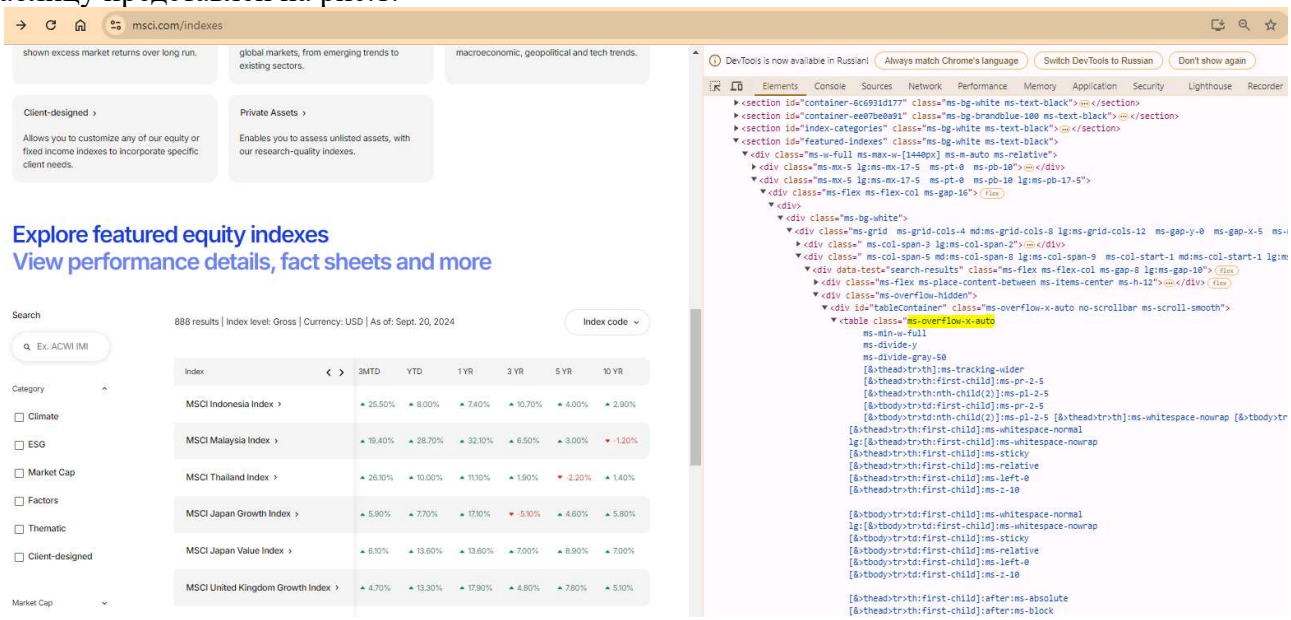


Рис. 1 -Анализ сайта **https://www.msci.com/indexes**

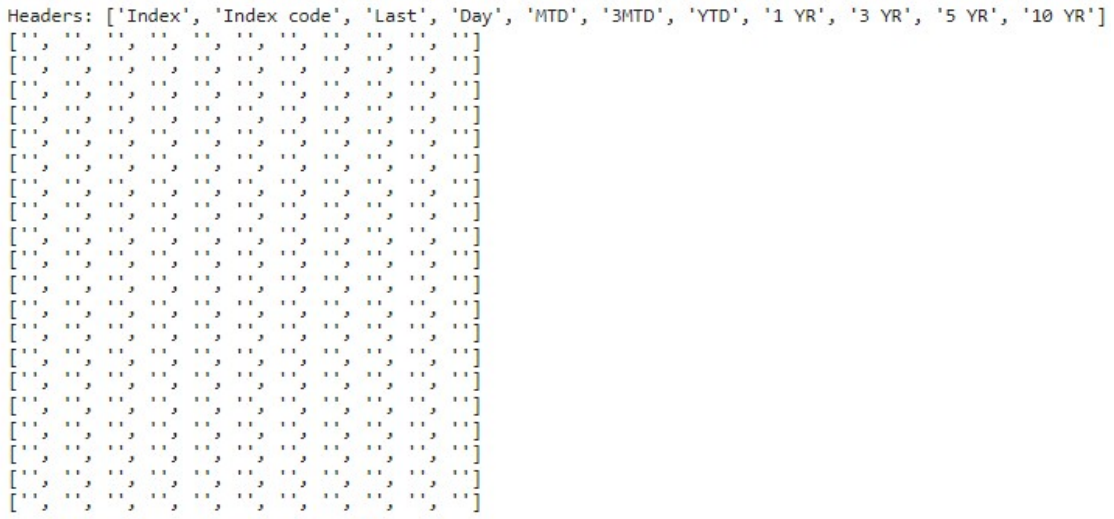
Класс, который отвечает за формирование таблицы - ms-overflow-x-auto.

Решим задачу парсинга стандартным методом с использованием библиотеки **BeautifulSoup.**

```
import requests
from bs4 import BeautifulSoup
# URL страницы с таблицей
url = 'https://www.msci.com/indexes'
# Получаем HTML страницы
response = requests.get(url)
# Проверяем статус ответа
if response.status_code == 200:
    html_content = response.text
    soup = BeautifulSoup(html_content, 'html.parser')
    # Находим таблицу с классом "ms-overflow-x-auto"
    table = soup.find('table', class_='ms-overflow-x-auto')
    if table:
        # Извлекаем заголовки таблицы
        headers = [th.text.strip() for th in table.find_all('th')]
        # Извлекаем строки таблицы
        rows = []
        for tr in table.find_all('tr'):
            cells = [td.text.strip() for td in tr.find_all('td')]
            if cells:
```

```
                rows.append(cells)
        # Печатаем заголовки и строки
        print("Headers:", headers)
        for row in rows:
            print(row)
    else:
        print("Таблица не найдена.")
else:
    print(f"Не удалось получить доступ к странице. Статус код:
{response.status_code}")
```

В результате получим таблицу вида:



Наличие заголовков говорит о том, что таблица определена правильно, однако данные на загружены.

Проблема с получением данных может быть связана с тем, что таблица на сайте динамически загружается через JavaScript после загрузки основной страницы, а при прямом запросе HTML через requests данные могут не быть частью HTML.

Поскольку requests не может работать с JavaScript-контентом, в таких случаях нужно использовать **Selenium** для имитации работы браузера и ожидания загрузки контента.

Шаг 1. Установка Selenium

Установить браузерный драйвер, например, **ChromeDriver** для Chrome:

- Скачайте его здесь, выберите версию, которая соответствует вашему браузеру.
- Установите ChromeDriver в папку, к которой имеет доступ ваш скрипт.

``**bash**
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt-get install -y ./google-chrome-stable_current_amd64.deb
sudo apt-get -y install python3-pip
pip3 install selenium
pip3 install webdriver-manager

Создать тестовый файл для проверки работоспособности Selenium

``**nano**

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
```

options = Options()
# options.add_argument('--headless')
# options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), opt>

driver.get("https://python.org")
print(driver.title)
driver.close()

``**bash**
python3 test.py

После работы Selenium в терминале получим парс-текст **Welcome to Python.org**
Определим путь драйвера **webdriver.Chrome.**
``**bash**
find / -name "chromedriver"

В поиске находим путь для виртуальной машины dba :
/home/dba/.wdm/drivers/chromedriver/linux64/129.0.6668.58/chromedriver-linux64/chromedriver

Обновленный код с использованием Selenium в **sel.py**.
``**python**

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
import pandas as pd
import time
# Укажите путь к chromedriver
chrome_driver_path = "/home/dba/.wdm/drivers/chromedriver/linux64/129.0.6668.58/chromedriver-linux64/chromedriver"
# Шаг 1: Настройка драйвера Selenium
service = Service(chrome_driver_path)
driver = webdriver.Chrome(service=service)
# Шаг 2: Открытие страницы
url = "https://www.msci.com/indexes"
driver.get(url)
# Шаг 3: Ждем несколько секунд для загрузки страницы и данных через JavaScript
time.sleep(5)
# Шаг 4: Поиск таблицы с классом 'ms-overflow-x-auto'
table = driver.find_element(By.CLASS_NAME, 'ms-overflow-x-auto')
# Шаг 5: Извлечение заголовков таблицы (thead)
headers = []
thead = table.find_element(By.TAG_NAME, 'thead')
for th in thead.find_elements(By.TAG_NAME, 'th'):
    headers.append(th.text.strip())
# Шаг 6: Извлечение строк данных (tbody)
```

```
rows = []
tbody = table.find_element(By.TAG_NAME, 'tbody')
for tr in tbody.find_elements(By.TAG_NAME, 'tr'):
    cells = tr.find_elements(By.TAG_NAME, 'td')
    row = [cell.text.strip() for cell in cells]
    rows.append(row)

# Шаг 7: Создание DataFrame
df = pd.DataFrame(rows, columns=headers)
# Шаг 8: Вывод первых нескольких строк DataFrame
print(df)
# Закрытие драйвера
driver.quit()
```

Описание изменений:

1. **Selenium** управляет браузером и открывает страницу так, как это делает человек.
2. Используется метод `time.sleep(5)`, чтобы дать время на подгрузку таблицы с сайта.
3. **find_elements** находит все строки и ячейки в таблице.

Результат работы скрипта:

```
dba@dba-vm:~/Downloads$ python3 sel.py
                          Index Index code      Last     Day     MTD  ...      YTD    1 YR    3 YR    5 YR   10 YR
0                MSCI Indonesia Index     105767   2398.81  -0.60%   4.90%  ...    8.00%   7.40%  10.70%   4.00%   2.90%
1                 MSCI Malaysia Index     105768    861.78   0.30%   3.60%  ...   28.70%  32.10%   6.50%   3.00%  -1.20%
2                 MSCI Thailand Index     105769   1123.28  -0.26%   8.60%  ...   10.00%  11.10%   1.90%  -2.20%   1.40%
3             MSCI Japan Growth Index     105795   1537.95   0.30%  -2.50%  ...    7.70%  17.10%  -5.10%   4.60%   5.80%
4              MSCI Japan Value Index     105796  13706.10  -0.18%  -2.20%  ...   13.60%  13.60%   7.00%   8.90%   7.00%
5     MSCI United Kingdom Growth Index     105823  17331.72  -1.03%  -2.30%  ...   13.30%  17.90%   4.80%   7.80%   5.10%
6      MSCI United Kingdom Value Index     105824  28512.45  -0.89%   0.00   ...   14.70%  18.40%  12.80%   7.00%   2.60%
7               MSCI USA Growth Index     105825  32257.63  -0.25%   1.60%  ...   24.20%  38.00%  10.00%  19.70%  16.20%
8                MSCI USA Value Index     105826  25165.69  -0.14%   0.60%  ...   16.60%  24.70%   9.50%  10.50%   9.20%
9              MSCI EAFE Growth Index     105833   6226.30  -1.13%  -2.30%  ...    9.50%  19.20%   0.20%   7.30%   6.40%
10              MSCI EAFE Value Index     105834  17528.31  -0.48%  -0.20%  ...   12.70%  18.10%   9.40%   8.40%   4.70%
11           MSCI Europe Growth Index     105843  12439.42  -1.94%  -3.20%  ...    9.10%  18.70%   2.10%   8.90%   6.90%
12            MSCI Europe Value Index     105844  17345.96  -0.68%   0.10%  ...   13.30%  19.90%  10.80%   8.50%   4.20%
13            MSCI World Growth Index     105867  14095.09  -0.47%   0.60%  ...   19.80%  32.30%   7.10%  15.70%  12.60%
14             MSCI World Value Index     105868  21735.78  -0.21%   0.50%  ...   15.60%  22.90%   9.60%   9.90%   7.50%
15      MSCI World Ex USA Growth Index     105873   5975.46  -1.03%  -2.00%  ...    9.60%  19.10%   0.50%   7.40%   6.20%
16       MSCI World Ex USA Value Index     105874  17559.21  -0.41%   0.20%  ...   13.10%  18.90%   9.70%   8.70%   4.90%
17           MSCI Brazil Growth Index     105882    451.08  -2.79%  -0.80%  ...  -14.40%  -6.20%  -5.50%  -7.20%  -4.30%
18            MSCI Brazil Value Index     105883    807.22  -2.30%  -1.90%  ...  -11.80%   1.70%  20.70%   7.80%   4.60%
19  MSCI AC Asia Ex Japan Growth Index     105972    349.43   1.09%   2.30%  ...   15.00%  19.90%  -4.60%   4.90%   5.50%
```

Файл **sel.py** размещен в **Downloads**. При открытии **Code** сразу открывается проект с файлом.

При выполнении скрипта, будет отрыт браузер без участия человека и так же закрыт после окончания работы скрипта.

Варианты заданий

Предоставлен уникальный URL, с которого требуется извлечь определённую информацию. Ниже приведены примеры заданий.

1. Извлечь название статей и даты публикации с главной страницы https://www.economist.com/latest/
2. Собрать данные о текущих котировках акций с https://finance.yahoo.com/most-active
3. Получить информацию о последних новостях с сайта https://www.bloomberg.com/markets
4. Извлечь данные о валютных курсах с https://www.xe.com/currencyconverter/
5. Собрать информацию о предложениях по кредитам на https://www.bankrate.com/loans/personal-loans/
6. Получить список последних статей и их авторов на https://www.reuters.com/finance
7. Извлечь данные о последних изменениях в ставках по ипотеке на https://www.zillow.com/mortgage-rates/
8. Собрать информацию о текущих ценах на нефть с https://oilprice.com/
9. Получить данные о последних экономических новостях на https://www.ft.com/world
10. Извлечь информацию о последних изменениях в налоговом законодательстве на https://www.taxpolicycenter.org/
11. Собрать данные о последних изменениях индексов на https://www.marketwatch.com/tools/marketsummary
12. Получить информацию о последних публикациях на тему криптовалют на https://www.coindesk.com/
13. Извлечь данные о текущих ставках по депозитам на https://www.investopedia.com/best-bank-cd-rates-4846922
14. Собрать информацию о последних новостях на тему недвижимости на https://www.realtor.com/news/
15. Получить данные о текущих ценах на золото на https://www.kitco.com/
16. Извлечь информацию о последних изменениях в банковских ставках на https://www.bankofamerica.com/
17. Собрать данные о последних изменениях в кредитных ставках на https://www.creditkarma.com/
18. Получить информацию о последних публикациях на тему страхования на https://www.insurancejournal.com/
19. Извлечь данные о текущих ставках по кредитным картам на https://www.nerdwallet.com/best/credit-cards
20. Собрать информацию о последних изменениях в рейтингах облигаций на https://www.bondbuyer.com/
21. Получить данные о текущих ставках на кредитные линии на https://www.lendingtree.com/
22. Извлечь информацию о последних изменениях в ставках по автокредитам на https://www.autotrader.com/
23. Собрать данные о последних публикациях на тему финансового планирования на https://www.financial-planning.com/
24. Получить информацию о текущих ставках по ипотеке на https://www.nerdwallet.com/mortgages
25. Извлечь данные о последних изменениях в мировых индексах на https://www.investing.com/indices/major-indices