**ШАГ 1. ИМПОРТИРОВАНИЕ БИБЛИОТЕК**

После установки библиотек давайте откроем вашу любимую среду разработки. Мы предлагаем использовать Spyder 4.2.5. Позже на некоторых этапах работы мы столкнемся с большими объемами выводимых данных и тогда Spyder будет удобнее в использовании чем Jupyter Notebook.

Итак, Spyder открыт и мы можем импортировать необходимую библиотеку:

# Import library  
from bs4 import BeautifulSoup  
import requests

**ШАГ 2. ВЫБОР СТРАНИЦЫ**

В этом проекте мы будем использовать [webscraper.io](https://webscraper.io/). Поскольку данный веб-сайт создан на HTML, код легче и понятнее даже новичкам. Мы выбрали эту страницу для парсинга данных:

Webscrapper.io

Она является прототипом веб-сайта онлайн магазина. Мы будем парсить данные о компьютерах и ноутбуках, такие как название продукта, цена, описание и отзывы.

**ШАГ 3. ЗАПРОС НА РАЗРЕШЕНИЕ**

После выбора страницы мы копируем ее URL-адрес и используем **request**, чтобы запросить разрешение у сервера на получение данных с их сайта.

# Define URL  
url = ‘<https://webscraper.io/test-sites/e-commerce/allinone/computers/laptops'>#

Ask hosting server to fetch url  
requests.get(url)

Результат <Response [200]> означает, что сервер позволяет нам собирать данные с их веб-сайта. Для проверки мы можем использовать функцию request.get.

pages = requests.get(url)  
pages.text

Когда вы выполните этот код, то на выходе получите беспорядочный текст, который не подходит для Python. Нам нужно использовать парсер, чтобы сделать его более читабельным.

# parser-lxml = Change html to Python friendly format  
soup = BeautifulSoup(pages.text, ‘lxml’)  
soup

До и После использования Парсера

**ШАГ 4. ПРОСМОТР КОДА ЭЛЕМЕНТА**

Для парсинга сайтов на Python мы рекомендуем использовать Google Chrome, он очень удобен и прост в использовании. Давайте узнаем, как с помощью Chrome просмотреть код веб-страницы. Сначала нужно щелкнуть правой кнопкой мыши страницу, которую вы хотите проверить, далее нажать **Просмотреть код** и вы увидите это:

Просмотр Кода Элемента

Затем щелкните **Выбрать элемент на странице для проверки**и вы заметите, что при перемещении курсора к каждому элементу страницы, меню элементов показывает его код.

Пример

Например, если мы переместим курсор на **Test Sites**, элемент покажет, что **Test Sites** находится в теге **h1**. В Python, если вы хотите просмотреть код элементов сайта, можно вызывать теги. Характерной чертой тегов является то, что они всегда имеют < в качестве префикса и часто имеют фиолетовый цвет.

[Как выбрать решение для парсинга сайтов: классификация и большой обзор программ, сервисов и фреймворков](https://idatica.com/blog/parser-obzor-i-klassifikaci/)

**ШАГ 5. ДОСТУП К ТЕГАМ**

Если мы, к примеру, хотим получить доступ к элементу h1 с помощью Python, мы можем просто ввести:

# Access h1 tag  
soup.h1

Результат будет:

soup.h1  
Out[11]: <h1>Test Sites</h1>

Вы можете получить доступ не только к однострочным тегам, но и к тегам класса, например:

# Access header tag

soup.header#

Access div tag

soup.div

Не забудьте перед этим определить **soup**, поскольку важно преобразовать HTML в удобный для Python формат.

Вы можете получить доступ к определенному из вложенных тегов. **Вложенные теги означают теги внутри тегов.** Например, тег <p> находится внутри другого тега <header>. Но когда вы получаете доступ к определенному тегу из <**header**>, Python всегда покажет результаты из первого индекса. Позже мы узнаем, как получить доступ к нескольким тегам из вложенных.

# Access string from nested tags  
soup.header.p

Результат:

soup.header.p  
Out[10]: <p>Web Scraper</p>

Вы также можете получить доступ к строке вложенных тегов. Нужно просто добавить в код**string**.

# Access string from nested tags  
soup.header.p  
soup.header.p.string

Результат:

soup.header.p  
soup.header.p.string  
Out[12]: ‘Web Scraper’

Следующий этап парсинга сайтов на Python — это получение доступа к атрибутам тегов. Для этого мы можем использовать функциональную возможность BeautifulSoup**attrs**. Как результат применения **attrs** мы получим словарь.

# Access ‘a’ tag in <header>

a\_start = soup.header.a

a\_start#

Access only the attributes using attrs

a\_start.attrs

Результат:

Out[16]:  
{‘data-toggle’: ‘collapse-side’,  
‘data-target’: ‘.side-collapse’,  
‘data-target-2’: ‘.side-collapse-container’}

Мы можем получить доступ к определенному атрибуту. Учтите, что Python рассматривает атрибут как словарь, поэтому **data-toggle**, **data-target** и **data-target-2**являются **ключом**. Вот пример получение доступа к ‘**data-target**:

a\_start[‘data-target’]

Результат:

a\_start[‘data-target’]  
Out[17]: ‘.side-collapse’

Мы также можем добавить новый атрибут. Имейте в виду, что изменения влияют только на веб-сайт локально, а не на веб-сайт в мировом масштабе.

a\_start[‘new-attribute’] = ‘This is the new attribute’  
a\_start.attrs  
a\_start

Результат:

a\_start[‘new-attribute’] = ‘This is the new attribute’  
a\_start.attrs  
a\_start  
Out[18]:  
<a data-target=”.side-collapse” data-target-2=”.side-collapse-container” data-toggle=”collapse-side” new-attribute=”This is the new attribute”>  
<button aria-controls=”navbar” aria-expanded=”false” class=”navbar-toggle pull-right collapsed” data-target=”#navbar” data-target-2=”.side-collapse-container” data-target-3=”.side-collapse” data-toggle=”collapse” type=”button”>  
...  
</a>

[Парсинг таблицы с сайта на Python: Пошаговое руководство](https://idatica.com/blog/parsing-tablitsy-s-sayta-na-python-poshagovoye-rukovodstvo/)

**ШАГ 6. ДОСТУП К КОНКРЕТНЫМ АТРИБУТАМ ТЕГОВ**

Мы узнали, что в теге может быть больше чем один вложенный тег. Например, если мы запустим **soup.header.div**, <div> будет иметь много вложенных тегов. Учтите, что мы вызываем только <div> внутри <header >, поэтому другой тег внутри <header> не будет показан.

Результат:

soup.header.div  
Out[26]:  
<div class=”container”>  
<div class=”navbar-header”>  
<a data-target=”.side-collapse” data-target-2=”.side-collapse-container” data-toggle=”collapse-side” new-attribute=”This is the new attribute”>  
<button aria-controls=”navbar” aria-expanded=”false” class=”navbar-toggle pull-right collapsed” data-target=”#navbar” data-target-2=”.side-collapse-container” data-target-3=”.side-collapse” data-toggle=”collapse” type=”button”>  
...  
</div>

Как мы видим, в одном теге находится много атрибутов и вопрос заключается в том, как получить доступ только к тому атрибуту, который нам нужен. В BeautifulSoup есть функция **‘find’** и **‘find\_all’**. Чтобы было понятнее, мы покажем вам, как использовать обе функции и чем они отличаются друг от друга. В качестве примера найдем цену каждого товара. Чтобы увидеть код элемента цены, просто наведите курсор на индикатор цены.

После перемещения курсора мы можем определить, что цена находится в теге **h4**, значение класса **pull-right price.**

Индикатор Цены

Далее мы хотим найти строку элемента h4, используя функцию find:

# Searching specific attributes of tags  
soup.find(‘h4’, class\_= ‘pull-right price’)

Результат:

Out[28]: <h4 class=”pull-right price”>**$295.99**</h4>

Как видно, **$295,99** — это атрибут (строка) h4. Но что будет, если мы используем find\_all.

# Using find\_all  
soup.find\_all(‘h4’, class\_= ‘pull-right price’)

Результат:

Out[29]:  
[<h4 class=”pull-right price”>$295.99</h4>,  
<h4 class=”pull-right price”>$299.00</h4>,  
<h4 class=”pull-right price”>$299.00</h4>,  
<h4 class=”pull-right price”>$306.99</h4>,  
<h4 class=”pull-right price”>$321.94</h4>,  
<h4 class=”pull-right price”>$356.49</h4>,  
....  
</h4>]

Вы заметили разницу между **find** и **find\_all**?

Да, все верно, **find** нужно использовать для поиска определенных атрибутов, потому что он возвращает только один результат. Для парсинга больших объемов данных (например, цена, название продукта, описание и т. д.), используйте **find\_all**.

Кроме того, можем получить часть результата функции find\_all. В данном случае мы хотим видеть только индексы с 3-го до 5-го.

# Slicing the results of find\_all  
soup.find\_all(‘h4’, class\_= ‘pull-right price’)[2:5]

Результат:

Out[32]:  
[<h4 class=”pull-right price”>$299.00</h4>,  
<h4 class=”pull-right price”>$306.99</h4>,  
<h4 class=”pull-right price”>$321.94</h4>]

[!] Не забывайте, что в Python индекс первого элемента в списке — 0, а последний не учитывается.

**ШАГ 7. ИСПОЛЬЗОВАНИЕ ФИЛЬТРА**

При необходимости мы можем найти несколько тегов:

# Using filter to find multiple tags  
soup.find\_all(['h4', 'a', 'p'])  
soup.find\_all(['header', 'div'])  
soup.find\_all(id = True) # class and id are special attribute so it can be written like this  
soup.find\_all(class\_= True)

Поскольку class и id являются специальными атрибутами, поэтому можно писать class\_ и id вместо ‘class’ или ‘id’.

Использование фильтра поможет нам получить необходимые данные с веб-сайта. В нашем случае это название, цена, отзывы и описания. Итак, сначала определим переменные.

# Filter by name

name = soup.find\_all(‘a’, class\_=’title’)

# Filter by price

price = soup.find\_all(‘h4’, class\_ = ‘pull-right price’)

# Filter by reviews

reviews = soup.find\_all(‘p’, class\_ = ‘pull-right’)

# Filter by description

description = soup.find\_all(‘p’, class\_ =’description’)

Фильтр по названию:

[<a class=”title” href=”/test-sites/e-commerce/allinone/product/545" title=”Asus VivoBook X441NA-GA190">Asus VivoBook X4…</a>,  
<a class=”title” href=”/test-sites/e-commerce/allinone/product/546" title=”Prestigio SmartBook 133S Dark Grey”>Prestigio SmartB…</a>,  
<a class=”title” href=”/test-sites/e-commerce/allinone/product/547" title=”Prestigio SmartBook 133S Gold”>Prestigio SmartB…</a>,  
...  
</a>]

Фильтр по цене:

[<h4 class=”pull-right price”>$295.99</h4>,  
<h4 class=”pull-right price”>$299.00</h4>,  
<h4 class=”pull-right price”>$299.00</h4>,  
<h4 class=”pull-right price”>$306.99</h4>,  
...  
</h4>]

Фильтр по отзывам:

[<p class=”pull-right”>14 reviews</p>,  
<p class=”pull-right”>8 reviews</p>,  
<p class=”pull-right”>12 reviews</p>,  
<p class=”pull-right”>2 reviews</p>,  
...  
</p>]

Фильтр по описанию:

[<p class=”description”>Asus VivoBook X441NA-GA190 Chocolate Black, 14", Celeron N3450, 4GB, 128GB SSD, Endless OS, ENG kbd</p>,  
<p class=”description”>Prestigio SmartBook 133S Dark Grey, 13.3" FHD IPS, Celeron N3350 1.1GHz, 4GB, 32GB, Windows 10 Pro + Office 365 1 gadam</p>,  
<p class=”description”>Prestigio SmartBook 133S Gold, 13.3" FHD IPS, Celeron N3350 1.1GHz, 4GB, 32GB, Windows 10 Pro + Office 365 1 gadam</p>,  
...  
</p>]

**ШАГ 8. ОЧИСТКА ДАННЫХ**

Очевидно, результаты все еще в формате HTML, поэтому нам нужно очистить их и получить только строку элемента. Используем для этого функцию **text**.

**Text** может служить для сортировки строк HTML кода, однако нужно определить новую переменную, например:

# Try to call price  
price1 = soup.find(‘h4’, class\_ = ‘pull-right price’)  
price1.text

Результат:

Out[55]: ‘$295.99’

На выходе получается только строка из кода, но этого недостаточно. На следующем этапе мы узнаем, как парсить все строки и сделать из них список.

**ШАГ 9.  ИСПОЛЬЗОВАНИЕ ЦИКЛА FOR ДЛЯ СОЗДАНИЯ СПИСКА СТРОК**

Чтобы сделать список из всех строк, необходимо создать цикл for.

# Create for loop to make string from find\_all list

product\_name\_list = []

for i in name:

name = i.text

product\_name\_list.append(name)price\_list = []

for i in price:

price = i.text

price\_list.append(price)

review\_list = []

for i in reviews:

rev = i.text

review\_list.append(rev)

description\_list = []

for i in description:

desc = i.text

description\_list.append(desc)

**ШАГ 10.  СОЗДАНИЕ ФРЕЙМА ДАННЫХ**

После того, как мы создали цикл for и все строки были добавлены в списки, остается заключительный этап парсинга сайтов на Python — построить фрейм данных. Для этой цели нам нужно импортировать библиотеку pandas.

# Create dataframe

# Import library

import pandas as pdtabel = pd.DataFrame({‘Product Name’:product\_name\_list,

‘Price’: price\_list,

‘Reviews’:review\_list,

‘Description’:description\_list})

Теперь эти данные можно использовать для работы в проектах по анализу и обработке данных, в машинном обучении, для получения другой ценной информации.

Фрейм данных