Parsing full code в файле **full\_code.py**

1. Начинаем с импорта необходимых нам модулей

установка Pandas - <https://pypi.org/project/pandas/#installation-from-sources>

import requests

from bs4 import BeautifulSoup

import pandas as pd

import csv

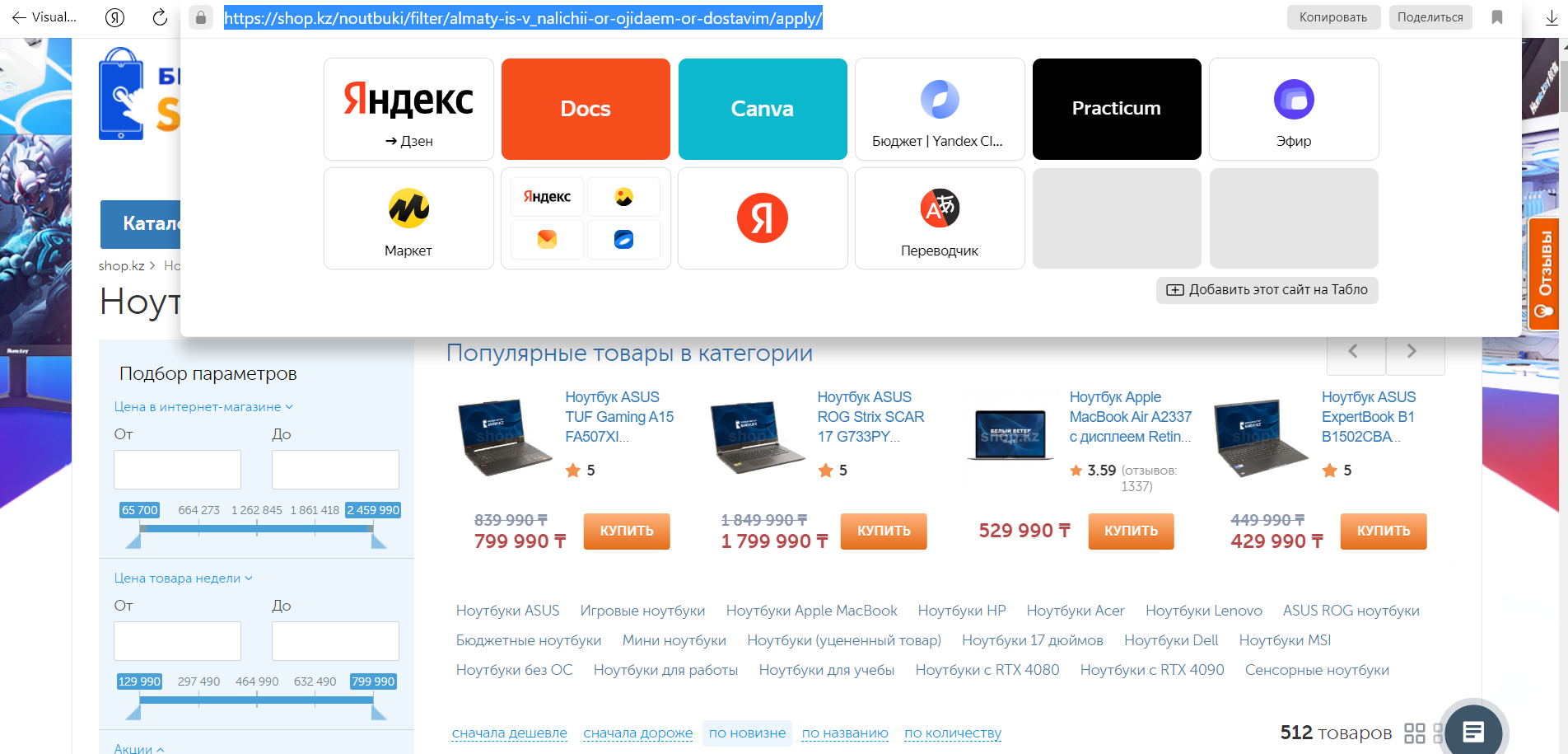
1. Определен пользовательский агент в переменной **headers**, который помогает в имитации запросов от браузера.

headers = {

    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36'

}

1. Создает базовый URL base\_url для страницы, которая содержит список ноутбуков на сайте shop.kz



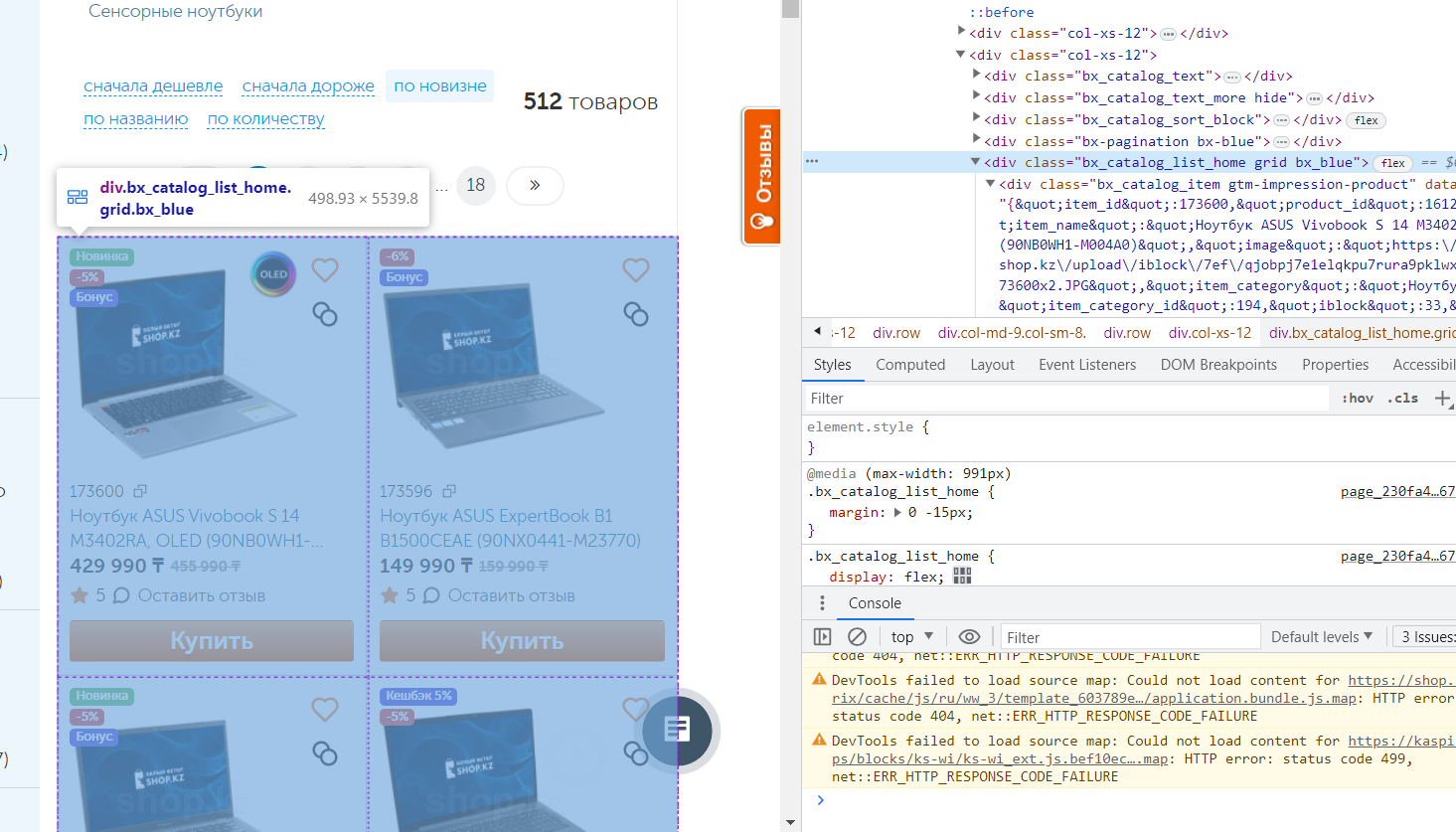
Берем ссылку именно страницы где **каталог** ноутбуков

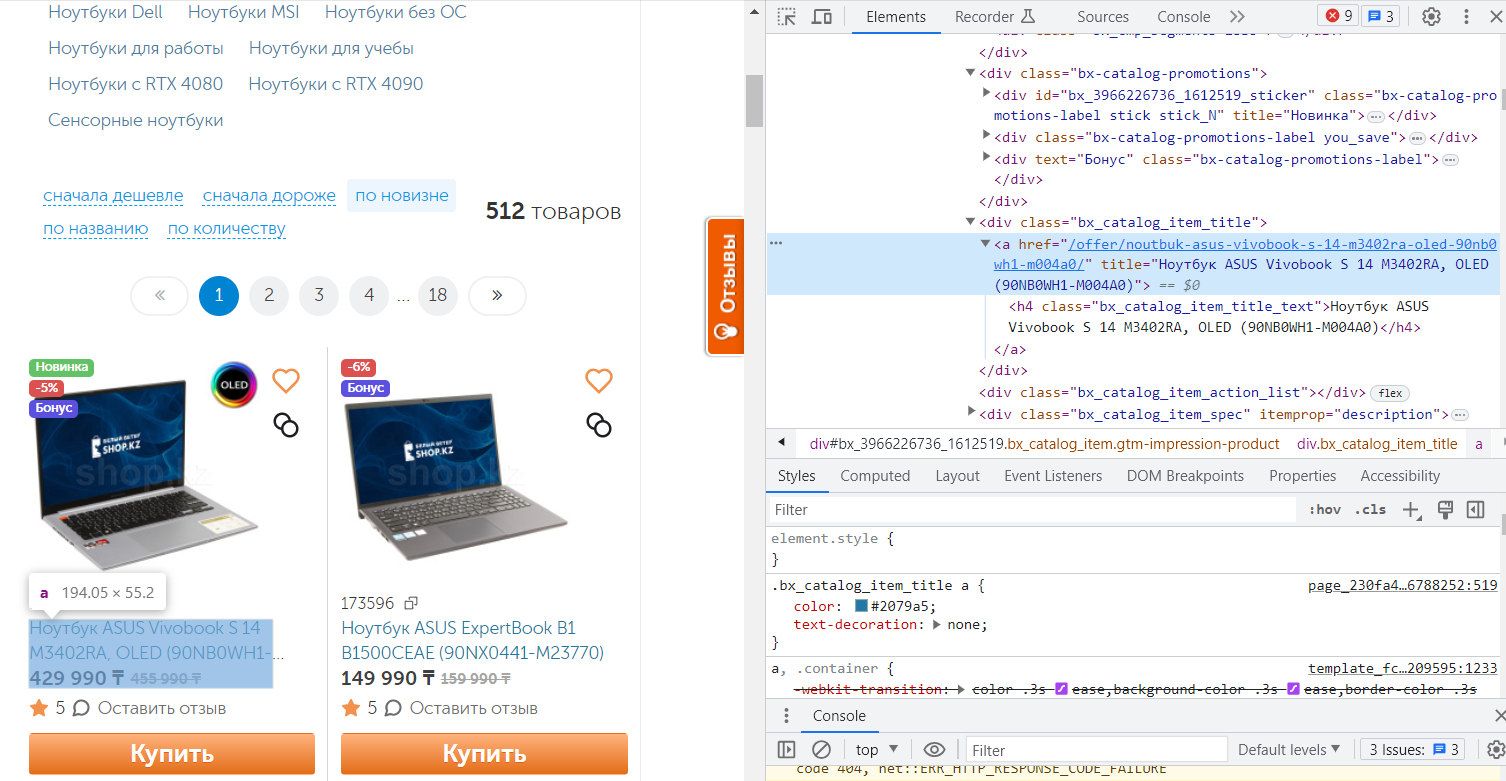
base\_url = 'https://shop.kz/noutbuki/filter/almaty-is-v\_nalichii-or-ojidaem-or-dostavim/apply/'

1. Выполняет GET-запрос к базовому URL с использованием **requests**

response\_all = requests.get(base\_url, headers=headers)

1. Находит список ноутбуков в каталоге с помощью класса bx\_catalog\_list\_home. Извлекаем значения атрибута 'href' для каждого ноутбука в списке и сохраняем их в href\_values.





Создаем пустой DataFrame all\_data.

Затем в цикле перебираем значения href и для каждого значения создает полный URL-адрес продукта.

if response\_all.status\_code == 200:

    soup = BeautifulSoup(response\_all.text, 'html.parser')

    catalog\_list = soup.find('div', class\_='bx\_catalog\_list\_home')

    href\_values = []

    if catalog\_list:

        titles = catalog\_list.find\_all('div', class\_='bx\_catalog\_item\_title')

        for title in titles:

            a\_tag = title.find('a')

            if a\_tag and 'href' in a\_tag.attrs:

                href\_values.append(a\_tag['href'])

    all\_data = pd.DataFrame()

1. Получаем данные о каждом ноутбуке с помощью функции getFromLink. Если данные успешно извлечены, они выводятся на экран, а затем добавляются в all\_data. После прохождения по всем ноутбукам, собранные данные сохраняются в файл CSV с помощью модуля csv.

for href in href\_values:

        # Construct the complete URL using the base\_url and href

        product\_url = "https://shop.kz" + href

        scraped\_data = getFromLink(product\_url)

        if scraped\_data is not None:

            print(scraped\_data)

            all\_data = pd.concat([all\_data, scraped\_data], ignore\_index=True)

    # Save all scraped data to an XLSX file

    with open('laptop\_data.csv', 'w', newline='', encoding='utf-8-sig') as file:

        writer = csv.writer(file)

        writer.writerow(all\_data.columns)  # запись заголовков

        for row in all\_data.values:

            writer.writerow(row)

1. Функция getFromLink(link) выполняет следующие действия:

Получает содержимое веб-страницы по заданной ссылке link с использованием модуля **requests** и заголовков **headers**.

Если код состояния ответа равен 200, то парсит HTML-код с помощью BeautifulSoup.

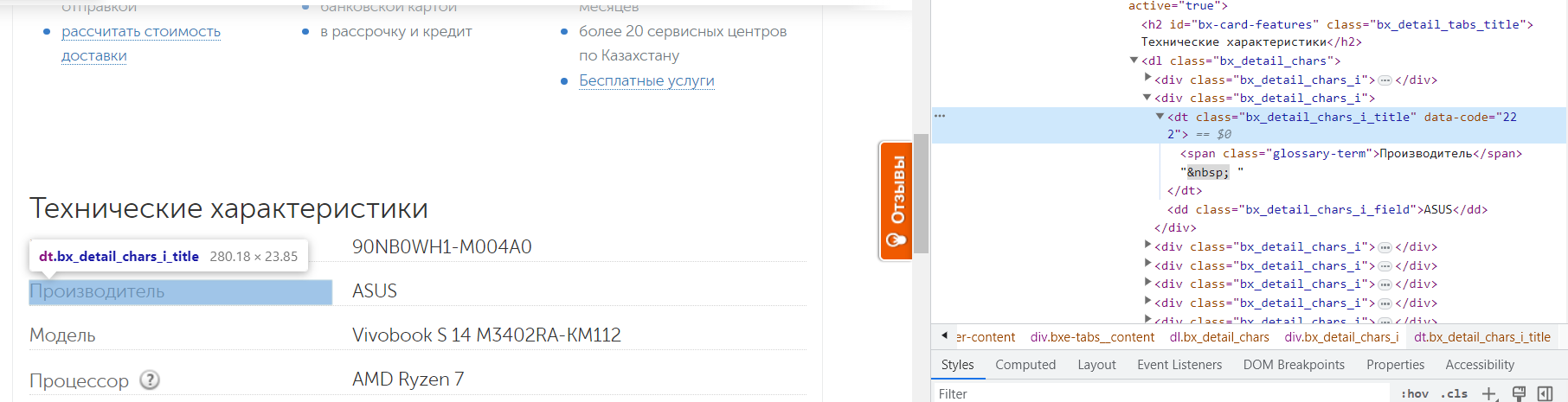
Инициализирует список laptop\_data и переменные для хранения информации о ноутбуке.

Извлекает информацию о ноутбуке, такую как название, цена, производитель, процессор, объем оперативной памяти (RAM), твердотельный накопитель (SSD), жесткий диск (HDD), вес и дисплей.

Проверяет, есть ли название и цена ноутбука, и если есть, добавляет полученные данные в список laptop\_data в виде словаря.

Преобразует список laptop\_data в DataFrame с использованием библиотеки pandas.

Если запрос не удался (код состояния ответа не равен 200), выводит сообщение об ошибке с кодом состояния и возвращает None



def getFromLink(link):

    response = requests.get(link, headers=headers)

    if response.status\_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')

        laptop\_data = []

        brand = "N/A"  # Set a default value

        processor="N/A"

        ram="N/A"

        ssd="N/A"

        hdd="N/A"

        weight="N/A"

        display="N/A"

        # Extract laptop information (modify as needed)s

        name = soup.find('h1', class\_='bx-title').text

        price = soup.find('div', class\_='item\_current\_price').text

        bx\_detail = soup.find\_all('dt', class\_='bx\_detail\_chars\_i\_title')

        if bx\_detail:

            details = soup.find\_all('span', class\_='glossary-term')

            for detail in details:

                glossary\_term = detail.text.strip().lower()

                if glossary\_term == 'производитель':

                    brand = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field').text

                if glossary\_term == 'процессор':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    processor = next\_dd.text if next\_dd else "N/A"

                if glossary\_term == 'объем оперативной памяти, гб':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    ram = next\_dd.text if next\_dd else "N/A"

                if glossary\_term == 'твердотельный накопитель':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    ssd = next\_dd.text if next\_dd else "N/A"

                if glossary\_term == 'жесткий диск':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    hdd = next\_dd.text if next\_dd else "N/A"

                if glossary\_term == 'вес, кг':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    weight = next\_dd.text if next\_dd else "N/A"

                if glossary\_term == 'вес, кг':

                    next\_dd = detail.find\_next('dd', class\_='bx\_detail\_chars\_i\_field')

                    display = next\_dd.text if next\_dd else "N/A"

        if name and price:

            laptop\_data.append({

                'Name': name,

                'Price': price,

                'Brand': brand,

                'Processor':processor,

                'Ram':ram,

                'SSD':ssd,

                'HDD':hdd,

                'Weight':weight,

                'Display':display,

            })

        laptop\_df = pd.DataFrame(laptop\_data)

        return laptop\_df

    else:

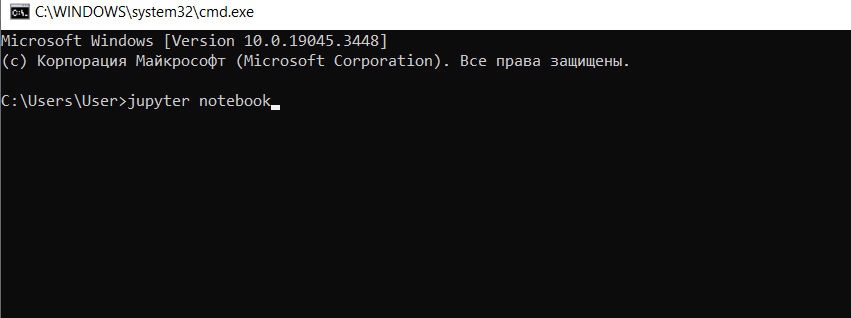
        print(f'Failed to retrieve the webpage. Status code: {response.status\_code}')

        return None

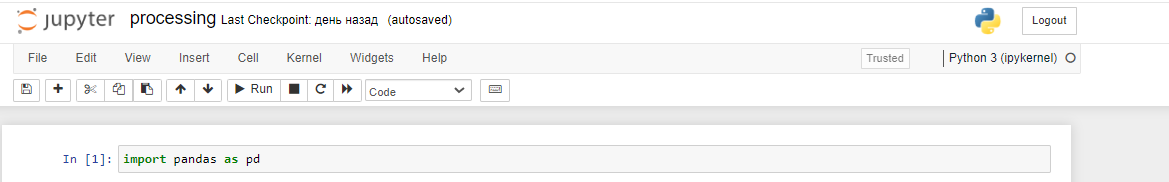
Processing

Установите jupyter <https://jupyter.org/install>

После установки открываем юпитер

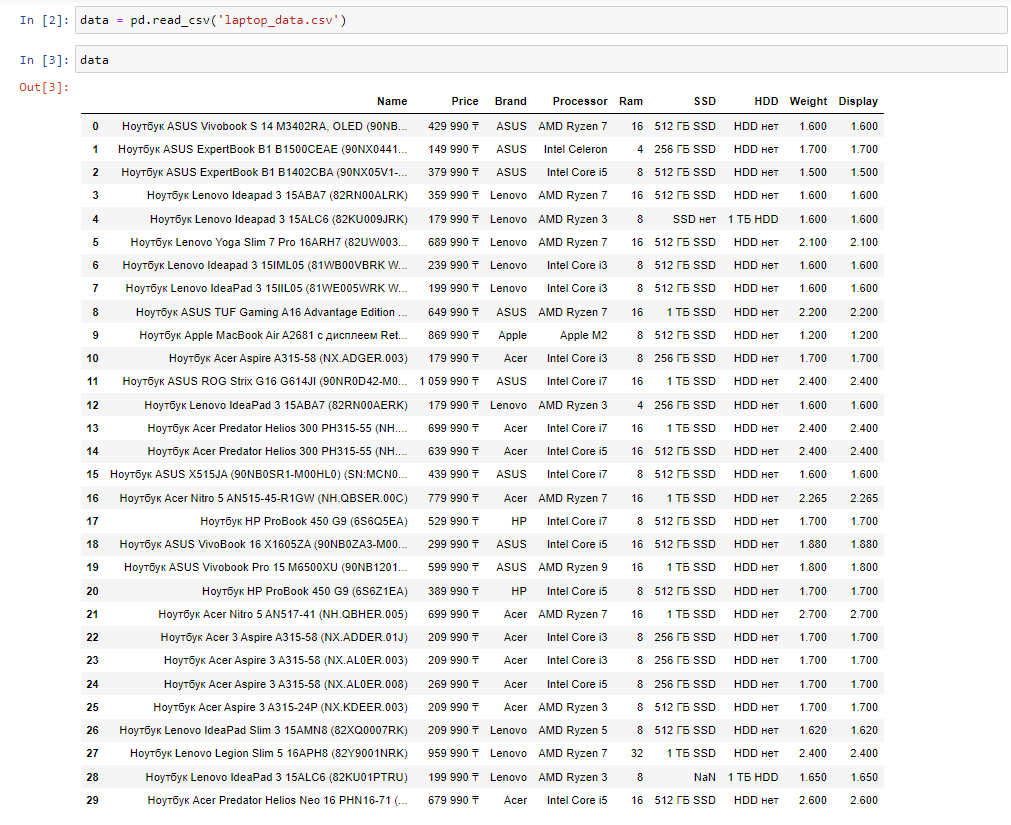
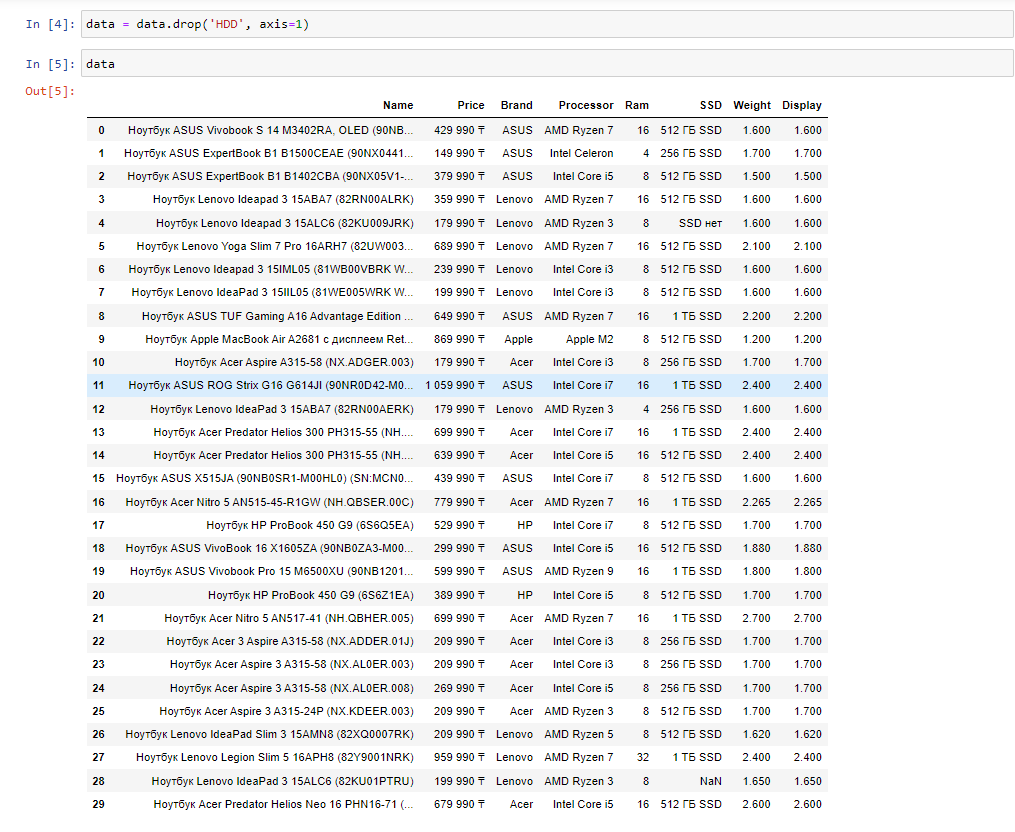


Открываем файл csv в котором содержится наши данные

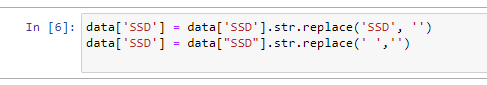


Для начала импортируем все нужные библиотеки

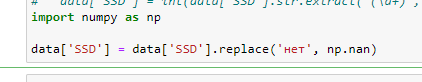
Через функции пандас открываем csv файл, затем выводим таблицу

 Мы видим, что у большинства ноутбуков отсутствует жесткий диск (HDD), поэтому мы удаляем столбец HDD, чтобы избежать некорректной визуализации.

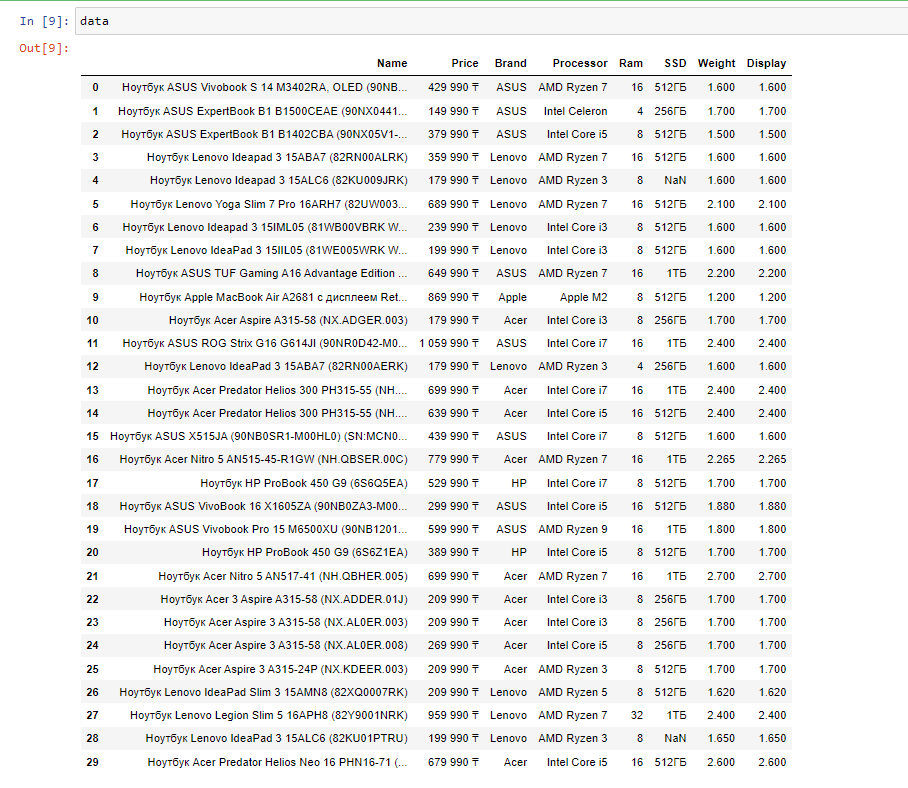
 В столбце SSD тип данных должен быть целым числом. Для этого мы удалим строковые данные, такие как 'ГБ', 'SSD', а также пробелы



Для ноутбуков не имеющих SSD мы присваиваем пустое значение



Отображаем данные

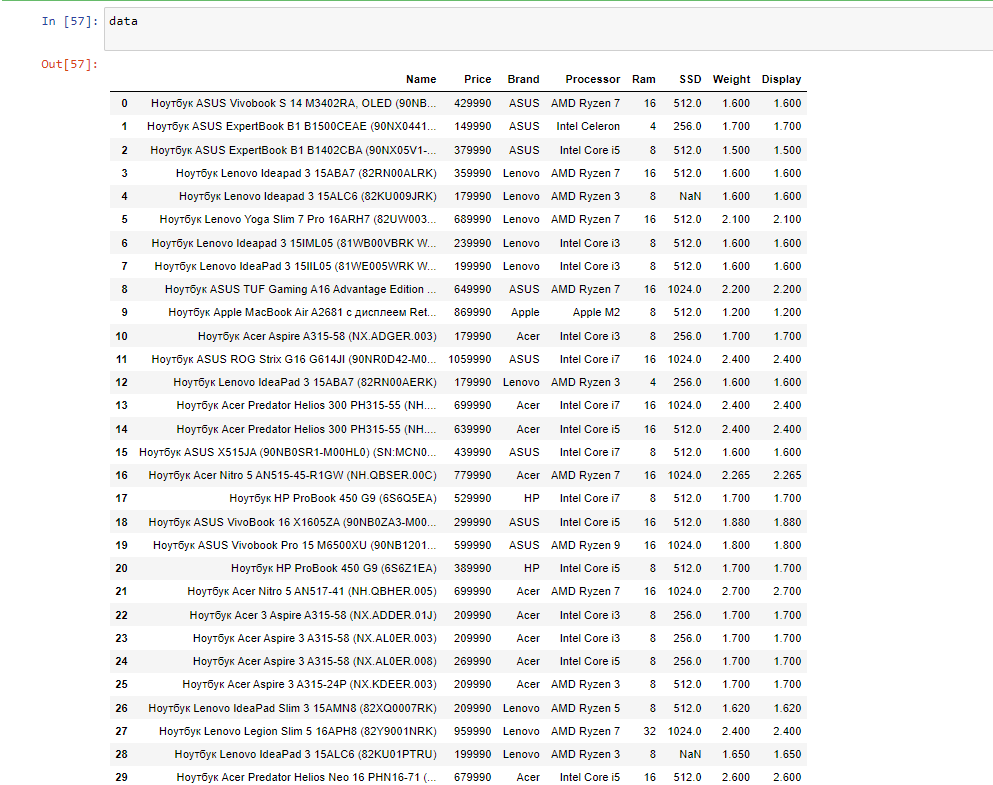


Некоторые SSD в формате гб а некоторые в ТБ и поэтому пишем условия чтобы конвертировать все значения в ГБ



Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Далее очищаем столбец цен, убираем пробелы и удаляем “₸”  


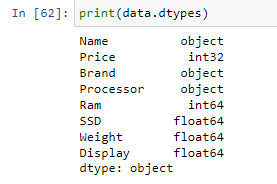
Теперь, когда мы очистили все символы и оставили только цифры, давайте проверим типы данных столбцов в таблице

Изображение выглядит как текст, снимок экрана, Шрифт, число

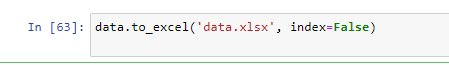
Автоматически созданное описание

Как видно, все столбцы с числовыми значениями имеют типы данных float и integer, за исключением столбца Price. Следующим шагом будет преобразование типа данных столбца Price с object (данные текстового типа) в integer. 

И проверяем



Наш датасет готов для визуализации! Преобразуем CSV-файл в XLSX



Доп материалы - Библиотека Пандас со всеми функциями <https://pandas.pydata.org/pandas-docs/stable/reference/frame.html>