Предпринята попытка совмещения приятного с полезным - получение реально полезных данных с целью их сравнения. Часто сталкивался с такой проблемой, когда необходимо приобрести какую-нибудь вещь, а предложение просто сбивает с мысли - чтобы не растеряться необходимо как то сравнить полученные данные, что представляет сложность, так как данных много, расположены они на разных ресурсах. В таком сравнительном анализе как раз может помочь скрапинг информации и представление ее в удобном для анализа и сравнения виде. Я осуществил скрапинг данных о ноутбуках с 3 популярных маркетплейсов: eBay, Amazon, Aliexpress с целью сведения их в единую таблицу для последующего анализа. Так как на всех 3 сайтах данные загружаются при помощи js, без selenium стреппинг их был бы невозможен. Брались: название (оно содержит и основные характеристики), цена и ссылка.

При скрапинге этих 3 сайтов пришлось использовать индивидуальные подходы к каждому.

На eBay вход на сайт проходил без сложностей, данные поискового запроса выгружались на страницу полностью, поэтому нужно было только дождаться их полной загрузки с помощью ф-ии wait.until().После чего перебрать список полученных элементов и распарсить. Далее перемещались к кнопке перехода на следующую страницу с помощью ф-ии find_element и с помощью ActionChains выполняли нажатие и соответствующий переход

В случае Amazone при входе сайт требует заполнить капчу, что пришлось делать вручную - для этого увеличить время ожидания. Здесь на выданных поиском страницах данные подгружаются динамически по мере прокрутки - используем ActionChains - симуляцию прокрутки страницы колесом мыши. Количество прокруток подобрано экспериментально.

Подобная ситуация имела место и в 3-ем случае (Aliexpress) - при входе - окно о проходящей акции, которое блокировало поисковый ввод, подгрузка данных зависела от прокрутки и данных по запросу оказалось очень много, но большая их часть не соответствовала запросу абсолютно - в итоге просто ограничил переход по страницам - 5-ой страницей, где еще были какие-то данные удовлетворяющие запросу.

Столкнулся с такой трудностью как динамическое название класса - через некоторое время XPATH запрос перестал работать - в пути среди классов был динамический - название - белиберда из букв и цифр - со временем на сайте эта белиберда сменилось другой белибердой - заменил этот XPATH.

Данные сохранил в файл laps.csv - для последующего анализа.