Очень важно для проведения такого рода анализов осуществить доскональное изучение доступных источников данных, провести полноценный анализ сферы коммерческой недвижимости и принципов оценки объектов. []

Для анализа рынка и построения в дальнейшем модели – необходимо собрать основную базу данных недвижимости. В организации, как правило, используется собственная база данных, однако в рамках конфиденциальности собственности компании возникает необходимость использовать данные с открытых источников, тем не менее, в самой компании регулярно практикуется сбор и использование открытых источников, в особенности для обновления базы данных и понимания функционирования рынка. Также, стоит отметить, в рамках работы будут использованы данные именно по Санкт-Петербургу и ближайшей части Ленинградской области, но это не отменяет возможностей использования данного подхода к любой местности, при наличии основных параметров.

На нынешнем рынке есть несколько крупных открытых баз недвижимости. В первую очередь это такие сайты как «Авито», «Циан», «Яндекс Недвижимость». Однако есть и другие, достаточно подробные информационные источники, однако они принадлежат определенным агентствам недвижимости, то есть подразумевают узконаправленность имеющихся данных и не предоставляют необходимый для данного рода работы объем и спектр объектов.

Из указанных источников необходимо выбрать один основной, соответствующий основным параметрам:

* Полнота данных – наличие максимума информации об объекте;
* Целостность и корректность данных;
* Отсутствие ложных данных;
* Наличие критически важных параметров: адрес, квадратура, стоимость.

Стоит рассмотреть конкретные примеры объектов на данных площадках. В таком случае будет возможность изучить возможный набор данных, который можно будет собрать с каждого объекта.

На данном рисунке видно пример объекта с сайта «Яндекс Недвижимость». Если изучить код страницы, а также визуальную составляющую – со страницы объявления можно извлечь следующую информацию:

* URL;
* Стоимость;
* Площадь;
* Стоимость за квадратный метр;
* Адрес;
* Этаж;
* Назначение (офис, склад, торговое);
* Фиктивные переменные, т.е. подробная информация;
* Агентство / Собственник;
* Расстояние до метро в минутах.

Далее рассмотрим пример страницы объявления Авито:

* Стоимость;
* Площадь;
* Стоимость за квадратный метр;
* Фиктивные переменные, т.е. подробная информация;
* Адрес;
* Этаж;
* Агентство / Собственник;

Как последний подробно-изученный источник для будущей базы данных – рассмотрим Циан.

* URL;
* Стоимость;
* Площадь;
* Стоимость за кв метр;
* Дом;
* Этаж;
* Назначение (офис, склад, торговое);
* Тип здания;
* Фиктивные переменные;
* Агентство;
* Расстояние до метро в минутах;

С учетом проведенного анализа, включая изучение корректности данных – было решено, что основным источником для данных стоит использовать Циан. Яндекс и Авито не так подробно изучают объявления и оставляют большое количество ложных объявлений, которые размещают некоторые агентства недвижимости для привлечения клиентов.

ААААААААААААААА

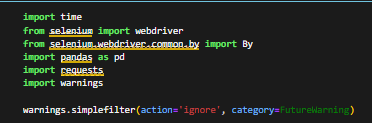
Для сбора данной информации придется обратиться к написанию программы-парсера. Парсер — это программа, сервис или скрипт, который собирает данные с указанных веб-ресурсов, обрабатывает их и выдает в нужном формате. Использование парсеров очень распространено в современном мире, во многом это один из основных способов получения информации из открытого доступа. Более того, в сфере недвижимости это цифровой фундамент для изучения рынка. Все современные агентства стараются постоянно следить за новыми данными и собирать максимум из них. Такой задачей и занимается один из отделов компании «Актив-Инвест».

После изучения основных источников на тему парсеров, их типов, принципов использования и основных библиотек на языке программирования python возникла необходимость также и в изучении основ написания сайтов – html, для понимания структуры страниц и данных на этих страницах. Проконсультировавшись со специалистами в «Актив-Инвест» – была проделана работа по написанию парсера специально для данного сайта, с целью собрать максимум возможной информации.

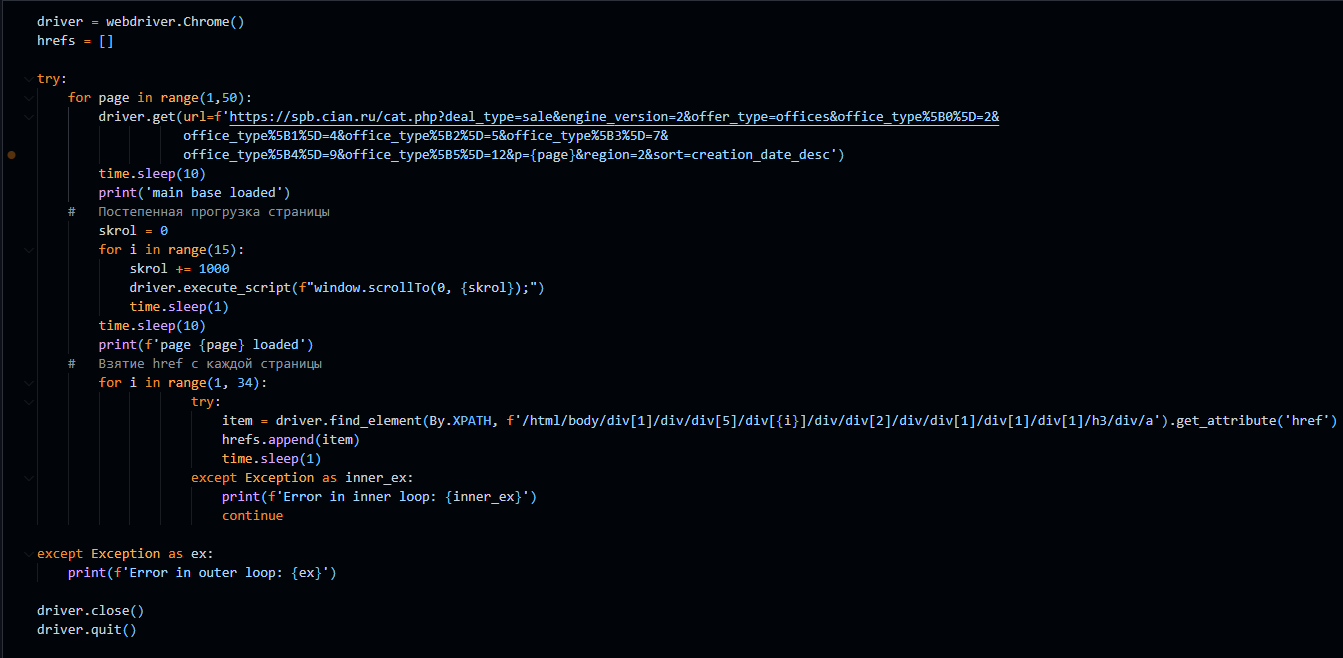
Для данной задачи были изучены основные библиотеки для парсинга на языке программирования python – “Beautifull4Soup” и “Selenium”. Используя основные API этих библиотек, были изучены возможности для сбора данных.

Выводом стало, что самым оптимальным и эффективным способом стоит считать использование библиотеки “Selenium”, во многом благодаря возможностям данной библиотеки более точно взаимодействовать с сайтами, используя имитацию нажатию кнопок, открытия ссылок и перехода от страницы к странице.

Для того, чтобы собрать все объекты, первоначально необходимо собрать все ссылки на них с предварительно настроенной фильтрацией сайта. Настроим среду для кода.



Вот итоговый код для парсинга ссылок всех объектов в сфере коммерческой недвижимости в Санкт-Петербурге и Ленинградской области.



Данный код включает в себя цикл с искусственными задержками, для прохождения по каждому из 34 объявлений на всех 50 страницах сайта с определенным нами заранее условиями фильтрации. В цикле предусмотрено использование скрипта для вертикального перемещения по странице, с целью загрузки и обработки всех объявления, взятие href-ссылок объектов, то есть url-ссылок опубликованных объявлений, учитывая структуру кода сайта «Циан».

Также было предусмотрено возникновение ошибок, почему после возникновения ошибки сам процесс парсинга не прекращается. Каждая ссылка объявления, при этом, записывается в список и затем сохраняется.

После успешного парсинга ссылок на объекты, возникает надобность парсинга самих объявлений, а для данной работы необходимо провести доскональное изучение кода страницы объявлений чтобы ответить на основные вопросы при такого рода парсинге.

Какие объекты размещены? Какой путь они имеют? В каком формате те или данные записываются? Имеет ли значение в каком порядке эти данные собираются? Обратившись к структуре кода одной из страниц (приложение), была написана программа-парсер (приложение)

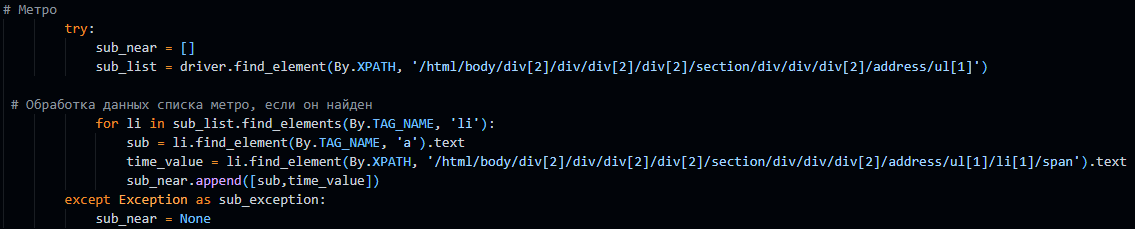
В данном коде происходят действия сбора данных по циклу всех объектов. При открытии каждого объявления, происходит задержка в 10 секунд для полной загрузки страницы и, соответственно, для корректного функционирования парсера. На каждой странице происходит обращение к следующим объектам с извлечением текста и дальнейшим сохранением его, вместе с ссылкой на данное объявление в соответствующем с кодом порядке:

* Название объявления
* Цена объекта
* Цена за квадрат
* Площадь м^2
* Этаж
* Свободно ли помещение
* Город
* Район
* Округ
* Улица
* Номер дома
* Ссылка на страницу продавца
* Описание
* Список ближайших станции метро (мин)
* Список ближайших выездов на шоссе (км).

Работа данного кода суммарно занимает приблизительно 7 часов с учетом обработки всех 1348 объявлений, полученных с работы предыдущего парсера. Такая длительность объясняется наличием задержек для прогрузки страниц и дополнительного времени для сбора информации с каждой страницы.

При изучении написанного парсера, необходимо обратить внимание на структуру внутри цикла. Для каждого из объектов используются операторы “try”, в случае если на какой-либо из страниц не окажется данного объекта или возникнет проблема с его интерпретацией библиотекой selenium. Такая система используется для всех объектов.

Однако ввиду особой структуры объектов, указывающих информацию о расположенных вблизи метро и выездов на шоссе, необходимо было сделать некоторые изменения, а именно:



Поиск элемента, в котором содержатся данные о станциях метро (шоссе), запуск цикла для каждого из элементов в так называемом «списке», затем вывод названия станции метро в отдельную переменную и затем поиск элемента, указывающего на время (км) до метро (шоссе), после чего такой элемент добавляется в список станций метро (выездов на шоссе) у этого объекта.

В случае если у объекта не указаны станции метро (шоссе) – значения столбца остается пустым.

Также стоит отметить, что все эти данные нуждаются в тщательной обработке, ведь все они имеют на данный момент текстовый или даже списочный формат и не подразумевают какой-либо практической ценности.

Рассмотрим пример полученных нами данных:



[]