Наша программа агрегирует информацию о скидках в трех популярных магазинах (Перекресток, Дикси, Магнолия) по выбранной категории товаров. Например, пользователь вводит в строку ввода слово «сок», и программа проверяет три сайта на наличие скидок на выбранный продукт, и по итогу выдает эксель-таблицу с информацией о скидках на товары по этой категории (название товара, размер скидки, оставшиеся дни скидки, если такая информация присутствует на сайте, а также новую цену на товар и ссылку на страницу этого товара на сайте магазина). Для того, чтобы воспользоваться программой, надо просто нажать кнопку «Run» и в появившейся строке ввода вписать интересующий вид товара. Далее надо подождать около минуты, чтобы программа осуществила парсинг сайтов. Затем в папке, где распакован webdriver (в нашем случае это папка Downloads) появится эксель-файл, в котором будет вся вышеперечисленная информация о выбранной категории товара. По желанию пользователь может отсортировать товары по нужной ему категории внутри эксель-файла, например, по цене.

Наша программа может быть полезна людям, которые хотят сэкономить и купить товар по самой выгодной цене.

Описание работы программы

В первую очередь надо импортировать библиотеки, необходимые для работы (**time**, для команды time.sleep(), чтобы страница для парсинга успела прогрузиться, **pandas,** для создания дата фрейма, **selenium**, из которого мы импортируем **webdriver**, для просмотра динамических страниц, по сути это имитация работы реального браузера, **BeautifulSoup** для проведения парсинга, **datetime** для работы с датой и временем.

Далее, для парсинга каждого сайта создается отдельная функция (PerekrestokParsing для Перекрестка, DiksiParser для Дикси, MagnoliaParser для Магнолии). Это делается для удобства работы с кодом. Однако перед этим

задаем функцию Soup, в которую мы через переменную br загружаем webdriver с помощью метода селениума get, а в круглых скобках указываем ссылку, которую мы укажем в переменной url позже. Задаем время сна (time.sleep(3))– три секунды, чтобы сайт успел прогрузиться и парсинг не был впустую, и возвращаем код этой страницы через команду bs(br.page_source). Этот небольшой фрагмент кода мы задаем в отдельную функцию, чтобы его потом не повторять несколько раз.

Перекресток:

Создаем пустой список data, где будет хранится результат парсинга. В переменную url задаем ссылку на сайт, откуда мы будем брать информацию о скидках. Это делается через f-строку с шаблоном. Шаблон – requestWord, то есть переменная, в которую мы через input будем вводить слово поиска, тем самым у нас получается ссылка на конкретную категорию товара. Её мы и будем парсить.

Далее мы применяем функцию Soup.

Через конструкцию try-except мы преодолеваем возможную ошибку, если страниц больше одной или, наоборот, только одна страница. Проблема заключается в том, что если одна страница, то цифры не отображаются для переключения страниц. И в коде мы говорим питону: «попробуй найти lastNum», и если выдаст ошибку, то есть если try не работает, то, скорее всего, страница одна, поэтому в except мы задаем lastNum = 1. Код нам должен найти последнюю страницу, то есть определенное число, которое потом мы в цикле for указываем в качестве последней страницы. Тем самым в цикле for мы прогоняем парсинг по всем страницам со скидками по заданной категории товара. И в список data с помощью метода append записываем цену, название, ссылку, размер скидки и сколько дней скидки осталось.

Дикси:

Также создаем список data, в который мы в конце парсинга магазина добавим пропарсенные данные. Также задаем url, как и в перекрестке, через f-строку. Однако, в отличие от перекрестка, нам не нужна функция Soup, потому что у нас неизвестно количество страниц, как это было в перекрестке.

Проблема данного сайта заключается в том, что в нет переключателя страниц, вместо этого к основной странице возможно добавить новые товары только при нажатии кнопки «показать ещё». То есть мы не можем переходить по страницам так, как мы это делали в Перекрестке. Поэтому мы создали кликер, который автоматически будет нажимать на кнопку «показать еще». Это мы делаем с помощью css selector. С помощью конструкции try-except и цикла while. Пока программа будет находить кнопку, она будет кликать и за каждый клик прибавлять к счетчику единицу. Тем самым сколько кликов получится, такое количество страниц у нас будет минус один. Потому что изначально у нас уже есть открытая страница, которая не требует клика. Чтобы избежать ошибку, которая возникает, когда кнопка больше не появляется, потому что больше нет товаров, мы используем конструкцию try-except, ведь у нас может быть всего одна страница и кнопка «показать еще» будет отсутствовать, для этого в except мы пишем pass.

Далее циклом for проходимся по всем страницам и делаем по сути то же самое, что делали в перекрестке. Единственное, что в дикси есть возможность посмотреть дату окончания скидки. А чтобы получить то, сколько дней осталось до окончания скидки, мы просто из даты окончания скидки вычитаем сегодняшнюю дату (datetime.now().date()). В конце полученную информацию добавляем через append в список data.

Магнолия:

Также создаем пустой список, также указываем ссылку в переменную url. Но с одним нюансом! В параметре limit= в ссылке надо написать большое число,

потому что отображение количества товаров на странице зависит от этого параметра. Наша задача сделать, чтобы страница была одна, а это можно сделать, если указать число заведомо больше возможного числа всех товаров выбранной категории по скидке. В нашем случае – это число 2048. А дальше просто парсим всё то же самое, что и в предыдущих магазинах. И добавляем всё в список data.

В конце идет наш основной код. В переменную br задаем wb.Chrome и задаем относительный путь к вебдрайверу на нашем компьютере, при условии, что exe файл лежит в той же папке, что и наш код, который написан в Jupiter Notebook. Переменная requestWord нужна для введения запроса на нужный товар через input(). Затем создаем пустой список allData, в который поочередно добавляем наши функции по парсингу трех сайтов от requestWord, то есть парсинг будет осуществляться поочередно в магазинах по введенному в переменную requestWord слову. И получается список списков, который мы переносим в дата фрейм в pandas, а уже потом созданный дата фрейм загружаем в эксель.