**Rudraksh Tyagi – 455851**

**WEB SCRAPING PROJECT**

**INTRODUCTION**: Making things affordable for the common man/woman! Elo is committed to making things affordable for common folks by challenging the authority of brands selling low-quality, high-priced goods online and at brick-and-mortar stores. By using recycled or surplus materials of high quality at a lower cost. Thus, passing on the savings to the customers. Elo tries to use recycled and excess materials to eliminate waste wherever possible. This results in great savings for the customer while at the same time helping the environment. Elo believes sustainability should be affordable and available for everyone.

**Description of the Project:**

The main goal of this project is to extract particular information from the Elo website. Subsequently, we will conduct a comparative analysis of three distinct web scraping approaches: Beautiful Soup, Scrapy, and Selenium. The information we intend to extract involves Product links, Product names, and Product prices. Through the implementation of these scraping methods, our objective is to accumulate at least 100 diverse links that provide the specified information. We will be utilizing three distinct scraping mechanisms for this purpose.

Beautiful Soup empowers us to navigate and extract key information from Elo's HTML and XML documents. Through strategic parsing, we're able to unveil the product link, product name, and product price, for comprehensive data collection. On the other hand, Selenium allows us to capture the ELO’s interactive features, enriching our data collection. Lastly, Scrappy is an extensible web scrapping framework that offers a more structured and scalable approach for data extraction.

**Description of all the libraries**   
Beautiful Soup used to extract information from the Elo website. The script commences by importing necessary libraries such as urllib for making HTTP requests, Beautiful Soup for HTML parsing, and pandas for data handling.

The script starts by setting a parameter called 'limit', which determines the number of pages to be scrapped. Empty lists are initialized to hold extracted data including product names, links, and prices. A loop iterates through each page within the specified limit. For each page, an HTTP request is made to retrieve the HTML content.

Beautiful Soup is then utilized to parse the HTML and target specific HTML elements like product names, prices, and links using class attributes. Extracted data is organized into the designated lists. Following data extraction from all pages, the collected information is structured into a Pandas data frame.

**Scrappy**The script commences by importing necessary modules, including Scrapy, Pandas, and Crawler Process from the Scrapy library. Here the 'parse' method processes the HTTP response from the URLs, extracting crucial information. Product links are collected using CSS selectors, while product names and prices are extracted and cleaned using list comprehension and text manipulation. The extracted data is meticulously organized into a Pandas Data Frame, aptly named 'Data Scraped'. This data frame accommodates product names, prices, and links as separate columns. Once data is successfully extracted from all desired pages, it is exported to a CSV file named 'test.csv

**Selenium**The Python script exemplifies the implementation of Selenium for advanced web scraping tasks on the "Elo" website. By proficiently interacting with web elements and employing data manipulation techniques, the script successfully extracts valuable product information. This information is then organized into a structured format, enhancing accessibility and analysis opportunities.

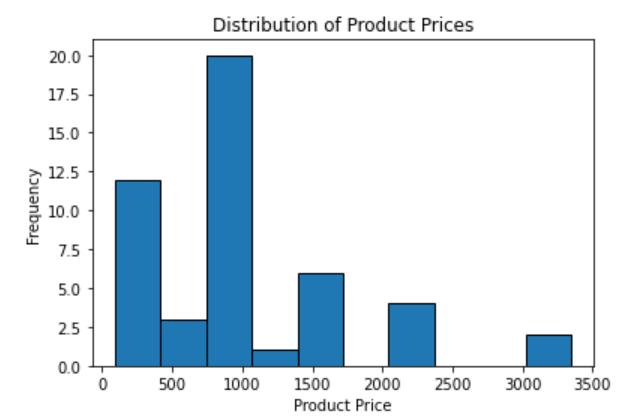
**Technical Description of the output**

The output consists of three essential pieces of product-related information: Product Name, Product Price, and Product Link. The 'Product Name' signifies the name or title of the products available in the Elo website's men's collection. 'Product Price' refers to the cost associated with each product. Lastly, 'Product Link' provides the URL link to access the detailed information of each product. This consistent output structure across the various scraping methods ensures uniformity and makes it convenient for subsequent analysis, visualization, and further utilization

**Data Analysis**

By looking at the given histogram we can identify that the most common price range for products on this page is near 1000. There appears to be an outlier product priced significantly higher than the rest of the products which is close to around 3500. The histogram also shows a narrow spread of prices, suggesting a diverse range of product price points.  
Therefore, Based on the observations, such as:

"The histogram illustrates a skewed distribution towards lower prices, indicating that a majority of products fall within the lower price range. There are a few products with significantly higher prices than the rest, potentially representing premium or unique offerings."



**Detailed Description of each scraper:**  
**Beautiful Soup:** The provided Python code is a web scraper designed using Beautiful Soup to extract information from the Elo website. The script commences by importing necessary libraries such as urllib for making HTTP requests, Beautiful Soup for HTML parsing, and pandas for data handling. The code is tailored to focus on the Elo website's men's collection, aiming to gather details about various products across multiple pages.

The script starts by setting a parameter called 'limit', which determines the number of pages to be scraped. Empty lists are initialized to hold extracted data including product names, links, and prices. A loop iterates through each page within the specified limit. For each page, an HTTP request is made to retrieve the HTML content.

Beautiful Soup is then utilized to parse the HTML and target specific HTML elements like product names, prices, and links using class attributes. Extracted data is organized into the designated lists. Following data extraction from all pages, the collected information is structured into a Pandas data frame.

To refine the data, every alternate entry is removed from the product name and link lists. This could imply a pattern in the webpage's layout. Finally, the extracted data is consolidated into a dictionary and saved into a CSV file named 'bsoup.csv' using Pandas DataFrame functionality.

In summary, the Python code demonstrates an automated approach to extracting valuable product information from the Elo website's men's collection. The use of Beautiful Soup and Pandas enables efficient data extraction, transformation, and storage, facilitating further analysis and utilization."

**Scrappy**: The provided Python code employs the Scrapy framework for web scraping, demonstrating a robust approach to gathering information from the Elo website. The script commences by importing necessary modules, including Scrapy, Pandas, and Crawler Process from the Scrapy library.

A Scrapy spider named Career Spider is defined. This spider is configured to navigate through the target website's men's collection pages, using a list of start URLs generated according to the specified page limit. The core functionality of the spider is encapsulated in the parse method. Upon receiving a response from each URL, the method employs CSS selectors to extract product links, names, and prices. Additionally, the script modifies the extracted product links by concatenating the base URL to each link, forming complete and accurate links to the product. The script employs the Pandas library to create a structured DataFrame named Data\_Scraped. It incorporates the cleansed and modified data, with columns corresponding to product names, prices and links. The script has been saved by saving the data frame as a CSV file named 'Scrapy'  
  
**Selenium:** The script starts by importing necessary libraries, including 'web driver' from Selenium and 'pandas' for data manipulation. These libraries provide the foundation for the web scraping process. The script sets up three empty lists, namely 'p\_name', 'p\_price', and 'p\_link', which are intended to store the extracted product information—names, prices, and links, respectively.

To establish a connection with the website, a Chrome webdriver instance is initialized. The driver is configured to use the Chrome driver executable, which enables controlled interaction with the Chrome browser. The script directs the Chrome browser to the URL of the "Elo" website's men's collection using the specified URL variable. Based on the value of 'page\_limit', the maximum number of pages to be scraped ('max\_pages') is determined as this allows us to control the extent of data extraction.

A loop is implemented to iterate through a range of pages. Selenium's 'find\_elements' method with CSS selectors is utilized to locate the desired elements on the webpage. Following another brief pause, the driver navigates to the next page using a constructed URL that includes the page number.

As data is extracted from each page, it is appended to the respective lists ('p\_price', 'p\_name', 'p\_link'). This iterative accumulation process is crucial for aggregating the complete dataset. Once data has been extracted from all the required pages, the script organizes the collected information into a well-structured Pandas DataFrame named 'Data\_Scraped'. This data frame consists of separate columns for product names, prices, and links. To facilitate further analysis and usability, the extracted data is saved as a CSV file.