Webscraping and Social Media Scraping
2400-DS1WSMS 2020L

Aleksandra Kowalewska, **434585**
Aleksandra Tomczak, **427840**
Marcin Karliński, **427836**

Project Description

Short description of the topic:

Our main idea for the project is to scrape the Top 100 books for a particular month, in the range of multiple months, and some additional information next to it for a post-hoc analysis. Having scraped the lists for the selection of months, we are going to secure the data in Pandas dataframe, being later appended into one aggregate data frame. With this project, we would like to see what and how much changes on the month to month basis in the Top 100 list. On top of that we would like to conduct simple statistical analysis - is there a recurring author, or maybe appearing multiple times on one list?, the maximum and average ratings, popularities etc.

The webpage we are using is https://lubimyczytac.pl/top100

Scraper mechanics:

Our scraper is run in three ways: Beautiful Soup, Scrapy and Selenium. All of them will scrap the same information from the Lubimy Czytać website. Based on the example of Beautiful Soup, first we will scrap the link of months, and afterwards, from each of the months, we scrap the list of attributes: position, title, author, rating, readers, opinions and the number of ratings. In Beautiful Soup this will be achieved through the find all function. Having run the scraper, we put the data into a previously created data frame, first each month separately, to be later on appended into one collective data frame for a clear view of the inputs.

Output description:

The output received is in the form of the concatenated Pandas data frame, consisting of 100 rows (for the Top 100 lists), and the same list of variables for each of the scraped months: position, title, author, rating, readers, opinions and the number of ratings. All that adds up to the dimensions of 100 rows × 84 columns that can be further modified and analysed.

Data analysis:

We performed a simple analysis on the scraped data that consists of:

1. Getting the titles of the books that appear the most often in the Top 100. In our case it is 12 titles that appeared 12 times over the last year. We obtained the list with consisting of : 'Nowy wspaniały świat', 'Gdzie śpiewają raki', '27 śmierci Toby'ego Obeda', 'Cień wiatru', 'Słowik', 'Piętno', 'Sapiens. Od zwierząt do bogów', 'Za zamkniętymi drzwiami', 'Pacjentka', 'Pułapki myślenia. O myśleniu szybkim i wolnym', 'Małe życie', 'Zabić drozda', by first getting the number of titles that appeared the most amount of times during the last year and turning a list of tuples into a list of titles.

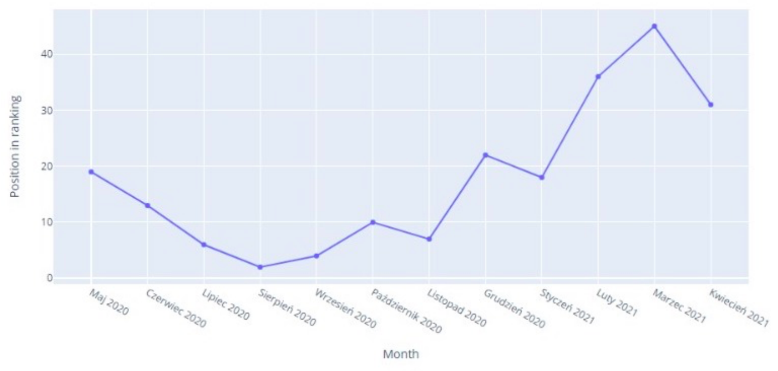
2. From this list of 12 titles we are choosing the Top 5 with the highest ratings.
Having obtained the dictionary of the 12 titles with their corresponding ratings:

```
{'Nowy wspaniały świat': 7.8,
 'Gdzie śpiewają raki': 8.0,
 '27 śmierci Toby'ego Obeda': 8.2,
 'Cień wiatru': 8.2,
 'Słowik': 8.3,
 'Piętno': 7.8,
 'Sapiens. Od zwierząt do bogów': 8.3,
 'Za zamkniętymi drzwiami': 7.5,
 'Pacjentka': 7.4,
 'Pułapki myślenia. O myśleniu szybkim i wolnym': 8.0,
 'Małe życie': 7.8,
 'Zabić drozda': 8.2}
```

We then turn it into a list containing 5 highest ratings:

```
['27 śmierci Toby'ego Obeda',
 'Cień wiatru',
 'Słowik',
 'Sapiens. Od zwierząt do bogów',
 'Zabić drozda']
```

3. We are plotting for every title from the Top 5 how their position in the ranking changed over the last 12 months.



With the simple analysis we performed, it is clear that the scraped data frame is applicable for further analysis and modifications. Having both textual and numerical data allows for a wider spectrum of applicable methods, such as the average rating for the given month, the difference between the most and least popular book on the list, or the author of the book that appears the most times in the dataframe, its genre, plot and so on. Thanks to the dataframe creation, the scraped data is in the user friendly format, available for evaluation.

Code efficiency:

Having compared the performance of three method we got following results:

Beautiful Soup: 53.37s, 118 lines
Scrapy: 4.96s, 98 lines
Selenium: 272.12s, 147 lines

Tasks division:

Aleksandra Tomczak	Beautiful Soup, README.md, Data Analysis
Aleksandra Kowalewska	Scrapy, Description.pdf, Github repository
Marcin Karliński	Selenium, Scrapy