Zuzanna Kostecka (420833)
Karolina Solarska (410858)
Kacper Sokołowski (419067)

Web page:

https://gratka.pl/

Short description of topic:

The project involves scraping data from the website "https://gratka.pl/motoryzacja/osobowe?page=1" which is a section of the website Gratka.pl dedicated to advertisements for personal vehicles. We decided to scrape data from first page to 15th of the "osobowe" (passenger cars) category. The scraped data includes various details about car advertisements such as the link, name of the car, mileage (Przebieg), engine capacity (Pojemność silnika [cm3]), production year (rok produkcji), fuel type (rodzaj paliwa), and price (cena).

Short description of your scraper mechanics - what the program is technically doing.

The “Scrapy” project is used to crawls the website and extracts information about car advertisements. The program starts by defining the Spider class, which includes the URLs to be scraped. In the parse method, it extracts the links to individual car advertisements from the response. For each link, it sends a new request and invokes the parse_advert method to extract specific data from the advertisement page. The extracted data, including the link, name, mileage, engine capacity, production year, fuel type, and price, is then yielded as a dictionary.

The "Beautiful Soup" program also serves as a web scraper. It uses the BeautifulSoup library to extract information from the same website. In the first step, it extracts the links to car advertisements and saves them to a CSV file. Then, in the second step, it reads the CSV file, visits each advertisement link, and extracts the name and mileage (Przebieg) values. It updates the CSV file with the additional information.

The “Selenium” program scrapes exactly the same information as BS and Scrapy ones. At the beginning it opens gratka.pl in order to accept page cookies policy. On the second step, the program extracts adverts URLs from pages that contain redirects. Finally, it opens each scraped advertisement link and gets required information about the car. The results are saved as a CSV file.

Short technical description of the output you get.

The result is a list of car offers with various details. Each row represents a list of cars and contains the following information:

Link: The URL of a particular car listing on Gratka.co.uk.
Name: The title of the advertisement, which includes the name of the car.
Mileage: The mileage of the car (measured in kilometers).
Engine Capacity: The engine displacement in cubic centimeters.
Year of manufacture: The year of production.
Fuel type: The type of fuel used by the car.
Price: The price of the car.

Extremely elementary data analysis - you need to prove, that collected data can be used for further analysis, but nothing more (hard limit of data analysis: one page).

The data consists of information related to different car listings. The provided data can be used for further analysis based on the following elementary data analysis:

```
statistics = data.describe()
print(statistics)
```

✓ 0.2s

	Mileage	Year of manufacture
count	95.000000	478.000000
mean	67537.610526	2021.995816
std	80227.403264	2.711104
min	1.000000	1999.000000
25%	4213.000000	2022.000000
50%	27151.000000	2023.000000
75%	145717.500000	2023.000000
max	260935.000000	2023.000000

The descriptive statistics summary provides an overview of the numerical columns in the dataset, including mileage and year of manufacture. These statistics give us insights into the central tendencies (mean, median), dispersion (standard deviation), and the range of values in these columns. The presence of such statistical measures suggests that basic analysis, such as calculating averages, identifying trends, or comparing values, can be performed on this dataset.

```
missing_values = data.isnull().sum()
print(missing_values)
```

✓ 0.3s

```
Link                    0
Name                    0
Mileage                385
Engine Capacity         19
Year of manufacture      2
Fuel type                0
Price                    0
dtype: int64
```

By checking the missing values, we can assess the completeness of the data. In this case, there are some missing values in the mileage, engine capacity and year of manufacture columns. However, the majority of columns seem to be populated with data, indicating that a significant portion of the dataset is complete and can be used for analysis.

```
unique_values = data.nunique()
print(unique_values)
```

✓ 0.2s

```
Link                    480
Name                    402
Mileage                  77
Engine Capacity          53
Year of manufacture      19
Fuel type                 6
Price                   415
dtype: int64
```

The "unique_values" output demonstrates the number of unique values for each column in the dataset. Having a variety of unique values indicates that there is diversity and differentiation in the data, allowing for more detailed analysis. For example, different car models, engine capacities, or fuel types are present, which can support various analytical approaches.

```
price_distribution = data['Price'].value_counts()
print(price_distribution)
```

✓ 0.0s

```
159 000    4
95 400     4
79 900     4
92 900     3
119 900    3
          ..
253 200    1
136 400    1
122 300    1
79 800     1
37 900     1
Name: Price, Length: 415, dtype: int64
```

Price distribution provides the count of each unique price value and its frequency. For example, there are 4 listings with a price of 95,400, 4 listings with a price of 79,900, and so on. It helps understand the range of prices, the frequency of occurrence for each price point, and any notable patterns or outliers in the data. This information allows also for further analysis of price patterns, identifying popular price points, and understanding the spread of prices across the dataset.

Given this simple analysis, we can conclude that the collected data is suitable for further analysis. They contain relevant information, are fairly complete, contain descriptive statistics, show a variety of unique values and provide insight into the distribution of prices. Further in-depth analysis can be carried out using statistical methods and visualization to gain more insight into the data.

Detailed description which group participant wrote which part of the project.
Zuzanna wrote a scraper using Beautiful Soup.
Karolina wrote a scraper using Scrapy.
Kacper wrote a scraper using Selenium.
We created a description.pdf, made a presentation and attached project on github together.