**Project implementation**

We used the web scraping method in our project. Thanks to the beautifulsoup4 library, we quickly wrote the necessary code in the PyCharm program. This method allowed us to obtain process-mediated data(prices of Lacoste toilet water) from websites of such brands as Empik, Supherpharm, Notino, Perfumeria and Perfumy.

A piece of code used to retrieve information from websites:

*import* requests  
r=requests.get("https://www.perfumeria.pl/collections/perfumy/producent-lacoste")  
*from* bs4 *import* BeautifulSoup  
soup = BeautifulSoup(r.text, 'html.parser')  
names= soup.findAll('a', attrs={"/collections/perfumy/products/lacoste-pour-femme-woda-perfumowana-spray"})  
prices = soup.findAll('span', attrs={"price"})  
  
*for* name *in* names:  
 print(name.text.strip())  
*for* price *in* prices:  
 print(price.text.strip())  
  
f=open("perfumerianame&price.csv","w")  
*for* price *in* prices:  
 print(price.text.strip())  
 f.write(price.text.strip("Od") + "\n")  
f.close()

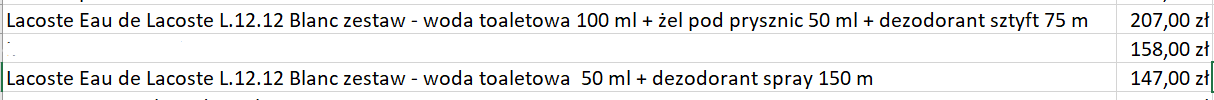
We received a variety of data, because stores often had different perfume models and a different amount of Lacoste assortment. We have included both the prices of women's and men's perfumes. Some prices included a discount, but they were available for all viewers of the site, not only selected customers. For this reason, we decided to keep the "new prices". We narrowed the test down by discarding perfumed water, which differs from toilet water in price and quality. Then we prepared the data in Microsoft Excel, and used the same program for data visualization and analysis.

A piece of code that helped us compare perfume models:

*import* csv  
f= open("superpharmprice&name.csv",'r')  
csvreader = list(csv.reader(f, delimiter=','))  
print(csvreader)  
SP=["Superpharm"]  
SPname=[]  
SPprice=[]  
a=0  
*for* i *in* range(1,9):  
 SPname.append(csvreader[i][0])  
 SPprice.append(csvreader[i][1])  
*for* name *in* SPname:  
 *if* name=="LACOSTE Red":  
 SP.append(name)  
 SP.append(SPprice[a])  
 a = a + 1

While collecting data, we came across some errors. In the beginning, we planned to collect the prices of contact lenses, but on a large scale their price depended on the size of the order. We then decided that we should deal with another topic. With perfumes, the price also depended on some factors, like quantity of perfume in milliliters. However, in this case, the size differences could easily be eliminated by using search filters in the online store. Despite this narrowing, we also had to remove the phrase "prices from" in one store using the split function. It allowed us to display a specific price rather than the price range.

Yet another obstacle was that the sites offer gift sets containing toilet water, which were included in the ‘toilet water’ section. It had to be filtered out in the code. Despite this fact, data from perfumy.pl still included the gift sets and required more filters, which was problematic.



Each brand structured their data a bit differently, for example Superpharm does not display perfume size on the page listing all the products – this data is only available once the user chooses a specific one:





After all, we decided on a more general approach. The results will not be filtered by size, as finding the cheapest available option would be our main goal. Users usually choose price filters first, and then focus on other options in the preferred price range.

Once the data was scraped and filtered, we could move on to writing the price comparing code. Product names and prices from each store were saved in separate .csv files, and the information on the product of interest was extracted using the toilet water name as a keyword. As a result, a .csv file containing all available prices of a given product has been created. (pewnie ostatecznie zrobiłaś to jakoś inaczej, popraw moją logikę :D)

<CODE>

Finally, the prices were sorted in two ways: high to low, and low to high. (tu napisalam o tym sortowaniu od najwyzszej/od najnizszej – wydaje mi sie ze to raczej izi ale jakby co to mozesz negocjowac albo to wywalic :D)

<CODE>

Once this data was obtained, we could move on to the result analysis.