**Introduction**

The purpose of this script is to scrape data from various e-commerce websites and store them in a way that they can be later used to create a new website with the products it gathered. For this purpose, a local MySQL database is being used. The code is static, which means that for every site from which we have gathered data we have also set specific tags in order to gather all the necessary information. However, by acknowledging this issue we have taken additional steps in order to make the code easily expandible by using dictionaries to store the tags for every kind of product.

**Setup Instructions**

**Libraries**

We’ve used various libraries in order to achieve the required outcome. Below is a list with a description for every library we utilized.

* **“requests”:** This is the de facto standard library for making HTTP requests.
* **“beatifulsoup4”:** Is a library that is being used to pull data out HTML and XML.
* **“pandas”:** Pandas is being used for data manipulation and analysis.
* **“selenium”:** Detailed analysis below.
* **“mysql-connector-python”:** This library is used for connecting and interacting with MySQL databases.

In order to use the above libraries, someone has to install them. To do so, open the command prompt and type the command as seen below:

* pip install requests
* pip install beautifulsoup4
* pip install pandas
* pip install selenium
* pip install mysql-connector-python

The uses of all libraries are quite straightforward except perhaps for the case of selenium.

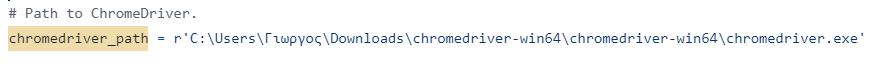
Selenium is being used in this script because of a specific case where we were unable to get the description of cookshop by using only *beutifulsoup4.* Beautifulsoup4 can only access static HTML content and in our case the description was loaded in the page in a second phase. With selenium we were able to wait for the element of description to load and then capture it.

In order to use the selenium someone has to take some additional steps other than pip installing the library

* **Download chromedriver:** In order to download the correct version of the chromedriver the version of chrome, on the terminal that is being used, has to be checked. Then the link below can be used to detect the correct version and download it.

<https://googlechromelabs.github.io/chrome-for-testing/latest-versions-per-milestone-with-downloads.json>

* **Unzip:** Unzip the chromedriver to the desired location.
* **Update path:** Update the path of chromedriver in the script in order to point the executable file



**Storing Data in MySQL**

In order for the connection with MySQL to be successful the following steps should be followed:

* Install MySQL: <https://dev.mysql.com/downloads/>
* Create a new instance
* Create a new database: CREATE DATABASE name\_of\_the\_database;
* Be sure to update the code with the correct credentials as seen in the picture below.

A computer screen shot of a code

Description automatically generated

The people who will use this script won’t have to create a table since it already creates one if none is found.

**Error and Missing Data Handling**

The script contains a lot of commands that print the state of the script while parsing the webpages. This was quite helpful to us in order to detect the exact web pages were the code was breaking.

Also, we have considered the cases where specific information is missing from a webpage. In these cases, we store the data in a way that the users can recognize it easily by looking into the database table. For example, if a description is missing the description cell for this specific product should be populated by the value: “No description”.

**Suggestion for Expansion**

As it was already mentioned the script was optimized (given the length of our knowledge) in way to make it easily expandible in order to add more shops and products. Another approach could be to make it more interactive and interesting in way that it asks the users the URLs (url, base\_url) and the tags in order to produce a tailored outcome without the need of code interaction. This could be used by people who are a little bit more familiar with HTML than they are with python but still requires more than basic knowledge of the matters. Also, the detection of tags is time-consuming and prone to errors process.

A more interesting idea would be to train a machine learning model to identify the required tags and update dynamically the dictionaries. Then the users will have to provide only the desired links. This approach requires a significant effort but it would get rid of the need of statically updating the script every time a new shop or product type is to be gathered.