**Web Scraper Methodology**

This Python code is a web scraper for the Moroccan website, Avito.ma. The scraper focuses on extracting information about various items for sale, with a particular focus on cars.

We used:

* Requests and Beautiful Soup Librairies to scrape the data from the web site.
* Pandas to process and store the data in a DataFrame
* Tkinter library to instantiate a GUI (Graphical User Interface) to make the web scraping more user-friendly.

The code is organized into several classes, functions, and methods:

* **handle\_request\_error**: A decorator function to handle request errors while making HTTP requests. Decorators are functions which take as argument another function as an object, which enables us to put our logic either at the start and end of the execution of the argument function.
* **AvitoScraper**: A class to represent the scraper.

It contains the following methods:

1. **\_\_init\_\_** - Initializes the class with the root Tkinter object, sets up variables, and creates the widgets for the user interface.
2. **create\_widgets** - Creates the user interface with input fields, dropdown menus, and buttons.
3. scrape - Scrapes the Avito website based on the user's selected city and keyword. It fetches the relevant information, such as title, description, price, date, and other attributes, depending on whether it is a car or another item.
4. **save\_to\_excel** - Saves the scraped data to an Excel file with a default file name based on the current date.
5. **schedule\_scrape** - Schedules the scrape method to run every day at 8 am.

The code first imports necessary libraries, such as pandas, requests, tkinter, BeautifulSoup, and others. When executed, the script initializes the AvitoScraper class with the root Tkinter object and starts the Tkinter main event loop to display the user interface. The user can then input their desired search parameters and click the "Scrape" button to start the web scraping process. Once the scraping is complete, the user can save the results to an Excel file by clicking the "Save to Excel" button, this request will be automatically rerun at 8am every day.

We circumvented the error 403 in web scraping for the website avito.com by adjusting the user agent string using Beautiful Soup. We modified the HTTP request headers by setting the "USER\_AGENT" parameter to a custom user agent string that mimicked a popular web browser like Chrome. By doing so, we were able to retrieve the desired data without encountering the error 403.

We also added a unittest file to the project, which contains test cases for the AvitoScraper class. The file imports the necessary libraries and classes, defines the test cases for the scraper, and uses the unittest module to run the tests.

**In summary, this code is a web scraper for the Avito website that allows users to search for items or cars based on specified keywords and city. It extracts relevant information and saves it to an Excel file for further analysis.**