**Python Assessment Report**

**Task 1: Web Scraping and Data Extraction**

In this task, I was assigned to scrape a website, handle a captcha input, extract data from an image, and then manipulate the extracted data using Python libraries. Below is a detailed explanation of the steps I followed to accomplish the task:

### **1. Web Scraping and Data Extraction**

#### **a. Website Scraping**

To begin, I set up the environment for web scraping. The website required user input through a form field, including the need to enter a captcha before obtaining the desired data. I chose **Selenium** for this task due to its ability to automate browser interactions.

Steps involved:

* **Installed Selenium**: Selenium was installed locally using pip install selenium. I also installed the **ChromeDriver** as Selenium requires a browser driver to control Chrome for scraping.
* **Web Automation**: I imported the necessary modules from Selenium (e.g., webdriver) and wrote functions to automate the website navigation. This included opening the website, filling in the input fields, and handling the captcha.

#### **b. Captcha Handling**

The next part of the task was to enter the captcha. This involved manually entering the captcha each time the script was executed, as captcha bypassing typically requires more complex solutions, which were outside the scope of this task.

#### **c. Data Extraction from Image**

After submitting the form, an image containing the relevant data was displayed on the webpage. The task involved extracting data from this image.

Steps involved:

* **Tesseract OCR Setup**: To extract data from images, I installed **Tesseract OCR** locally. I configured it by adding the Tesseract path to the system environment variables to ensure the pytesseract Python library could interact with it.
* **Image Processing**: I used the **PIL (Python Imaging Library)** to preprocess the image. Preprocessing steps included converting the image to grayscale, adjusting contrast, and resizing it to improve the accuracy of OCR.
* **Text Extraction**: Using pytesseract, I extracted the text from the image and converted it into a **pandas DataFrame** for further manipulation. This step involved cleaning the extracted text to match the required format.

### **2. Second Part of the Task**

The second part of the task was similar to the first, where I had to repeat the process of web scraping and data extraction for another input. The same steps of Selenium-based scraping, captcha entry, and image-based data extraction were followed to accomplish this.

### **3. Storing Data in SQLite Database**

After successfully extracting and cleaning the data, I needed to store it in a local database. I used **SQLite** for this purpose.

Steps involved:

* **SQLite Setup**: I used Python's sqlite3 library to create a database and a table for storing the extracted data.
* **Data Insertion**: The cleaned pandas DataFrame was inserted into the SQLite database using df.to\_sql().

### **4. Writing SQL Queries for Data Analysis**

Once the data was stored in the SQLite database, I wrote queries to answer the given questions. The queries were designed to:

* **Filter data**: For example, extracting rows where KWH units were less than a certain value.
* **Filter by date range**: Querying data for a specific period (e.g., between February 2023 and June 2023).
* **Add calculated fields**: I added a new column to the DataFrame that calculated values based on the existing data.
* **Update specific rows**: For instance, I modified the rows where the month was December 2023, replacing all numeric values with a constant value.

### **Summary of Tools and Technologies Used:**

* **Selenium**: For browser automation and web scraping.
* **BeautifulSoup4**: For parsing and extracting data from HTML (when needed).
* **Pandas**: For data manipulation and conversion into DataFrames.
* **Pytesseract**: For extracting text from images using Optical Character Recognition (OCR).
* **SQLite**: For storing the extracted and processed data.
* **SQL Queries**: For data retrieval and filtering based on specific conditions.

### **Conclusion**

This task involved a combination of web scraping, image-based data extraction using OCR, and database manipulation using SQLite. By leveraging various Python libraries, I successfully automated the process and retrieved the desired data efficiently, answering the required questions through SQL queries and DataFrame operations.