Scraping Profiles and Jobs from LinkedIn

The code you provided is using Selenium to scrape profiles and jobs from LinkedIn. Here's a breakdown of each part:

First Part:

1. Import necessary libraries and modules from Selenium and other dependencies.
2. Create an instance of the Chrome web driver.
3. Navigate to the LinkedIn login page.

Second Part:

1. Set the login credentials (username and password) to variables.
2. Find the username and password input fields on the login page using their IDs.
3. Enter the login credentials into the respective fields.
4. Find the login button using its class name.
5. Click on the login button to submit the form.

Third Part:

1. The code defines several functions that are responsible for extracting specific information from job listings on LinkedIn.
2. Each function uses different CSS selectors and methods provided by Selenium to locate and extract the desired information from the web page.
3. The functions handle different scenarios where the information may not be found on the page, and they return appropriate default values in those cases.

Let's go through each function in more detail:

Profile's Function

1. **find_name_location_description(driver=driver)**: Extracts the name, location, and description of a profile from a web page.
2. **find_current_company(driver=driver)**: Retrieves the current company information from a web page.
3. **find_experience(driver=driver)**: Retrieves the experience information from a web page.
4. **find_about(driver=driver)**: Captures the "About" section of a profile on a web page.
5. **find_education(driver=driver)**: Extracts the education information from a web page.
6. **find_skills(driver=driver)**: Retrieves the skills information from a web page.
7. **have_experience(driver=driver)**: Checks if a profile has experience information.
8. **find_university_education(driver=driver)**: Extracts university education information from a web page.
9. **find_certifications(driver=driver)**: Retrieves the certifications information from a web page.

Job's Function

1. Sure! Here's a concise summary of each function in the fourth part:
2. **`get_company_name(driver=driver)`:** Retrieves the company name from a job listing, handling exceptions.
3. **`get_location(driver=driver**)`: Gets the location of a job listing, handling exceptions.
4. **`get_type(driver=driver)`:** Retrieves the type of workplace (e.g., full-time, part-time) for a job listing, handling exceptions.
5. **`get_description(driver=driver)`:** Extracts the job description from a job listing, handling exceptions.
6. **`get_industry(driver=driver)`:** Retrieves the industry of a job listing, handling exceptions.
7. **`get_type_work(driver=driver)`:** Gets the type of work (e.g., Information Technology, Finance) for a job listing, handling exceptions.
8. **`get_skills(driver=driver)`:** Extracts the skills mentioned in a job listing, handling exceptions.
9. **`get_jobTitle(driver=driver)`:** Retrieves the job title from a job listing, handling exceptions.
10. **`extract_degree_requirement(job_description)`:** Extracts the degree requirement line from the job description. If no degree requirement is found, it returns an empty string.

These functions provide a way to extract specific information from job listings on LinkedIn using Selenium. They handle different scenarios where the information may not be available and return appropriate default values to ensure the code's robustness.

Fourth Part:

Explain Automation Code

profiles

1. The user is prompted to select the type of scrape: either scraping profiles or scraping jobs.
2. If the user chooses to scrape profiles (scrape = 1), they are prompted to select the country to scrape, the number of profiles to scrape, and a CSV file named "profiles.csv" is read.
3. A random number between 2 and 50 is generated as the start page for scraping.
4. The loop iterates through the pages, navigates to each page, and collects the profile links.
5. For each profile link, the necessary information (name, location, description, about, experience, university, skills) is extracted and stored in the "info" list.
6. The "info" list is converted into a DataFrame, and the skills and university columns are separated into individual columns using the "separate" function.
7. The "description" column is modified to categorize profiles based on specific keywords.
8. The location column is modified to standardize the location values for Lebanon.
9. The DataFrame is concatenated with the existing "profiles.csv" file, and the updated DataFrame is saved back to "profiles.csv".

Job

1. If the user chooses to scrape jobs (scrape = 2), they are prompted to select the country to scrape, the number of jobs to scrape, and a CSV file named "jobs.csv" is read.
2. The job links are collected by navigating through multiple pages and scrolling down to load more job listings.
3. For each job link, the necessary information (job title, company name, location, type, description, degree requirement, industry, type of work, skills) is extracted and stored in the "info" list.
4. The job title is modified to include only the first three words.
5. The degree requirement column is updated to convert abbreviated degrees into their full names.

6. The location column is modified to standardize the location values for Lebanon.
7. The skills column is separated into individual columns using the "separate" function.
8. The DataFrame is concatenated with the existing "jobs.csv" file, and the updated DataFrame is saved back to "jobs.csv".
9. The degree columns in the DataFrame are converted from abbreviation to full name.