Web Scraping and File Download Script

Introduction

This Python script is designed for web scraping and file download tasks. It uses the Selenium WebDriver and Requests library to navigate web pages, identify links, and download both HTML and PDF files. The script is particularly tailored for the website Urban Redevelopment Authority (URA) of Singapore.

Requirements

Make sure you have the following Python packages installed:

- **requests**
- **selenium**

You also need to download the appropriate ChromeDriver version compatible with your Chrome browser and update the **chrome_driver_path** variable in the script with the path to the downloaded **chromedriver** executable.

Functionality

is_pdf(url)

This function checks if a given URL points to a PDF file by sending a HEAD request and examining the **Content-Type** header.

download_file(url, directory)

Downloads a file (PDF or otherwise) from a given URL and saves it to the specified directory. If a file with the same name already exists, a new filename with an incremental index is created.

download_page(url, directory)

Downloads an HTML page from a given URL and saves it to the specified directory. Similar to **download_file**, it handles filename conflicts by appending an incremental index.

get_related_links_recursive(driver, base_url, url, excluded_keywords=[], visited_urls=set(), download_directory='downloads', depth=1)

This is the main recursive function for retrieving related links from a specified URL. It navigates the page, identifies links, filters them, and downloads HTML pages. It continues to follow links recursively up to a specified depth.

- **Parameters:**
    - **driver**: Selenium WebDriver instance
    - **base_url**: The starting URL for the scraping process

    - **url**: The current URL being processed
    - **excluded_keywords**: A list of keywords to exclude from the filtered links
    - **visited_urls**: A set to keep track of visited URLs
    - **download_directory**: The directory where files will be downloaded
    - **depth**: The depth of recursive crawling

- **Returns:**
    - A list of filtered links

Usage

- Update the **chrome_driver_path** variable with the correct path to your **chromedriver** executable.
- Specify the **base_url** variable with the starting URL for scraping.
- Run the script.

```
python script.py
```

Note

- The script is tailored for the URA website structure, so modifications might be needed for other websites.
- It handles PDF and HTML files differently during download to account for potential duplicates.