Documentation for PDF Scraper Project

Project Source

- **GitHub Repository:** GitHub Link For Code Files
- **Google Drive:** Click here for link that contain Complete Scraping Code & Sample PDF Files

Introduction

The PDF Scraper project is designed to download PDF files from a specific website for a given date range. This documentation provides an in-depth explanation of the code, the approach used for scraping PDF data, the APIs utilized, and the step-by-step flow of the code.

Approach

The main approach of the project involves:

1. Iterating through a specified date range.
2. Making POST requests to retrieve HTML content for each date.
3. Parsing the HTML content to extract links to PDF files.
4. Downloading the PDF files and saving them in date-specific folders.

APIs Used

- **Main API Endpoint**: `https://monitoruloficial.ro/ramo_customs/emonitor/get_mo.php`
  - This API is hit with a POST request to retrieve the HTML content for a specific date.
- **Document Metadata API**: `https://monitoruloficial.ro/ramo_customs/emonitor/gidf.php`
  - This API is used to retrieve metadata required to construct the URL for downloading complex PDFs.

Code Explanation

Main Functions

```
Complex_pdf_Download(initial_url, directory_name, file_name)
```

- **Purpose**: Downloads complex PDFs that require additional metadata extraction.
- **Process**:
  1. Initiates a session and sets headers for the request.
  2. Fetches the initial HTML page and extracts the `fid` value from a script tag.
  3. Uses the `fid` to request document metadata.
  4. Constructs the PDF URL using the metadata and downloads the PDF.

5. Saves the PDF in the specified directory.

`simple_download_pdf(url, session, directory_name, file_name)`

- **Purpose**: Downloads simple PDFs directly from the provided URL.
- **Process**:
    1. Initiates a session and sets headers for the request.
    2. Downloads the PDF from the provided URL.
    3. Saves the PDF in the specified directory.

`iterate_dates(start_date, end_date)`

- **Purpose**: Generates dates between the start and end dates.
- **Process**:
    1. Converts the start and end dates from string format to `datetime.date` objects.
    2. Iterates through each date in the range, yielding one date at a time.

`scrape_html(html_content)`

- **Purpose**: Extracts PDF links from the HTML content.
- **Process**:
    1. Parses the HTML content using BeautifulSoup.
    2. Finds all div elements with the class `card-body`.
    3. Extracts links to both simple and complex PDFs based on the content structure.

`main(start_date, endDate)`

- **Purpose**: Coordinates the overall scraping process.
- **Process**:
    1. Iterates through each date in the specified date range.
    2. Makes a POST request to retrieve the HTML content for each date.
    3. Uses `scrape_html` to extract PDF links from the HTML content.
    4. Downloads both simple and complex PDFs using the respective functions.

Dependencies

The script relies on several Python libraries:

- `requests` for making HTTP requests.
- `beautifulsoup4` for parsing HTML content.
- `os` and `datetime` for file handling and date manipulation.

Example Usage

To scrape PDFs from June 17, 2024, to June 20, 2024, update the `Start_Date1` and `End_Date1` variables in the `scraper.py` file and run the script:

```
# Example:
Start_Date1 = '2024-06-17'
End_Date1 = '2024-06-20'
```

Request for Feedback

If you require any changes or have any feedback, please let me know. I am happy to improve the project and meet your requirements, as I offer unlimited revisions in my package.