Web Scraping Module for Slave Labor Data

Module: ws_df_escravo.py

System Technical Documentation

January 3, 2026

Automated data extraction system from
Brazilian Federal Government portal

Contents

1 Overview

This module implements an automated web scraping system for extracting data related to the fight against slave labor in Brazil, made available by the Ministry of Labor and Employment through the official Brazilian government portal.

1.1 Purpose

The main objectives of this code are:

- Automatically access the governmental portal for combating slave labor
- Identify and filter available CSV files for download
- Download and process the files
- Organize data into pandas DataFrame structures

1.2 Data Source

Base URL:

```
https://www.gov.br/trabalho-e-emprego/pt-br/assuntos/
inspecao-do-trabalho/areas-de-atuacao/combate-ao-trabalho-escravo-e-analogo-ao-de-escrav
```

2 Dependencies

The module requires the following Python libraries:

Library	Alias	Function
BeautifulSoup	bs	HTML document parsing and manipulation
requests	-	HTTP request execution
urllib.parse	urljoin (uj)	Absolute URL construction
pandas	pd	Tabular data manipulation and analysis
io	StringIO	String to file-like object conversion

Table 1: Module dependencies

3 Code Components

3.1 Function: is_file_link

```
1 def is_file_link(href):
2     if not href:
3         return False
4     href = href.lower()
```

```
5     return href.endswith('.csv')
```

Listing 1: Link filter function

3.1.1 Description

Validates whether a given hyperlink reference points to a CSV file.

3.1.2 Parameters

- `href` (str): Hyperlink reference to be validated

3.1.3 Returns

- `bool`: `True` if the link ends with `.csv`, `False` otherwise

3.1.4 Logic Flow

1. Checks if `href` exists (not `None` or empty)
2. Converts the string to lowercase for case-insensitive comparison
3. Verifies if the string ends with the `.csv` extension

4 Main Script Execution

4.1 Configuration and Initial Setup

```
1 url = 'https://www.gov.br/trabalho-e-emprego/pt-br/assuntos/
    inspecao-do-trabalho/areas-de-atuacao/combate-ao-trabalho-
    escravo-e-analogo-ao-de-escravo'
2
3 headers = {
4     'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
        AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
        Safari/537.36'
5 }
```

Listing 2: URL and headers configuration

4.1.1 User-Agent Header

The User-Agent header is configured to simulate a request from a Chrome browser on Windows 10. This is necessary because:

- Many servers block requests without proper User-Agent identification
- It ensures compatibility with the server's access policies
- Prevents the request from being identified as an automated bot

4.2 HTML Content Retrieval

```
page = requests.get(url, headers=headers).text
soup = bs(page, 'html.parser')
```

Listing 3: HTTP request and HTML parsing

4.2.1 Process

1. Executes an HTTP GET request to the target URL
2. Retrieves the HTML content as plain text
3. Parses the HTML using BeautifulSoup with the `html.parser` engine

4.3 Link Extraction and Filtering

```
links = soup.find_all('a', class_='internal-link')
file_links = [a for a in links if is_file_link(a.get('href'))]
```

Listing 4: CSV link identification

4.3.1 Operation

1. Searches for all anchor tags (`<a>`) with class `'internal-link'`
2. Applies list comprehension to filter only links pointing to CSV files
3. Uses the `is_file_link` function as filter criteria

4.4 Data Collection and Processing

```
df = []
nomes = []
for link in file_links:
    href = link.get('href')
    arquivo_url = uj(url, href)
    nomes.append('a' + arquivo_url.split('/')[-1].replace('.csv',
        '').lower())

    r = requests.get(arquivo_url, headers=headers)
    csv_text = r.content.decode("latin1")
    data = pd.read_csv(StringIO(csv_text), sep=";")
    df.append(data)
```

Listing 5: CSV file download and processing loop

4.4.1 Processing Flow

1. **URL Construction:** Builds absolute URLs using `urljoin`
2. **Name Generation:** Creates identifiers by:
   - Extracting the filename from the URL
   - Removing the `.csv` extension
   - Converting to lowercase
   - Prefixing with 'a'
3. **File Download:** Executes HTTP GET request for each CSV file
4. **Encoding Conversion:** Decodes binary content using `latin1` encoding
5. **DataFrame Creation:** Parses CSV with semicolon (;) as delimiter
6. **Storage:** Appends DataFrame to the collection list

5 Data Structures

5.1 Output Variables

Variable	Type	Description
`df`	list	List containing pandas DataFrames, one for each processed CSV file
`nomes`	list	List of strings with generated identifiers for each file

Table 2: Global output variables

6 Technical Considerations

6.1 Character Encoding

The code uses `latin1` (ISO-8859-1) encoding for decoding CSV files, which is appropriate for Portuguese language content from Brazilian government sources.

6.2 CSV Delimiter

The CSV files use semicolon (;) as field delimiter, which is common in Brazilian government datasets and differs from the comma-separated standard.

7 Usage Example

```python
# Import the module
import ws_df_escravo

# Access the processed data
dataframes = ws_df_escravo.df
file_names = ws_df_escravo.nomes

# Example: Display first DataFrame
if dataframes:
    print(dataframes[0].head())
    print(f"File identifier: {file_names[0]}")
```

Listing 6: Module usage

8 Conclusion

This module provides a straightforward and effective solution for automated extraction of slave labor data from the Brazilian government portal. The implementation leverages industry-standard Python libraries to ensure reliability and maintainability.

End of Documentation