Scraping Daily Pennsylvanian Headlines - Claude

claude.ai/chat/e4b845d2-52ee-479b-a42a-c5ad0ef17a25

JW

Explain this code """ """ Scrapes a headline from The Daily Pennsylvanian website and saves it to a JSON file that tracks headlines over time. """ import os import sys import daily_event_monitor import bs4 import requests import loguru def scrape_data_point(): """ Scrapes the main headline from The Daily Pennsylvanian home page. Returns: str: The headline text if found, otherwise an empty string. """ headers = { "User-Agent": "cis3500-scraper" } req = requests.get("https://www.thedp.com", headers=headers) loguru.logger.info(f"Request URL: {req.url}") loguru.logger.info(f"Request status code: {req.status_code}") if req.ok: soup = bs4.BeautifulSoup(req.text, "html.parser") target*element* = *soup.find("a", class*="frontpage-link") data_point = "" if target_element is None else target_element.text loguru.logger.info(f"Data point: {data_point}") return data_point if **name** == "main": # Setup logger to track runtime loguru.logger.add("scrape.log", rotation="1 day") # Create data dir if needed loguru.logger.info("Creating data directory if it does not exist") try: os.makedirs("data", exist_ok=True) except Exception as e: loguru.logger.error(f"Failed to create data directory: {e}") sys.exit(1) # Load daily event monitor loguru.logger.info("Loading daily event monitor") dem = daily_event_monitor.DailyEventMonitor("data/daily_pennsylvanian_headlines.json") # Run scrape loguru.logger.info("Starting scrape") try: data_point =

scrape_data_point() except Exception as e: loguru.logger.error(f"Failed to scrape data point: {e}") data_point = None # Save data if data_point is not None: dem.add_today(data_point) dem.save() loguru.logger.info("Saved daily event monitor") def print_tree(directory, ignore_dirs=[".git", "pycache"]): loguru.logger.info(f"Printing tree of files/dirs at {directory}") for root, dirs, files in os.walk(directory): dirs[:] = [d for d in dirs if d not in ignore_dirs] level = root.replace(directory, "").count(os.sep) indent = " " * 4 * (level) loguru.logger.info(f" {indent}+--{os.path.basename(root)}/") sub_indent = " " * 4 * (level + 1) for file in files: loguru.logger.info(f"{sub_indent}+--{file}") print_tree(os.getcwd()) loguru.logger.info("Printing contents of data file {}".format(dem.file_path)) with open(dem.file_path, "r") as f: loguru.logger.info(f.read()) # Finish loguru.logger.info("Scrape complete") loguru.logger.info("Exiting")"""

Let me break down this code which scrapes headlines from The Daily Pennsylvanian website:

1. Main Purpose:

- The script scrapes the main headline from The Daily Pennsylvanian (Penn's student newspaper) website
- It saves headlines over time in a JSON file
- It includes logging functionality to track the scraping process

2. Key Components:

The `scrape_data_point()` function:

- Makes an HTTP request to thedp.com with a custom user agent
- Uses BeautifulSoup to parse the HTML
- Looks for an element with class "frontpage-link"
- Returns the headline text if found, empty string if not
- Logs the URL, status code, and retrieved data point

The main execution block:

- Sets up daily log rotation using loguru
- Creates a "data" directory if it doesn't exist
- Initializes a DailyEventMonitor object to track headlines
- Runs the scraping function
- Saves the scraped headline if successful
- Prints a directory tree and the contents of the data file
- Logs each step of the process

3. Error Handling:

- Includes try/except blocks around critical operations
- Logs errors when they occur
- Sets data_point to None if scraping fails
- Exits with error code 1 if directory creation fails

4. Utility Functions: `print_tree()`:

- Recursively prints the directory structure
- Ignores certain directories (.git, **pycache**)
- Shows files and directories with indentation

5. Dependencies:

- requests: For making HTTP requests
- bs4 (BeautifulSoup): For parsing HTML
- loguru: For logging
- daily_event_monitor: Custom module for managing the headline data
- os, sys: For system operations

There's a small bug in the code: `if **name**` should be `if __name__`. The correct syntax uses double underscores.

Would you like me to explain any particular part in more detail?

JW

Explain in more detail with the code

I'll break down the code in detail with explanations:

Key Features and Design Choices:

1. **Robust Error Handling**:
   - The script uses try/except blocks around critical operations
   - Failed scraping doesn't crash the script but is logged and handled gracefully
   - Directory creation failures cause immediate exit to prevent data loss
2. **Comprehensive Logging**:
   - Uses loguru for advanced logging capabilities
   - Logs rotate daily to prevent huge log files
   - Every significant operation is logged for debugging
   - Includes request details, scraping results, and file operations

3. **Clean Data Management**:
    - Uses a separate DailyEventMonitor class to handle data storage
    - Saves headlines in a structured JSON format
    - Creates necessary directories automatically
    - Validates data before saving
4. **Web Scraping Best Practices**:
    - Uses a custom User-Agent to identify the scraper
    - Checks response status before parsing
    - Uses proper HTML parsing with BeautifulSoup
    - Handles missing data gracefully
5. **Debugging Features**:
    - Prints directory structure for verification
    - Shows contents of data file after saving
    - Ignores irrelevant directories in tree display

The script appears to be part of a larger system (possibly a class project given the "cis3500" user agent) and is designed for reliability and maintainability. The code follows Python best practices with good documentation, error handling, and logging.

Would you like me to elaborate on any particular aspect of the code?

Detailed Daily Pennsylvanian Scraper Explanation

```
# 1. Imports and Setup import os import sys import daily_event_monitor # Custom module to manage headline data import bs4 # BeautifulSoup for HTML parsing import requests # For making HTTP requests import loguru # For advanced logging # 2. Core Scraping Function def scrape_data_point(): """ Scrapes the main headline from The Daily Pennsylvanian home page. Returns: str: The headline text if found, otherwise an empty string. """ # Define custom user agent to identify our scraper headers = { "User-Agent": "cis3500-scraper" } # Make HTTP request to the website req = requests.get("https://www.thedp.com", headers=headers) # Log request details for debugging loguru.logger.info(f"Request URL: {req.url}") loguru.logger.info(f"Request status code: {req.status_code}") # Process the response if request was successful if req.ok: # Parse HTML content soup =
```

```
bs4.BeautifulSoup(req.text, "html.parser") # Find the headline element
target_element = soup.find("a", class_="frontpage-link") # Extract text or
return empty string if not found data_point = "" if target_element is None
else target_element.text loguru.logger.info(f"Data point: {data_point}")
return data_point # 3. Main Execution Block if __name__ == "__main__": # Only
run if script is executed directly # Setup logging with daily rotation
loguru.logger.add("scrape.log", rotation="1 day") # Create data directory
loguru.logger.info("Creating data directory if it does not exist") try:
os.makedirs("data", exist_ok=True) except Exception as e:
loguru.logger.error(f"Failed to create data directory: {e}") sys.exit(1) #
Exit with error code # Initialize headline tracker loguru.logger.info("Loading
daily event monitor") dem = daily_event_monitor.DailyEventMonitor(
"data/daily_pennsylvanian_headlines.json" ) # Perform scraping
loguru.logger.info("Starting scrape") try: data_point = scrape_data_point()
except Exception as e: loguru.logger.error(f"Failed to scrape data point:
{e}") data_point = None # Save scraped data if successful if data_point is not
None: dem.add_today(data_point) # Add today's headline dem.save() # Save to
JSON file loguru.logger.info("Saved daily event monitor") # 4. Utility
Function for Directory Visualization def print_tree(directory, ignore_dirs=
[".git", "__pycache__"]): """ Prints a tree structure of the given directory,
ignoring specified directories. """ loguru.logger.info(f"Printing tree of
files/dirs at {directory}") for root, dirs, files in os.walk(directory): #
Filter out ignored directories dirs[:] = [d for d in dirs if d not in
ignore_dirs] # Calculate indentation level level = root.replace(directory,
"").count(os.sep) indent = " " * 4 * (level) # Print directory name
loguru.logger.info(f"{indent}+--{os.path.basename(root)}/") # Print files in
directory sub_indent = " " * 4 * (level + 1) for file in files:
loguru.logger.info(f"{sub_indent}+--{file}") # Print directory structure and
file contents print_tree(os.getcwd()) loguru.logger.info("Printing contents of
data file {}".format(dem.file_path)) with open(dem.file_path, "r") as f:
loguru.logger.info(f.read()) # Log completion loguru.logger.info("Scrape
complete") loguru.logger.info("Exiting")
```

Last edited just now