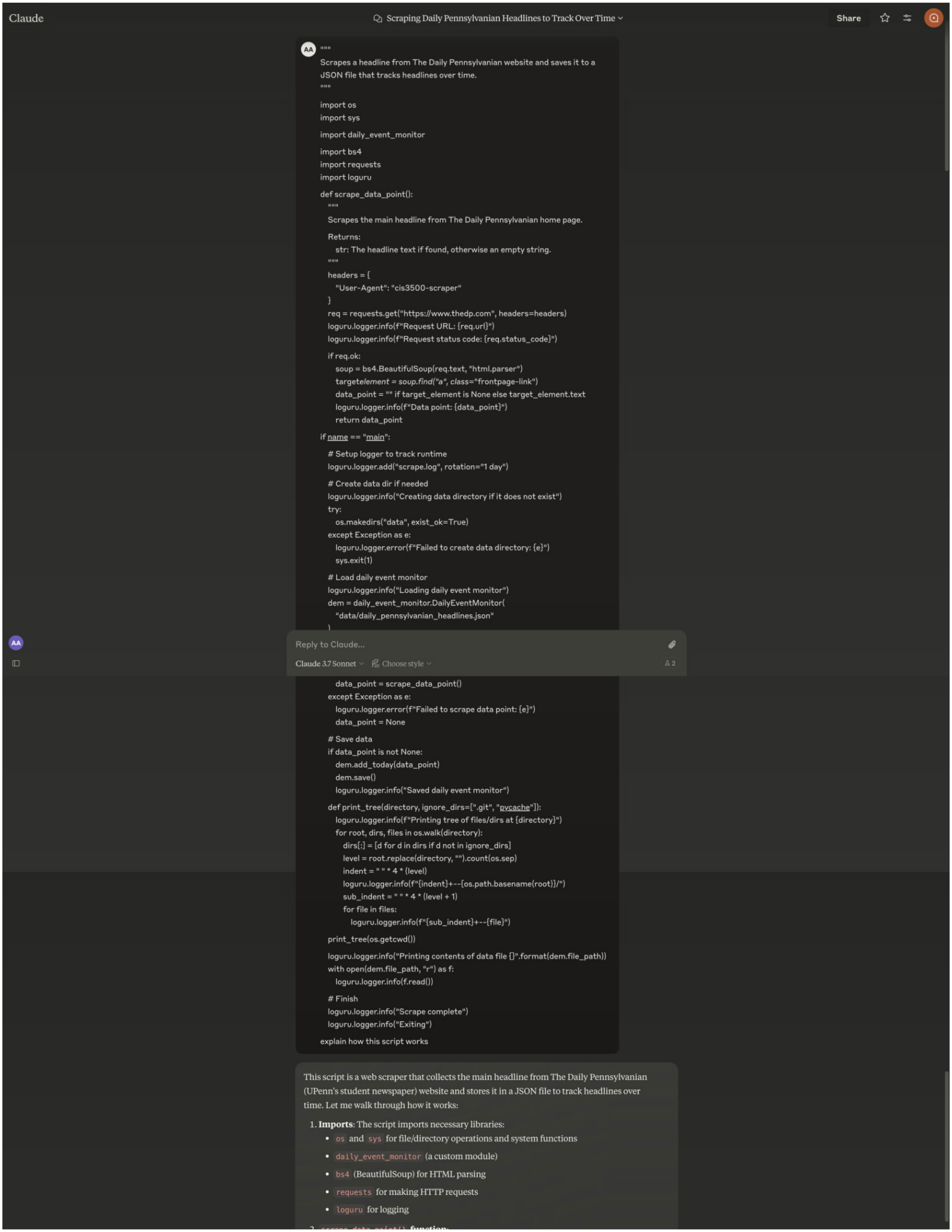


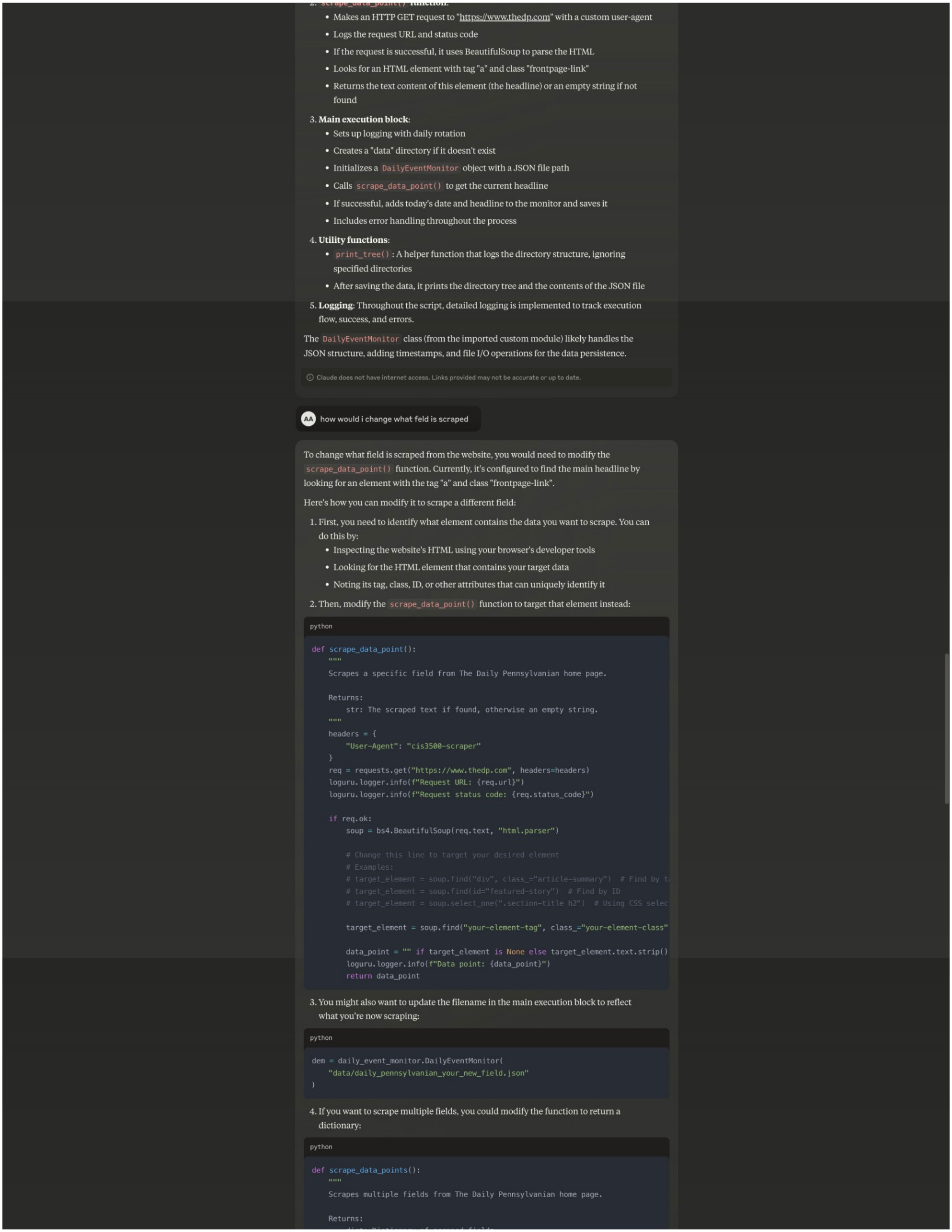
```
"""
Scrapes a headline from The Daily Pennsylvanian website and saves it to a
JSON file that tracks headlines over time.
"""

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")
        targetelement = 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"
    )
```

```
        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")
```

explain how this script works

This script is a web scraper that collects the main headline from The Daily Pennsylvanian (UPenn's student newspaper) website and stores it in a JSON file to track headlines over time. Let me walk through how it works:

1. **Imports**: The script imports necessary libraries:
    - `os` and `sys` for file/directory operations and system functions
    - `daily_event_monitor` (a custom module)
    - `bs4` (BeautifulSoup) for HTML parsing
    - `requests` for making HTTP requests
    - `loguru` for logging
2. `scrape_data_point()` **function**:



2. `scrape_data_point()` **function**:
   - Makes an HTTP GET request to "https://www.thedp.com" with a custom user-agent
   - Logs the request URL and status code
   - If the request is successful, it uses BeautifulSoup to parse the HTML
   - Looks for an HTML element with tag "a" and class "frontpage-link"
   - Returns the text content of this element (the headline) or an empty string if not found
3. **Main execution block**:
   - Sets up logging with daily rotation
   - Creates a "data" directory if it doesn't exist
   - Initializes a `DailyEventMonitor` object with a JSON file path
   - Calls `scrape_data_point()` to get the current headline
   - If successful, adds today's date and headline to the monitor and saves it
   - Includes error handling throughout the process
4. **Utility functions**:
   - `print_tree()`: A helper function that logs the directory structure, ignoring specified directories
   - After saving the data, it prints the directory tree and the contents of the JSON file
5. **Logging**: Throughout the script, detailed logging is implemented to track execution flow, success, and errors.

The `DailyEventMonitor` class (from the imported custom module) likely handles the JSON structure, adding timestamps, and file I/O operations for the data persistence.

ⓘ Claude does not have internet access. Links provided may not be accurate or up to date.

AA how would i change what feld is scraped

To change what field is scraped from the website, you would need to modify the `scrape_data_point()` function. Currently, it's configured to find the main headline by looking for an element with the tag "a" and class "frontpage-link".

Here's how you can modify it to scrape a different field:

1. First, you need to identify what element contains the data you want to scrape. You can do this by:
   - Inspecting the website's HTML using your browser's developer tools
   - Looking for the HTML element that contains your target data
   - Noting its tag, class, ID, or other attributes that can uniquely identify it
2. Then, modify the `scrape_data_point()` function to target that element instead:

python

```python
def scrape_data_point():
    """
    Scrapes a specific field from The Daily Pennsylvanian home page.

    Returns:
        str: The scraped 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")

        # Change this line to target your desired element
        # Examples:
        # target_element = soup.find("div", class_="article-summary")  # Find by t
        # target_element = soup.find(id="featured-story")  # Find by ID
        # target_element = soup.select_one(".section-title h2")  # Using CSS selec

        target_element = soup.find("your-element-tag", class_="your-element-class"

        data_point = "" if target_element is None else target_element.text.strip()
        loguru.logger.info(f"Data point: {data_point}")
        return data_point
```

3. You might also want to update the filename in the main execution block to reflect what you're now scraping:

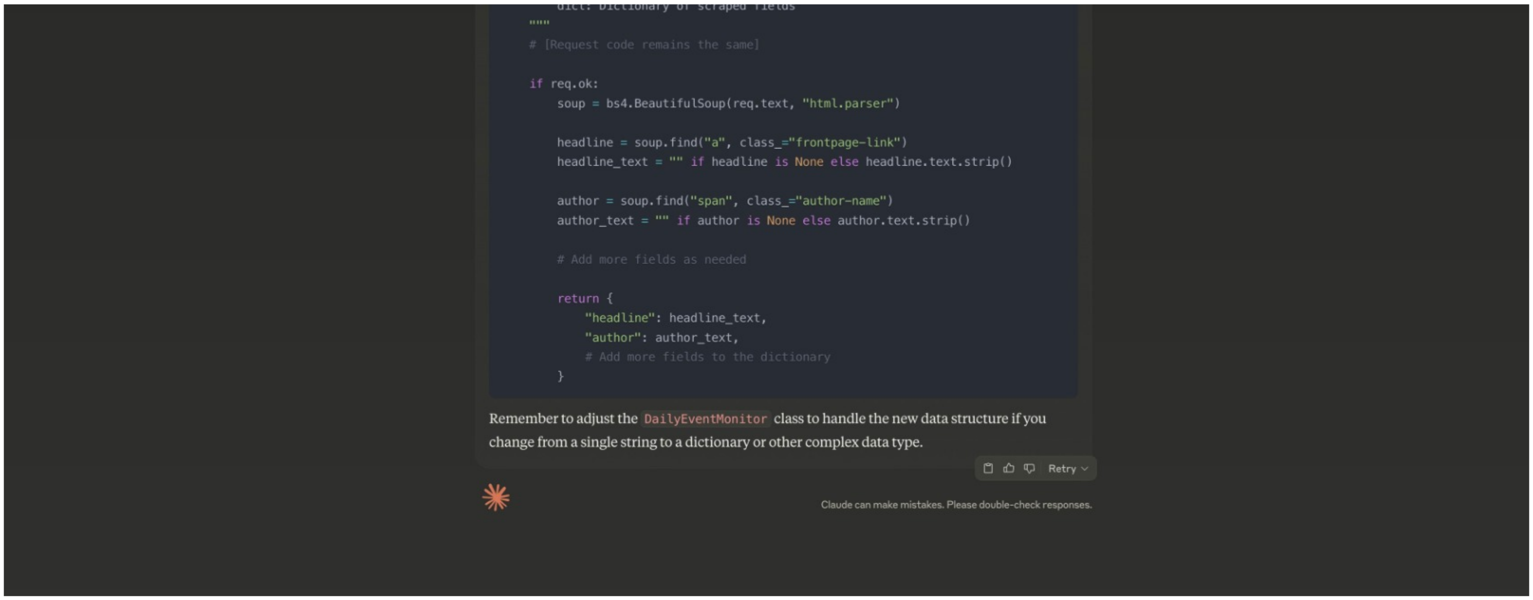
python

```python
dem = daily_event_monitor.DailyEventMonitor(
    "data/daily_pennsylvanian_your_new_field.json"
)
```

4. If you want to scrape multiple fields, you could modify the function to return a dictionary:

python

```python
def scrape_data_points():
    """
    Scrapes multiple fields from The Daily Pennsylvanian home page.

    Returns:
```



```
        dict: Dictionary of scraped fields
    """
    # [Request code remains the same]

    if req.ok:
        soup = bs4.BeautifulSoup(req.text, "html.parser")

        headline = soup.find("a", class_="frontpage-link")
        headline_text = "" if headline is None else headline.text.strip()

        author = soup.find("span", class_="author-name")
        author_text = "" if author is None else author.text.strip()

        # Add more fields as needed

        return {
            "headline": headline_text,
            "author": author_text,
            # Add more fields to the dictionary
        }
```

Remember to adjust the `DailyEventMonitor` class to handle the new data structure if you change from a single string to a dictionary or other complex data type.

Claude can make mistakes. Please double-check responses.