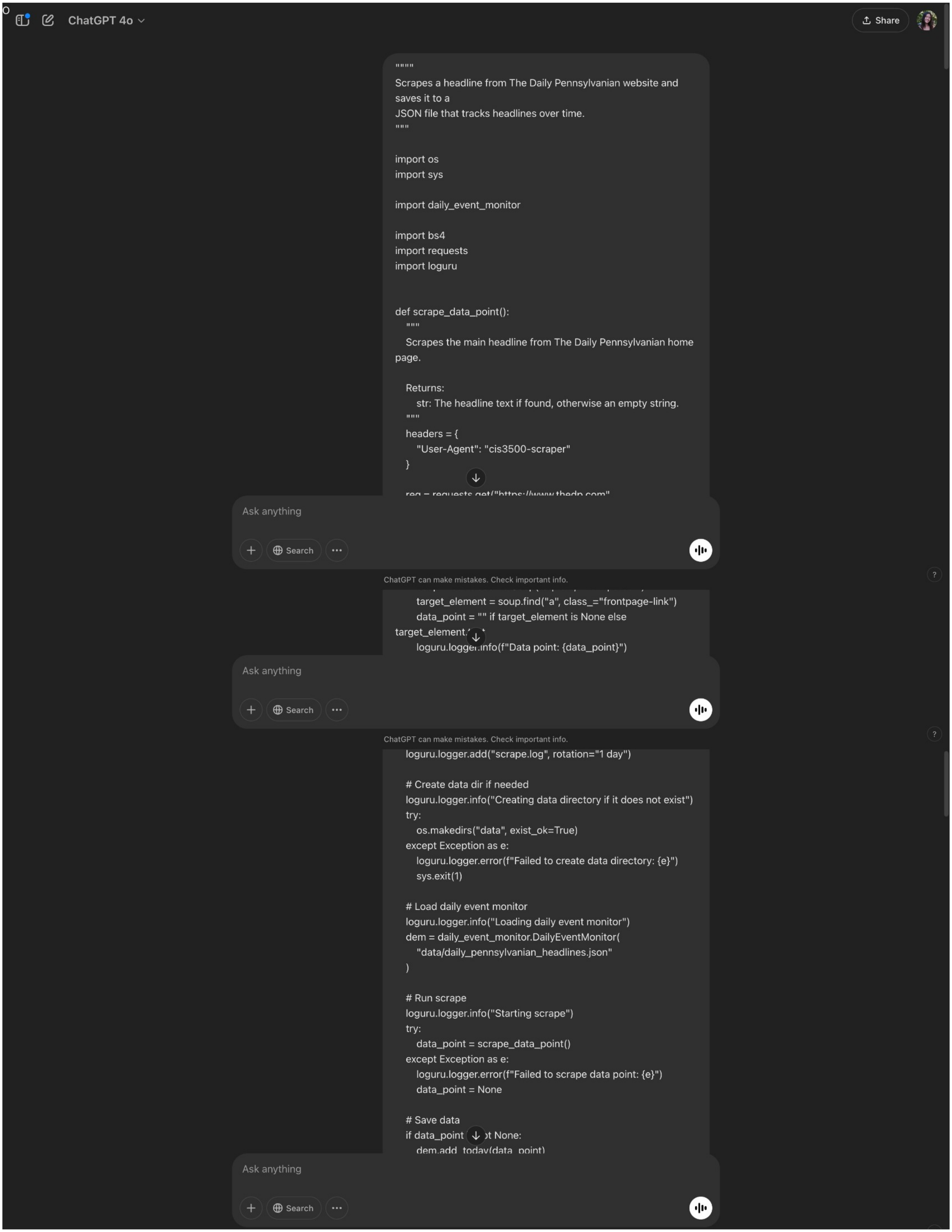


```
"""
Scrapes a headline from The Daily Pennsylvanian website and
saves it to a
JSON file that tracks headlines over time.
"""

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"
```

```
        target_element = soup.find("a", class_="frontpage-link")
        data_point = "" if target_element is None else
target_element.
        loguru.logger.info(f"Data point: {data_point}")
```

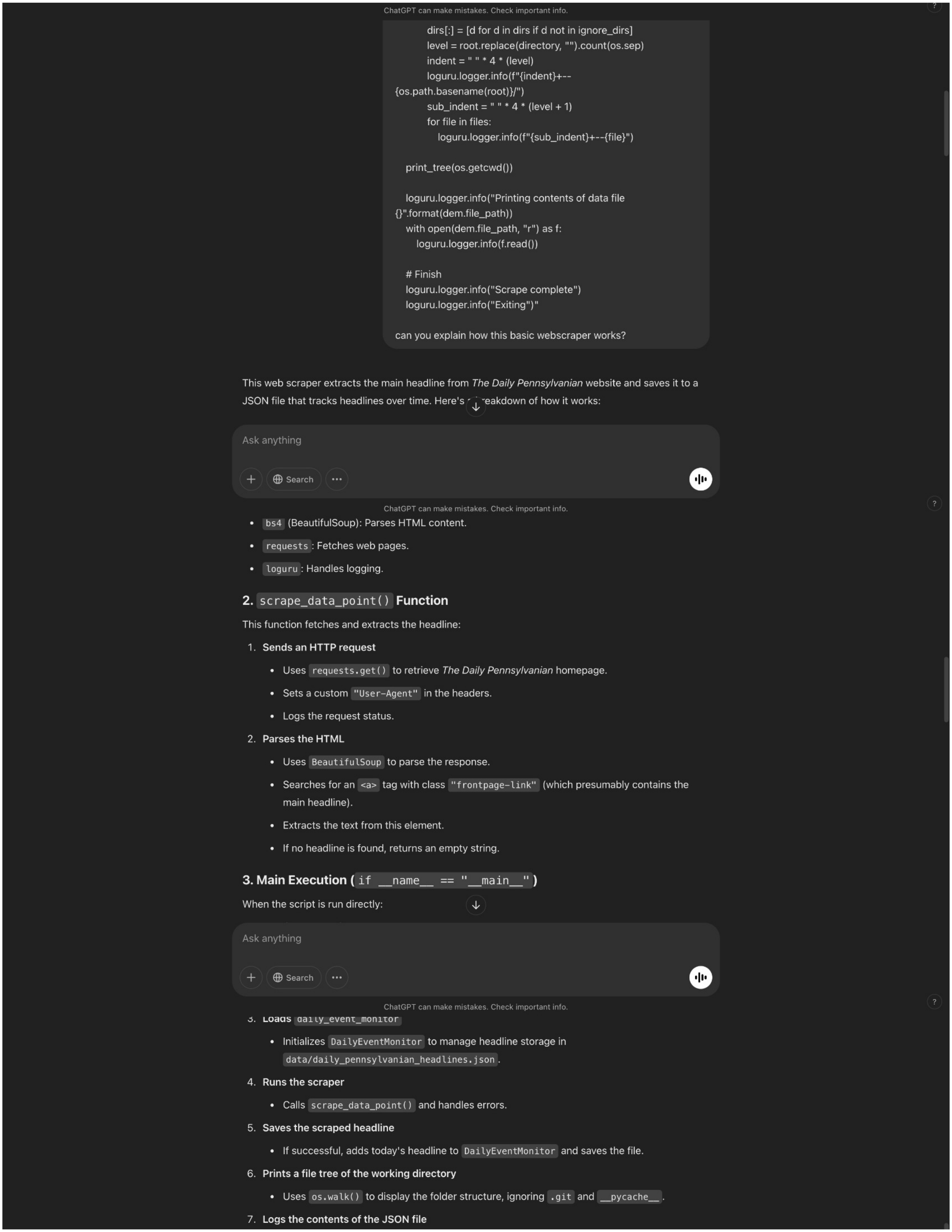
```
    loguru.logger.add("scrape.log", rotation="1 day")

    # 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   t None:
        dem.add_today(data_point)
```



```
            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")"
```

can you explain how this basic webscraper works?

This web scraper extracts the main headline from *The Daily Pennsylvanian* website and saves it to a JSON file that tracks headlines over time. Here's a breakdown of how it works:

- `bs4` (BeautifulSoup): Parses HTML content.
- `requests`: Fetches web pages.
- `loguru`: Handles logging.

2. `scrape_data_point()` Function

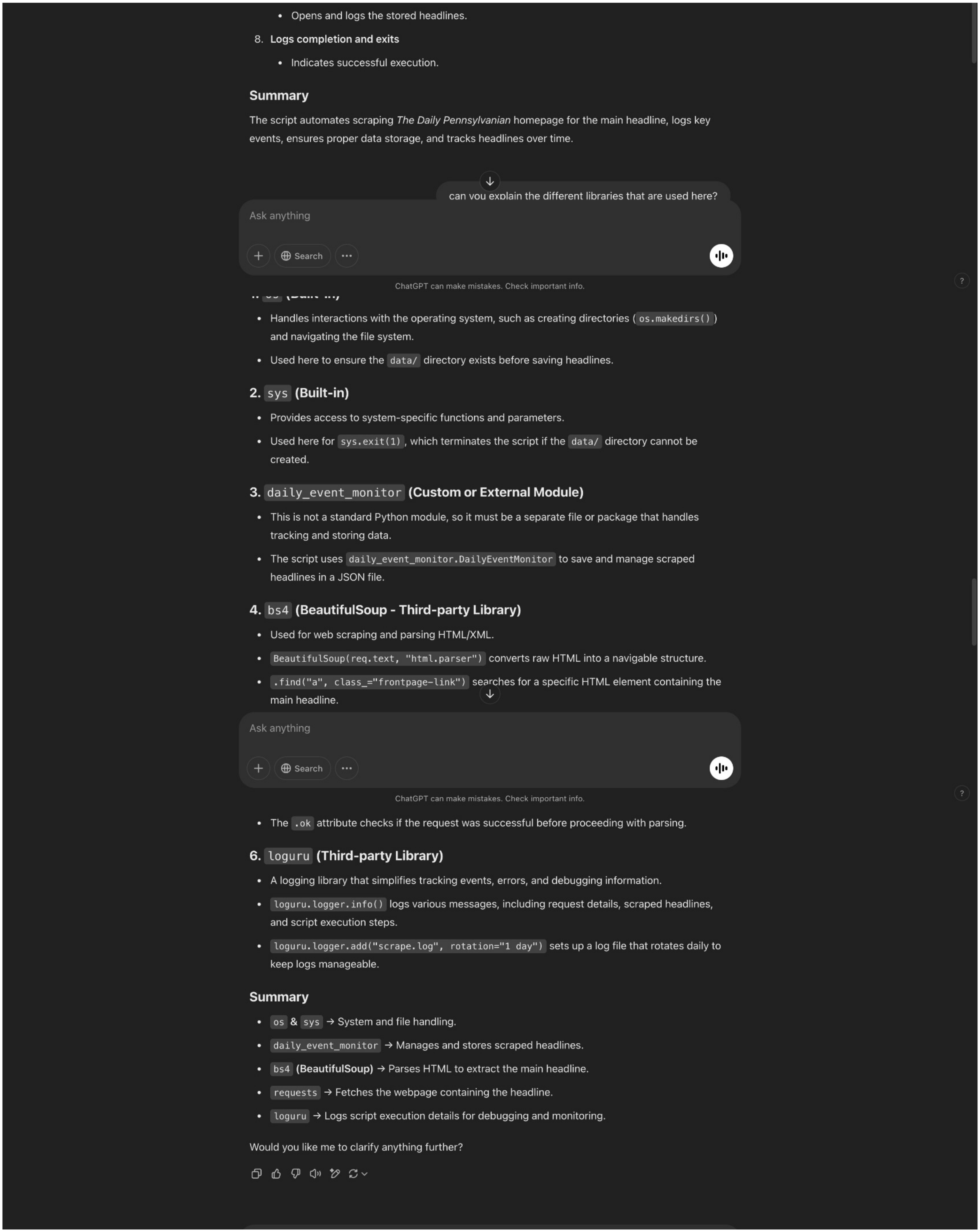
This function fetches and extracts the headline:

1. **Sends an HTTP request**
    - Uses `requests.get()` to retrieve *The Daily Pennsylvanian* homepage.
    - Sets a custom `"User-Agent"` in the headers.
    - Logs the request status.
2. **Parses the HTML**
    - Uses `BeautifulSoup` to parse the response.
    - Searches for an `<a>` tag with class `"frontpage-link"` (which presumably contains the main headline).
    - Extracts the text from this element.
    - If no headline is found, returns an empty string.

3. Main Execution (`if __name__ == "__main__"`)

When the script is run directly:

3. **Loads** `daily_event_monitor`
    - Initializes `DailyEventMonitor` to manage headline storage in `data/daily_pennsylvanian_headlines.json`.
4. **Runs the scraper**
    - Calls `scrape_data_point()` and handles errors.
5. **Saves the scraped headline**
    - If successful, adds today's headline to `DailyEventMonitor` and saves the file.
6. **Prints a file tree of the working directory**
    - Uses `os.walk()` to display the folder structure, ignoring `.git` and `__pycache__`.
7. **Logs the contents of the JSON file**



- Opens and logs the stored headlines.

8. **Logs completion and exits**
   - Indicates successful execution.

Summary

The script automates scraping *The Daily Pennsylvanian* homepage for the main headline, logs key events, ensures proper data storage, and tracks headlines over time.

can you explain the different libraries that are used here?

- Handles interactions with the operating system, such as creating directories (`os.makedirs()`) and navigating the file system.
- Used here to ensure the `data/` directory exists before saving headlines.

2. `sys` (Built-in)

- Provides access to system-specific functions and parameters.
- Used here for `sys.exit(1)`, which terminates the script if the `data/` directory cannot be created.

3. `daily_event_monitor` (Custom or External Module)

- This is not a standard Python module, so it must be a separate file or package that handles tracking and storing data.
- The script uses `daily_event_monitor.DailyEventMonitor` to save and manage scraped headlines in a JSON file.

4. `bs4` (BeautifulSoup - Third-party Library)

- Used for web scraping and parsing HTML/XML.
- `BeautifulSoup(req.text, "html.parser")` converts raw HTML into a navigable structure.
- `.find("a", class_="frontpage-link")` searches for a specific HTML element containing the main headline.

- The `.ok` attribute checks if the request was successful before proceeding with parsing.

6. `loguru` (Third-party Library)

- A logging library that simplifies tracking events, errors, and debugging information.
- `loguru.logger.info()` logs various messages, including request details, scraped headlines, and script execution steps.
- `loguru.logger.add("scrape.log", rotation="1 day")` sets up a log file that rotates daily to keep logs manageable.

Summary

- `os` & `sys` → System and file handling.
- `daily_event_monitor` → Manages and stores scraped headlines.
- `bs4` **(BeautifulSoup)** → Parses HTML to extract the main headline.
- `requests` → Fetches the webpage containing the headline.
- `loguru` → Logs script execution details for debugging and monitoring.

Would you like me to clarify anything further?