```
"""
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.
    """
    req = requests.get("https://www.thedp.com")
    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
```

Reply to Claude...

Claude 3.5 Sonnet Choose style

```
# 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")
```

explain this line by line

I'll explain this script line by line. It's a web scraper designed to collect headlines from The Daily Pennsylvanian website and track them over time.

python

```
import os
import sys
```

- Imports standard Python modules for operating system functions (`os`) and system-specific parameters/functions (`sys`).

python

```
import daily_event_monitor
```

- Imports a custom module called `daily_event_monitor` that likely handles tracking events over time.

python

```
import bs4
import requests
import loguru
```

- Imports external libraries: `bs4` (BeautifulSoup) for HTML parsing, `requests` for HTTP requests, and `loguru` for logging.

python

```
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.
    """
```

- Defines a function called `scrape_data_point` with a docstring explaining its purpose.

python

```
    req = requests.get("https://www.thedp.com")
```

- Makes an HTTP GET request to The Daily Pennsylvanian website.

python

```
    loguru.logger.info(f"Request URL: {req.url}")
    loguru.logger.info(f"Request status code: {req.status_code}")
```

- Logs the URL that was requested and the HTTP status code that was returned.

python

```
    if req.ok:
```

- Checks if the request was successful (status code 200-299).

python

```
        soup = bs4.BeautifulSoup(req.text, "html.parser")
```

- Creates a BeautifulSoup object to parse the HTML content of the page.

python

```
        target_element = soup.find("a", class_="frontpage-link")
```

- Looks for the first anchor (`<a>`) element with class="frontpage-link", which likely contains the headline.

```
python
    except Exception as e:
        loguru.logger.error(f"Failed to create data directory: {e}")
        sys.exit(1)
```

- Creates a "data" directory if it doesn't exist already. If it fails, logs the error and exits with code 1.

```
python
    # Load daily event monitor
    loguru.logger.info("Loading daily event monitor")
    dem = daily_event_monitor.DailyEventMonitor(
        "data/daily_pennsylvanian_headlines.json"
    )
```

- Creates an instance of `DailyEventMonitor` from the custom module, specifying the JSON file to store headlines.

```
python
    # 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
```

- Calls the scraping function within a try/except block to handle errors gracefully.

```
python
    except Exception as e:
        loguru.logger.error(f"Failed to create data directory: {e}")
        sys.exit(1)
```

- Creates a "data" directory if it doesn't exist already. If it fails, logs the error and exits with code 1.

```
python
    # Load daily event monitor
    loguru.logger.info("Loading daily event monitor")
    dem = daily_event_monitor.DailyEventMonitor(
        "data/daily_pennsylvanian_headlines.json"
    )
```

- Creates an instance of `DailyEventMonitor` from the custom module, specifying the JSON file to store headlines.

```
python
    # 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
```

- Calls the scraping function within a try/except block to handle errors gracefully.

```
python
    # Save data
    if data_point is not None:
        dem.add_today(data_point)
        dem.save()
        loguru.logger.info("Saved daily event monitor")
```

- If a headline was successfully scraped, adds today's date with the headline to the monitor and saves it.

python

```
def print_tree(directory, ignore_dirs=[".git", "__pycache__"]):
```

- Defines a nested function to print the directory structure, ignoring certain directories.

python

```
    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}")
```

- Walks through all files/directories, calculating the proper indentation to create a tree view, and logs everything.

python

```
print_tree(os.getcwd())
```

- Calls the function to print the current working directory structure.

python

```
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())
```

- Opens and reads the JSON file containing the headlines, then logs its contents.

python

```
# Finish
loguru.logger.info("Scrape complete")
loguru.logger.info("Exiting")
```

- Logs completion messages before the script finishes execution.

This script is designed to run regularly (probably as a scheduled task) to build up a historical archive of headlines from The Daily Pennsylvanian.