News Article Upvote Tracker

# A BeautifulSoup-Based Web Scraping and Analysis Tool

Vishvam Ganesh

Individual Project

Python Web Scraping Experiment

## ABSTRACT:

The News Article Upvote Tracker is a Python-based application that leverages BeautifulSoup and requests to scrape data from the Hacker News homepage. It extracts article titles, URLs, and upvote counts, identifies the article with the highest upvotes, and presents it along with a summary of all collected articles. This tool is designed for users interested in analyzing trending news articles based on community engagement without relying on external APIs or authentication.

## INTRODUCTION:

Hacker News is a popular platform where users share news articles, and community upvotes indicate the relevance and importance of posts. Given the fast-paced nature of news consumption, users benefit from tools that help identify popular stories quickly.

Manual browsing is inefficient when trying to find the most upvoted or engaging articles in real time. Existing solutions may rely on APIs, which have rate limits or require authentication keys. This project introduces a lightweight solution to scrape article data directly from the Hacker News webpage using Python's BeautifulSoup library and requests module.

The News Article Upvote Tracker automatically extracts article information and highlights the most upvoted story, enabling users to quickly identify trending content.

## EXISTING METHODS:

Currently, real-time news tracking is achieved using the following approaches:

### 1. Manual Browsing:

• Users visit news websites like Hacker News and browse articles manually.

• Inefficient for discovering trending topics rapidly.

### 2. API-Based Solutions:

• Some platforms offer APIs to retrieve articles and engagement metrics.

• API rate limits and authentication requirements restrict usability.

### 3. Third-Party News Aggregators:

• Tools like Feedly, Google News, and others offer curated news feeds.

• Lack customization and may not provide direct access to raw upvote data.

### Limitations:

• Manual processes are slow and error-prone.  
• API restrictions limit access frequency and require credentials.  
• Third-party tools may not expose detailed metrics like upvotes.

The News Article Upvote Tracker addresses these limitations by scraping the live webpage content dynamically without dependency on APIs or subscriptions.

## PROPOSED SOLUTION:

Python-Based Web Scraping System

### Key Features:

1. Automated Extraction:  
 Retrieves article titles, URLs, and upvotes from the web page.

2. Data Processing:  
 Identifies the article with the highest upvotes.

3. Simple Implementation:  
 Requires no API keys or complex setup.

4. Data Presentation:  
 Prints the most popular article along with all scraped article details.

## TECHNOLOGIES USED:

### 1. Programming Language:

• Python – chosen for its readability, extensive libraries, and ease of use in web scraping.

### 2. Libraries:

• requests: Fetches webpage content.

• BeautifulSoup (bs4): Parses HTML and extracts structured data.

## METHODS:

1. Webpage Retrieval:  
 The Hacker News homepage is accessed using the requests library.

2. HTML Parsing:  
 BeautifulSoup parses the HTML content to extract article links and titles.

3. Data Extraction:  
 Article titles and URLs are extracted by locating anchor tags with the class 'storylink'.  
 Upvote counts are extracted from 'span' elements with the class 'score'.

4. Data Processing:  
 Upvote counts are converted from text to integers.  
 The article with the highest upvotes is determined using Python’s max() and index() functions.

5. Output Presentation:  
 The article with the most upvotes is printed, followed by lists of all articles, their URLs, and upvote counts for reference.

## CODE SNIPPET:

from bs4 import BeautifulSoup  
import requests  
  
response = requests.get("https://appbrewery.github.io/news.ycombinator.com/")  
yc\_web\_page = response.text  
  
soup = BeautifulSoup(yc\_web\_page, "html.parser")  
articles = soup.find\_all(name="a", class\_="storylink")  
article\_texts = []  
article\_links = []  
for article\_tag in articles:  
 text = article\_tag.getText()  
 article\_texts.append(text)  
 link = article\_tag.get("href")  
 article\_links.append(link)  
 article\_upvotes = [int(score.getText().split()[0]) for score in soup.find\_all(name="span", class\_="score")]  
  
largest\_number = max(article\_upvotes)  
largest\_index = article\_upvotes.index(largest\_number)  
  
print(article\_texts[largest\_index])  
print(article\_links[largest\_index])  
  
print(article\_texts)  
print(article\_links)  
print(article\_upvotes)

## OUTPUT:

Upon execution, the tool outputs:  
- The article with the highest number of upvotes.  
- The corresponding article link.  
- Lists of all article titles, links, and upvote counts for reference.

## LIMITATIONS:

• Dependent on the HTML structure of the webpage; changes may break the scraper.  
• Does not handle JavaScript-rendered content; suitable only for static pages.  
• No error handling for missing or incomplete data.  
• Extracts upvotes only if they are present; articles with no upvotes are ignored.

## FUTURE ENHANCEMENTS:

1. Error Handling:  
 Implement checks for missing elements or network errors.

2. Support for JavaScript Pages:  
 Integrate with Selenium or Playwright to handle dynamic content.

3. Data Storage:  
 Save scraped data into CSV or a database for historical tracking.

4. User Interface:  
 Build a simple GUI to allow users to scrape and filter articles without modifying the code.

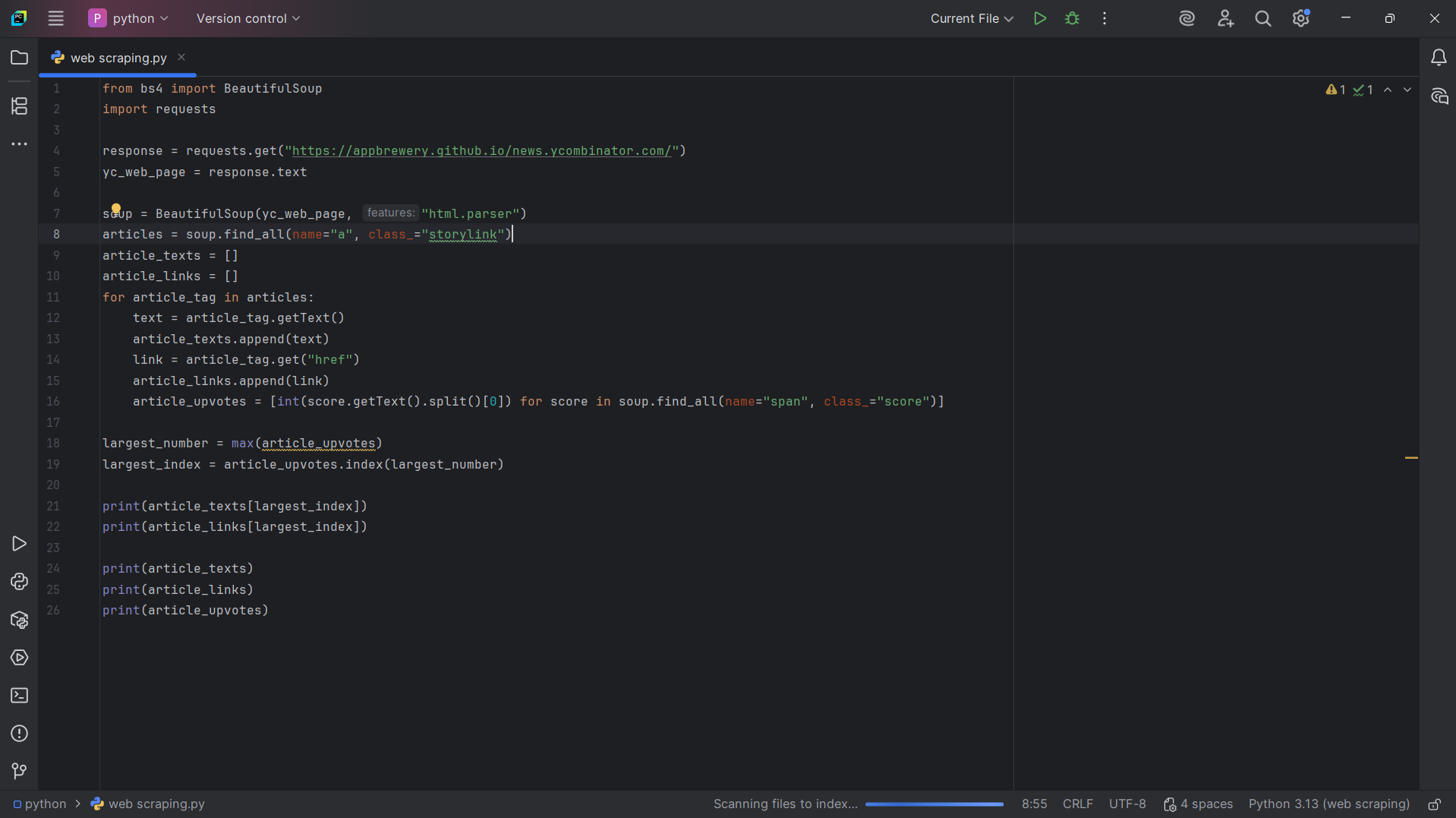
5. Scheduled Execution:  
 Automate periodic scraping using task schedulers.

6. Advanced Filtering:  
 Add options to filter by keywords, time, or user-defined thresholds.

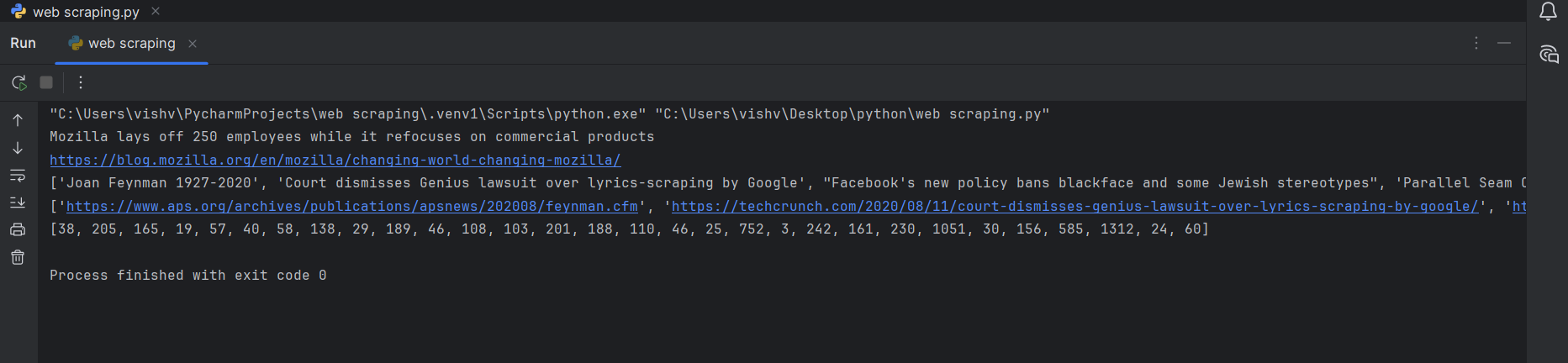
## CONCLUSION:

The News Article Upvote Tracker demonstrates how Python’s BeautifulSoup library can be used for lightweight, efficient web scraping of popular news articles. By identifying the most upvoted content, it provides users with actionable insights in real time without relying on third-party APIs. Though simple in scope, this tool offers a foundational structure that can be extended with error handling, dynamic content scraping, and data visualization for more robust applications.

**Screenshots of code snippets:**

****

**Output:**

****

