Hello!

My name’s Dan, and in this video, I’ll be walking you through our Python Information Aggregator assignment.

This project is all about building a flexible Python application that can fetch, combine, and display news data using both a public API extraction and html web scraping. What makes it interesting isn’t just the data but how we built it. This tool was developed using object-oriented programming principles, and it features a graphical user interface to make it easier and more engaging to use.

Now, to be honest, some parts of the code may not be the most elegant or minimal. But that’s by design. The assignment asks us to incorporate encapsulation, inheritance, and polymorphism, so instead of coding the shortest and most efficient solution, we focused on clear design that demonstrates these concepts in action.

Let’s begin by looking at the design.

At the core of our app is the KeyList class. This class manages a list of 32-character API keys. It’s simple but functional, with a method to return a key at random when needed.

Then, we extend that with the NewsAPI\_KeyList subclass. This is where things get interesting. It overrides the get\_key method to randomly select a key, check this key’s validity with the NewsAPI site before returning it. If the tested key is not valid it will select another key. So here we demonstrate both inheritance and polymorphism in a clean and functional way.

Now, moving on to testing. Some functions in this application are intentionally simple. Why is this? Because testing graphical user interface functions which use tkinter, and functions which perform API class or web scraping can be challenging. So, we started by writing clean, modular functions that could be unit tested easily. This helped us verify unit testing functionality before proceeding to more complicated GUI and external system testing.

As for the information sources, the application pulls news from two locations. First is the NewsAPI, which returns structured JSON data. The second is a direct web scrape of the ABC News Australia website. We chose the ABC website because it’s not behind a paywall, making it ideal for ethical and consistent scraping.

Our team also implemented unit tests on various components to ensure robustness. These include data validation, key handling, and article formatting. It was all about making sure the code could handle real-world usage.

We used several external libraries too. Like pycountry, which helps with country and language lookups, pandas is great for managing our dataset of articles, BeautifulSoup is used for HTML parsing, and matplotlib handles the visualization side of things.

So, let’s get to the fun bit, the graphical user interface.

When you first launch the application, it automatically fetches a valid API key and then retrieves the latest list of news sources, categories, and countries from the NewsAPI. These are then displayed in listboxes for the user to choose from.

Now, the main window is divided into several parts:

On the top right, we have an API option selector using radio buttons. The user can choose to search by either news source or a combination of category and country.

Below that, there are buttons to fetch or clear news articles via the NewsAPI or via ABC News scraping.

To the top right, there’s an “Extracted News Stories” display panel, which shows each article's title, source, date, author (if available), description, extraction method, and a URL, which can be copied and pasted into a browser.

On the bottom right, we’ve got a keyword frequency graph that shows the 20 most common words from the article titles.

A few usability features worth mentioning:

When you select a category and a country, for example “Technology” and “United States,” and click “Get API News,” the articles are pulled from NewsAPI and shown in the “Extracted News Stories” panel. The “Common Title Words” plot is also generated.

The button will then toggle to a “Clear API News”, which, when clicked, either clears the articles dataset or removes only the NewsAPI articles, leaving any scraped ABC articles untouched.

Similarly, the “Get ABC News” button scrapes the ABC News homepage and either creates a new dataset or adds the articles to the current dataset. These can also be removed independently with the “Clear ABC News” button.

The information summary panel tells the user how many total articles are in the dataset, which source contributed the most, and shows the publication date range.

And the omitted words text box? That lets you exclude common English words from the keyword graph. Words like “the,” “and,” “is,” etc. You can add more unwanted words to this list and click “Replot” to update the graph dynamically. It’s simple and effective.

Now, from an assessment perspective, we were required to:

Provide an object-oriented Python codebase;

Include unit tests;

Submit a README with required libraries and usage instructions;

And write a short report on our design, challenges, and any bonus features.  
  
And this Video.

One thing we were very mindful of is ethical scraping. When scraping data, always follow the site’s terms and conditions, avoid overloading servers. You must never collect sensitive or personal information, and always credit your sources.   
  
In our case, both sources allow basic public scraping for headlines, which made it a suitable choice.

And that wraps up our tour of the Python Information Aggregator project!

Thanks so much for watching. I hope this has given you a clear picture of the design and functionality behind the application. If you’ve got any questions or want to dive deeper into a particular part of the project, feel free to reach out via email or the subject discussion board. I’ll be happy to help.

Until next time