Thought Process and Approach for Google AdSense Checker

Initial Considerations

The first thing I thought about was how to spot the signs of Google AdSense on a website. Google AdSense leaves multiple indications in the site's source code, which can be used to identify its presence. These indicators include:

- `adsbygoogle.js`

- `googlesyndication.com`

- Specific `<div>` elements with class names related to ads

Identifying Patterns and Indications

After understanding how to spot these indicators, I planned to create a scraper function to detect and collect the data needed to classify a site as having Google AdSense.

Challenges and Dynamic Detection

I realized that scraping the HTML alone might not always work because a site can support Google AdSense without showing ads all the time. To reliably detect ads, I needed a way to do it dynamically and in real time.

Solution with Selenium

That's when I came across Selenium, a tool for web testing that was perfect for this. Selenium allows for real-time interaction with web pages, making it possible to detect dynamically loaded ads. Using iframe detection works for many Google AdSense ads since they are often served within iframe elements.

Developing the Code

Here's a breakdown of how I developed the code:

1. \*\*Preparation: I started by setting up Selenium WebDriver and pandas for handling input and output in CSV format. The CSV files included one for the list of websites and another for the results.

2. \*\*Data Collection: I used an API to fetch a bulk of websites and append them to the websites file.

3. Ad Detection: The main part of the code involves detecting ads. I created the `check\_adsense\_ads` function to initialize the Edge WebDriver, navigate to a URL, and wait up to 30 seconds for an iframe containing Google ads to load. If such an iframe is found, the function returns True, indicating that AdSense ads are present.

4. Running the Script: The `main` function reads URLs from the input CSV file, checks each URL for ads, and writes the results to an output CSV file. By using `WebDriverWait` with `EC.frame\_to\_be\_available\_and\_switch\_to\_it`, the script can wait for dynamically loaded ads, making it robust for websites where ads load after the main content.

5. Execution: The script uses the standard Python idiom (`if \_\_name\_\_ == "\_\_main\_\_":`) to ensure the `main` function runs only when the script is executed directly, not when imported as a module.

This setup ensures that websites are accurately checked for AdSense ads, with results systematically logged in a CSV file for further analysis.