Script Overview:

The script is designed to scrape public notices from the website "georgiapublicnotice.com," specifically for the state of Georgia.

Dependencies:

Selenium: This script relies on the Selenium library for web automation.

MySQL: It uses a MySQL database to store the scraped data.

Anti-Captcha: Integrate with an Anti-Captcha service to solve CAPTCHA challenges.

Various Python standard libraries and utilities for logging and data manipulation.

Key Functions and Sections:

Initialization and Setup:

The script starts by importing necessary libraries and setting up options for the Selenium WebDriver.

It also sets up logging and initializes some global variables.

Helper Functions:

print_log(text, error=False): Conditional logging function for debugging.

time_elapsed_str(start, end): Calculates and returns the elapsed time in a readable format.

wait_time(): Generates a random wait time interval.

init_driver(): Initializes the Selenium WebDriver.

Main Functions:

wait_loader(driver): Waits for a loading spinner to disappear on the webpage.

count_total_records(driver): Counts the total number of records on the website.

get_all_pages(link, driver, limited, database, source, env): Scrapes data from multiple pages.

get_data(driver, state_name): Extracts data from a specific page.

main(param): The main function that orchestrates the scraping process.

Execution:

The script is set to execute when it's run directly (if __name__ == '__main__':).

It takes parameters such as the scraping limit, environment settings, and production mode.

How to Use:

Install the required Python libraries, including Selenium and MySQL connector.

Make sure you have a MySQL database set up and configure the script to connect to it.

Provide the necessary Anti-Captcha API key if you intend to use CAPTCHA solving.

Adjust the scraping parameters such as the limit and environment settings in the args dictionary.

Run the script.

Note:

The script seems to be designed for a specific website and use case (scraping public notices for Georgia).

It includes error handling, logging, and database integration.

Ensure you have the required dependencies and correct database configuration before running it.

Additional documentation or comments within the script could provide more context and usage information.

The script the calls the below script (popstream.py)

Automated Data Scraping and Analysis Script

Introduction:

This Python script (popstream.py) is designed to automate the process of scraping property-related data from gapubs publications, such as PropStream and TruthFinder, and store it in a database. The script uses Selenium for web automation and interacts with both PropStream and TruthFinder websites. The scraped data is then saved to both a SQLite database and a MySQL database.

Dependencies:

undetected_chromedriver: A headless Chrome driver for web automation.

mysql.connector: A MySQL driver for Python.

requests: A library for making HTTP requests.

Selenium: A web automation library.

sqlite3: A built-in Python library for SQLite database operations. (Optional – for backup reasons)

Main Components:

Setting Up Dependencies:

Proxy: Set your proxy settings.

Imports necessary libraries and sets up options for Chrome driver.

Scrapping Properties:

Gapubs (gapubs.py) script runs and scrapes all property addresses and after scrapping the details it calls the popstream script to start.

Login to Various Websites:

Automates the login process for PropStream and TruthFinder websites.

Data Scraping from PropStream:

Searches for property information on PropStream using an address.

Retrieves various property details and stores them in a dictionary.

Data Scraping from TruthFinder:

Searches for resident information on TruthFinder using property details.

Retrieves contact information (phone numbers, emails) and stores it in the dictionary.

Saving Data:

Saves the scraped data to both SQLite and MySQL databases.

Uses SQL queries to insert data into the respective databases.

Fetching Existing IDs:

Retrieves existing property IDs from the SQLite database to avoid duplicate entries.

Iterating Through Data:

Iterates through a list of property data.

Checks if the property ID is not already in the database to avoid duplicates.

Running the script (python3 popstream.py)

Usage:

The script can be executed to automate the data scraping and analysis process for multiple properties.

The main function main() handles the entire process.

Notes:

This documentation provides a high-level overview of the code's functionality. Further details and customization may be required based on specific use cases and requirements. Currently the script performs all steps mentioned above in the order.

To run the whole script. Simply run python3 gapubs.py (scrap property addresses (gapubs.py) > scrap resident’s contact details (propstream.py) > save to mysql)