**Web Scraping & Security Report**

Website Scraped: <http://quotes.toscrape.com>/

Security Approach: Option B: Data Masking

**1. Description of the Website Scraped**

The website <http://quotes.toscrape.com>/ was scraped. This is a public sandbox site for scraping practice, featuring quotes, authors, and tags across ten pages.

**2. Fields Extracted**

Quote: The full text of the quote.

Author: The name of the person who said the quote.

Tags: A comma-separated list of keywords for the quote.

**3. Security Approach Used**

The security measure implemented was Option B: Data Masking to protect Personally Identifiable Information (PII). A new column, masked\_author, was created by applying a function to the original author's name.

This function keeps the first letter and replaces all other characters with an asterisk (\*).

Example:

Original Author: Albert Einstein

Masked Author: A\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The final DataFrame, including this masked column, was saved to quotes\_masked.csv.

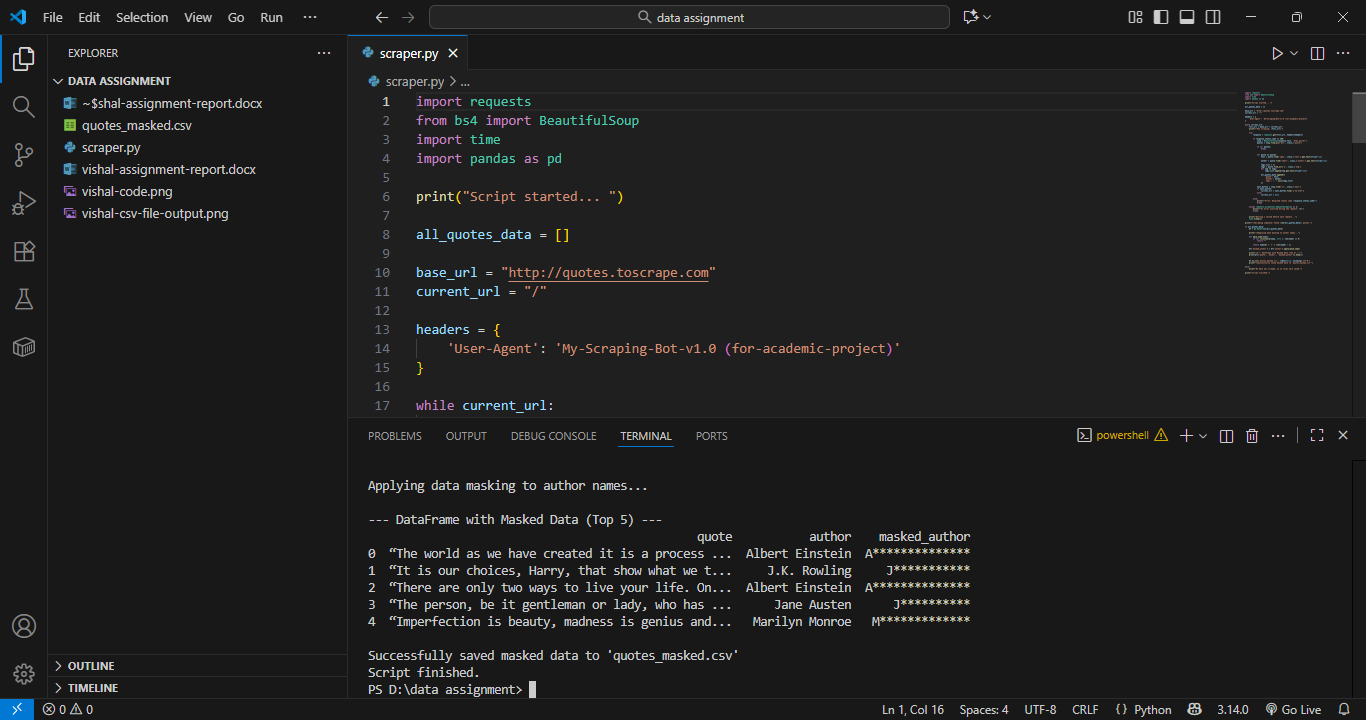
**4. Challenges Faced**

Handling Pagination: The scraper was designed with a while loop to find the "Next" page button's link and loop through all ten pages.

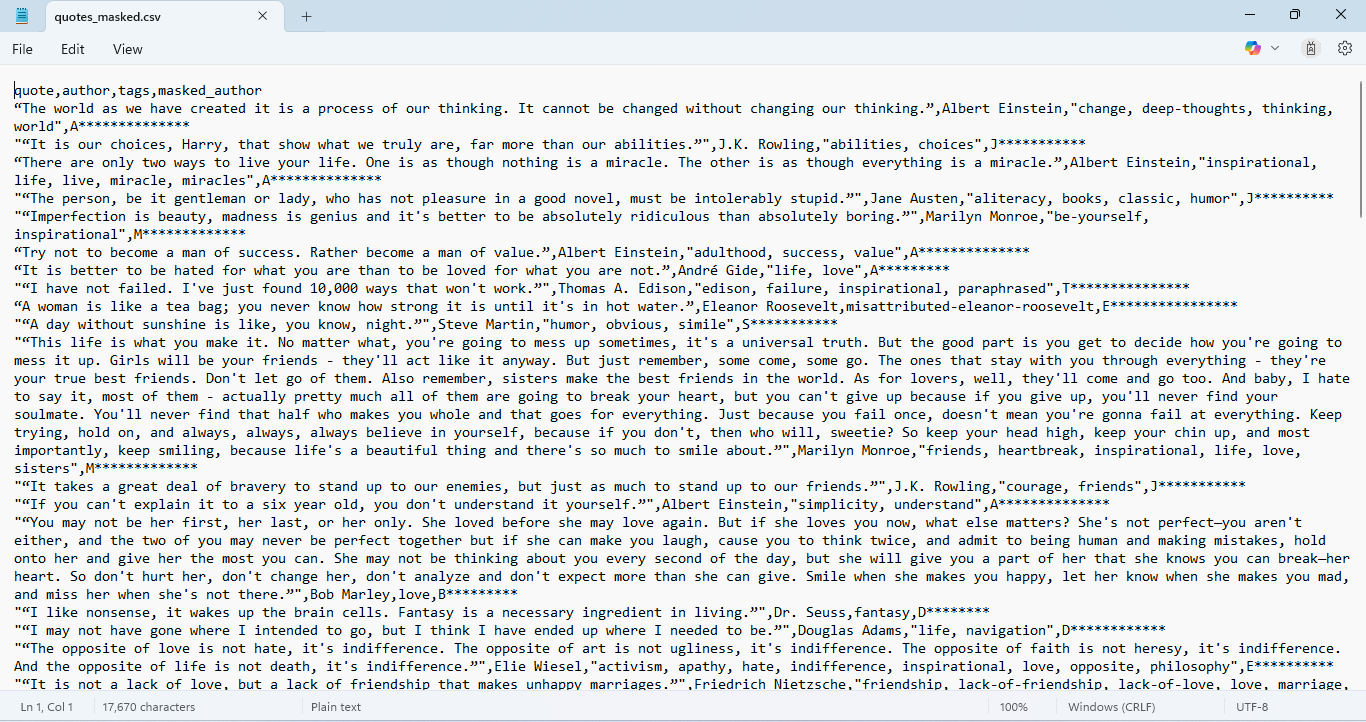
Ethical Scraping: To avoid overwhelming the server, a time.sleep(1) delay was added between page requests, and a User-Agent header was set to identify the script.

**5. Screenshots and Code Snippets**

**Screenshot 1**: Script Execution in Terminal



**Screenshot 2:** Final quotes\_masked.csv File



**Code Snippet**:

import requests

from bs4 import BeautifulSoup

import time

import pandas as pd

print("Script started... ")

all\_quotes\_data = []

base\_url = "http://quotes.toscrape.com"

current\_url = "/"

headers = {

    'User-Agent': 'My-Scraping-Bot-v1.0 (for-academic-project)'

}

while current\_url:

    full\_url = base\_url + current\_url

    print(f"Now scraping: {full\_url}")

    try:

        response = requests.get(full\_url, headers=headers)

        if response.status\_code == 200:

            soup = BeautifulSoup(response.text, 'html.parser')

            quotes = soup.find\_all('div', class\_='quote')

            if not quotes:

                break

            for quote in quotes:

                text = quote.find('span', class\_='text').get\_text(strip=True)

                author = quote.find('small', class\_='author').get\_text(strip=True)

                tags\_list = []

                tags = quote.find\_all('a', class\_='tag')

                for tag in tags:

                    tags\_list.append(tag.get\_text(strip=True))

                all\_quotes\_data.append({

                    'quote': text,

                    'author': author,

                    'tags': ", ".join(tags\_list)

                })

            next\_button = soup.find('li', class\_='next')

            if next\_button:

                current\_url = next\_button.find('a')['href']

            else:

                current\_url = None

        else:

            print(f"Error: Received status code {response.status\_code}")

            break

    except requests.exceptions.RequestException as e:

        print(f"An error occurred during the request: {e}")

        break

    print("Waiting 1 second before next request...")

    time.sleep(1)

print(f"\nScraping complete! Found {len(all\_quotes\_data)} quotes.")

if all\_quotes\_data:

    df = pd.DataFrame(all\_quotes\_data)

    print("\nApplying data masking to author names...")

    def mask\_name(name):

        if not isinstance(name, str) or len(name) == 0:

            return ""

        return name[0] + '\*' \* (len(name) - 1)

    df['masked\_author'] = df['author'].apply(mask\_name)

    print("\n--- DataFrame with Masked Data (Top 5) ---")

    print(df[['quote', 'author', 'masked\_author']].head())

    df.to\_csv('quotes\_masked.csv', index=False, encoding='utf-8')

    print("\nSuccessfully saved masked data to 'quotes\_masked.csv'")

else:

    print("No data was scraped, so no files were saved.")

print("Script finished."

**6. Ethical Considerations**

Respecting Server Resources: A time.sleep(1) delay was added between requests.

Identifying the Scraper: A User-Agent header was set to identify the script's purpose.

Error Handling: try-except blocks were used to handle network errors gracefully.