To bypass sophisticated bot detection mechanisms, you can use browser automation tools like Selenium along with proxies. Selenium allows you to control a web browser programmatically, which can help you emulate human behavior more effectively. Additionally, using proxies can help you avoid IP bans.

Here’s how you can set up Selenium with proxies to scrape data:

### Step-by-Step Guide

1. \*\*Install Selenium and WebDriver\*\*:

First, you need to install Selenium and download the appropriate WebDriver for your browser (e.g., ChromeDriver for Google Chrome).

```sh

pip install selenium

```

Download ChromeDriver from [here](https://sites.google.com/a/chromium.org/chromedriver/downloads) and place it in a directory included in your system's PATH.

2. \*\*Set Up Proxies\*\*:

You can use a list of proxies to rotate IP addresses. Make sure to use reliable proxies to avoid detection.

3. \*\*Update

quotes2scrape.py

to Use Selenium\*\*:

Here’s an example of how you can update your

quotes2scrape.py

to use Selenium with proxies:

```python

# quotes2scrape2.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.proxy import Proxy, ProxyType

from bs4 import BeautifulSoup

import random

def get\_random\_proxy():

# List of free proxies (example)

proxies = [

"http://103.216.82.153:6666",

"http://103.216.82.22:6666",

"http://103.216.82.29:6666",

# Add more proxies here

]

return random.choice(proxies)

def scrape\_data():

url = "http://quotes.toscrape.com/"

# Set up proxy

proxy = get\_random\_proxy()

proxy\_options = Proxy()

proxy\_options.proxy\_type = ProxyType.MANUAL

proxy\_options.http\_proxy = proxy

proxy\_options.ssl\_proxy = proxy

# Set up Chrome options

chrome\_options = Options()

chrome\_options.add\_argument("--headless") # Run in headless mode

chrome\_options.add\_argument("--disable-gpu")

chrome\_options.add\_argument("--no-sandbox")

chrome\_options.add\_argument("--disable-dev-shm-usage")

chrome\_options.proxy = proxy\_options

# Set up Chrome driver

service = Service(executable\_path="path/to/chromedriver")

driver = webdriver.Chrome(service=service, options=chrome\_options)

try:

driver.get(url)

soup = BeautifulSoup(driver.page\_source, "html.parser")

quotes = [

{

"text": quote.find("span", class\_="text").get\_text(),

"author": quote.find("small", class\_="author").get\_text(),

}

for quote in soup.find\_all("div", class\_="quote")

]

return {"quotes": quotes}

except Exception as e:

return {"error": str(e)}

finally:

driver.quit()

**Using Paid Proxy Services**

If you decide to use a paid proxy service, follow these steps:

1. **Sign Up for a Proxy Service**: Choose a proxy service provider and sign up for an account.
2. **Obtain Proxy Credentials**: Get the proxy IP addresses, ports, and authentication credentials (username and password) from the provider.
3. **Update the Proxy List**: Update the proxies list in the script with the provided proxy details.

**Example with Paid Proxies**

def get\_random\_proxy():

# List of paid proxies (example)

proxies = [

"http://username:password@proxy1.com:port",

"http://username:password@proxy2.com:port",

# Add more proxies here

]

return random.choice(proxies)

This setup should help you bypass basic bot detection mechanisms and scrape data more effectively.

### Additional Tips

- \*\*Rotate User Agents\*\*: Use different user agents to mimic different browsers and devices.

- \*\*Add Delays\*\*: Introduce random delays between requests to mimic human behavior.

- \*\*Handle Captchas\*\*: Use services like 2Captcha to solve captchas if encountered.

### Install ChromeDriver

Make sure to download ChromeDriver and place it in a directory included in your system's PATH. You can download ChromeDriver from [here](https://sites.google.com/a/chromium.org/chromedriver/downloads).

### Running the Script

Ensure you have the necessary dependencies installed:

```sh

pip install selenium beautifulsoup4

```

Then, run your script:

```sh

python quotes2scrape.py

```

This setup should help you bypass basic bot detection mechanisms and scrape data more effectively.

# quotes2scrape2.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from bs4 import BeautifulSoup

import os

import time

# Path to your locally downloaded ChromeDriver (assuming it's extracted)

CHROME\_DRIVER\_PATH = "C:\\Program Files\\chrome\\chromedriver-win64\\chromedriver.exe" # Adjust if different

# Verify that the ChromeDriver path is correct

if not os.path.exists(CHROME\_DRIVER\_PATH):

raise FileNotFoundError(f"ChromeDriver not found at {CHROME\_DRIVER\_PATH}")

def scrape\_data():

url = "http://quotes.toscrape.com/"

# Set up Chrome options

chrome\_options = Options()

chrome\_options.add\_argument("--headless") # Run in headless mode

chrome\_options.add\_argument("--disable-gpu")

chrome\_options.add\_argument("--no-sandbox")

chrome\_options.add\_argument("--disable-dev-shm-usage")

# Set up Chrome driver

service = Service(executable\_path=CHROME\_DRIVER\_PATH)

driver = webdriver.Chrome(service=service, options=chrome\_options)

try:

driver.get(url)

# Wait for page to load

time.sleep(3) # Wait for 3 seconds or adjust as needed

# Check if the quotes are loaded and print page source for debugging

soup = BeautifulSoup(driver.page\_source, "html.parser")

print("Debug - Page source loaded")

print(driver.page\_source) # Print the page source to confirm content structure

# Attempt to locate quotes by checking for 'div' tags with class 'quote'

quotes = [

{

"text": (

quote.find("span", class\_="text").get\_text()

if quote.find("span", class\_="text")

else ""

),

"author": (

quote.find("small", class\_="author").get\_text()

if quote.find("small", class\_="author")

else ""

),

}

for quote in soup.find\_all("div", class\_="quote")

]

# Check if any quotes were found

if not quotes:

print(

"No quotes found. Check if the structure of HTML matches the selector."

)

return {"quotes": quotes}

except Exception as e:

return {"error": str(e)}

finally:

driver.quit()

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

result = scrape\_data()

print(result)

# quotesWoProxyTest.py import requests response = requests.get("http://quotes.toscrape.com/") print(response.text) (venv) (base) PS C:\Users\nhari\OneDrive\Documents\iw tp projects\scrapper\_api\_project> python quotesWoProxyTest.py <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Quotes to Scrape</title> <link rel="stylesheet" href="/static/bootstrap.min.css"> <link rel="stylesheet" href="/static/main.css"> </head> <body> <div class="container"> <div class="row header-box"> <div class="col-md-8"> <h1> <a href="/" style="text-decoration: none">Quotes to Scrape</a> </h1> </div> <div class="col-md-4"> <p> <a href="/login">Login</a> </p> </div> </div> <div class="row"> <div class="col-md-8"> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”</span> <span>by <small class="author" itemprop="author">Albert Einstein</small> <a href="/author/Albert-Einstein">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="change,deep-thoughts,thinking,world" / > <a class="tag" href="/tag/change/page/1/">change</a> <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a> <a class="tag" href="/tag/thinking/page/1/">thinking</a> <a class="tag" href="/tag/world/page/1/">world</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“It is our choices, Harry, that show what we truly are, far more than our abilities.”</span> <span>by <small class="author" itemprop="author">J.K. Rowling</small> <a href="/author/J-K-Rowling">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="abilities,choices" / > <a class="tag" href="/tag/abilities/page/1/">abilities</a> <a class="tag" href="/tag/choices/page/1/">choices</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”</span> <span>by <small class="author" itemprop="author">Albert Einstein</small> <a href="/author/Albert-Einstein">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="inspirational,life,live,miracle,miracles" / > <a class="tag" href="/tag/inspirational/page/1/">inspirational</a> <a class="tag" href="/tag/life/page/1/">life</a> <a class="tag" href="/tag/live/page/1/">live</a> <a class="tag" href="/tag/miracle/page/1/">miracle</a> <a class="tag" href="/tag/miracles/page/1/">miracles</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”</span> <span>by <small class="author" itemprop="author">Jane Austen</small> <a href="/author/Jane-Austen">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="aliteracy,books,classic,humor" / > <a class="tag" href="/tag/aliteracy/page/1/">aliteracy</a> <a class="tag" href="/tag/books/page/1/">books</a> <a class="tag" href="/tag/classic/page/1/">classic</a> <a class="tag" href="/tag/humor/page/1/">humor</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“Imperfection is beauty, madness is genius and it&#39;s better to be absolutely ridiculous than absolutely boring.”</span> <span>by <small class="author" itemprop="author">Marilyn Monroe</small> <a href="/author/Marilyn-Monroe">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="be-yourself,inspirational" / > <a class="tag" href="/tag/be-yourself/page/1/">be-yourself</a> <a class="tag" href="/tag/inspirational/page/1/">inspirational</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“Try not to become a man of success. Rather become a man of value.”</span> <span>by <small class="author" itemprop="author">Albert Einstein</small> <a href="/author/Albert-Einstein">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="adulthood,success,value" / > <a class="tag" href="/tag/adulthood/page/1/">adulthood</a> <a class="tag" href="/tag/success/page/1/">success</a> <a class="tag" href="/tag/value/page/1/">value</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“It is better to be hated for what you are than to be loved for what you are not.”</span> <span>by <small class="author" itemprop="author">André Gide</small> <a href="/author/Andre-Gide">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="life,love" / > <a class="tag" href="/tag/life/page/1/">life</a> <a class="tag" href="/tag/love/page/1/">love</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“I have not failed. I&#39;ve just found 10,000 ways that won&#39;t work.”</span> <span>by <small class="author" itemprop="author">Thomas A. Edison</small> <a href="/author/Thomas-A-Edison">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="edison,failure,inspirational,paraphrased" / > <a class="tag" href="/tag/edison/page/1/">edison</a> <a class="tag" href="/tag/failure/page/1/">failure</a> <a class="tag" href="/tag/inspirational/page/1/">inspirational</a> <a class="tag" href="/tag/paraphrased/page/1/">paraphrased</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“A woman is like a tea bag; you never know how strong it is until it&#39;s in hot water.”</span> <span>by <small class="author" itemprop="author">Eleanor Roosevelt</small> <a href="/author/Eleanor-Roosevelt">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="misattributed-eleanor-roosevelt" / > <a class="tag" href="/tag/misattributed-eleanor-roosevelt/page/1/">misattributed-eleanor-roosevelt</a> </div> </div> <div class="quote" itemscope itemtype="http://schema.org/CreativeWork"> <span class="text" itemprop="text">“A day without sunshine is like, you know, night.”</span> <span>by <small class="author" itemprop="author">Steve Martin</small> <a href="/author/Steve-Martin">(about)</a> </span> <div class="tags"> Tags: <meta class="keywords" itemprop="keywords" content="humor,obvious,simile" / > <a class="tag" href="/tag/humor/page/1/">humor</a> <a class="tag" href="/tag/obvious/page/1/">obvious</a> <a class="tag" href="/tag/simile/page/1/">simile</a> </div> </div> <nav> <ul class="pager"> <li class="next"> <a href="/page/2/">Next <span aria-hidden="true">&rarr;</span></a> </li> </ul> </nav> </div> <div class="col-md-4 tags-box"> <h2>Top Ten tags</h2> <span class="tag-item"> <a class="tag" style="font-size: 28px" href="/tag/love/">love</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 26px" href="/tag/inspirational/">inspirational</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 26px" href="/tag/life/">life</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 24px" href="/tag/humor/">humor</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 22px" href="/tag/books/">books</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 14px" href="/tag/reading/">reading</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 10px" href="/tag/friendship/">friendship</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 8px" href="/tag/friends/">friends</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 8px" href="/tag/truth/">truth</a> </span> <span class="tag-item"> <a class="tag" style="font-size: 6px" href="/tag/simile/">simile</a> </span> </div> </div> </div> <footer class="footer"> <div class="container"> <p class="text-muted"> Quotes by: <a href="https://www.goodreads.com/quotes">GoodReads.com</a> </p> <p class="copyright"> Made with <span class='zyte'>❤</span> by <a class='zyte' href="https://www.zyte.com">Zyte</a> </p> </div> </footer> </body> </html>

**ChatGPT said:**

ChatGPT

It seems that your script successfully retrieved the HTML content from the "Quotes to Scrape" website, which is a simple page displaying quotes. The response.text contains the full HTML code of the page, including quotes, authors, and tags. If you're looking to scrape specific information from this page, such as the quotes or the authors, you will need to parse this HTML data using a library like BeautifulSoup to extract the desired details.

Here's an example of how you can extract all the quotes and their authors using BeautifulSoup:

python

Copy code

import requests

from bs4 import BeautifulSoup

# Send a GET request to the website

response = requests.get("http://quotes.toscrape.com/")

# Parse the HTML content with BeautifulSoup

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

# Find all quote elements

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

# Loop through each quote and print the quote and author

for quote in quotes:

text = quote.find("span", class\_="text").get\_text()

author = quote.find("small", class\_="author").get\_text()

print(f"Quote: {text}\nAuthor: {author}\n")

**Explanation:**

* We use requests.get() to fetch the content of the page.
* We then parse the HTML content with BeautifulSoup.
* The find\_all() function is used to find all the div elements with the class quote, which contain the quotes.
* Inside each quote, we use .find() to extract the quote text and the author's name.
* Finally, we print each quote and its corresponding author.

This should give you a more readable format of the quotes and authors from the webpage. Let me know if you need further assistance!

4o mini

**You said:**

No what I want is the following code snippet to work with proxy settings please - # quotes2scrape2.py  
  
  
The

quotes2scrape2.py

script can be adapted to scrape other websites, but there are a few considerations to keep in mind:

1. \*\*HTML Structure\*\*: The script relies on specific HTML structure and selectors to extract data. You will need to update the selectors to match the structure of the new website.

2. \*\*Bot Detection\*\*: Different websites have varying levels of bot detection. You may need to adjust the script to handle these mechanisms, such as adding more sophisticated user-agent rotation, handling CAPTCHAs, or using more advanced proxy services.

3. \*\*Rate Limiting\*\*: Some websites implement rate limiting to prevent excessive scraping. You may need to introduce delays between requests to avoid being blocked.

### Adapting the Script for Other Websites

To adapt the script for other websites, follow these steps:

1. \*\*Update the URL\*\*: Change the

url

variable to the new website's URL.

2. \*\*Update Selectors\*\*: Modify the selectors used in the `BeautifulSoup` parsing to match the new website's HTML structure.

3. \*\*Handle Bot Detection\*\*: Adjust the proxy and user-agent settings as needed to handle the new website's bot detection mechanisms.

### Example: Adapting the Script for Another Website

Here is an example of how you might adapt the script to scrape a different website:

```python

# quotes2scrape2.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.proxy import Proxy, ProxyType

from bs4 import BeautifulSoup

import random

import os

import time

# Path to your locally downloaded ChromeDriver (assuming it's extracted)

CHROME\_DRIVER\_PATH = "C:\\Program Files\\chrome\\chromedriver-win64\\chromedriver.exe" # Adjust if different

# Verify that the ChromeDriver path is correct

if not os.path.exists(CHROME\_DRIVER\_PATH):

raise FileNotFoundError(f"ChromeDriver not found at {CHROME\_DRIVER\_PATH}")

# List of user agents (example)

USER\_AGENTS = [

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3",

"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_6) AppleWebKit/602.3.12 (KHTML, like Gecko) Version/10.0.3 Safari/602.3.12",

"Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110 Safari/537.36",

# Add more user agents here

]

# List of proxies (example)

PROXIES = [

"http://103.216.82.153:6666",

"http://103.216.82.22:6666",

"http://103.216.82.29:6666",

# Add more proxies here

]

def get\_random\_proxy():

return random.choice(PROXIES)

def get\_random\_user\_agent():

return random.choice(USER\_AGENTS)

def random\_delay(min\_delay=1, max\_delay=5):

time.sleep(random.uniform(min\_delay, max\_delay))

def scrape\_data():

url = "http://example.com/" # Change this to the new website's URL

# Set up proxy

proxy = get\_random\_proxy()

proxy\_options = Proxy()

proxy\_options.proxy\_type = ProxyType.MANUAL

proxy\_options.http\_proxy = proxy

proxy\_options.ssl\_proxy = proxy

# Set up Chrome options

chrome\_options = Options()

chrome\_options.add\_argument("--headless") # Run in headless mode

chrome\_options.add\_argument("--disable-gpu")

chrome\_options.add\_argument("--no-sandbox")

chrome\_options.add\_argument("--disable-dev-shm-usage")

chrome\_options.add\_argument(f"--proxy-server={proxy}") # Configure the proxy

chrome\_options.add\_argument(f"user-agent={get\_random\_user\_agent()}") # Rotate user agent

# Set up Chrome driver

service = Service(executable\_path=CHROME\_DRIVER\_PATH)

driver = webdriver.Chrome(service=service, options=chrome\_options)

try:

driver.get(url)

# Wait for page to load

random\_delay(3, 5) # Random delay between 3 to 5 seconds

# Check if the content is loaded and print page source for debugging

soup = BeautifulSoup(driver.page\_source, "html.parser")

print("Debug - Page source loaded")

print(driver.page\_source) # Print the page source to confirm content structure

# Update the selectors to match the new website's HTML structure

# Example: Extracting titles and authors

data = [

{

"title": (

item.find("h2", class\_="title").get\_text()

if item.find("h2", class\_="title")

else ""

),

"author": (

item.find("span", class\_="author").get\_text()

if item.find("span", class\_="author")

else ""

),

}

for item in soup.find\_all("div", class\_="item")

]

# Check if any data was found

if not data:

print(

"No data found. Check if the structure of HTML matches the selector."

)

return {"data": data}

except Exception as e:

return {"error": str(e)}

finally:

driver.quit()

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

result = scrape\_data()

print(result)

```

### Explanation

1. \*\*Update the URL\*\*: Change the

url

variable to the new website's URL.

2. \*\*Update Selectors\*\*: Modify the selectors used in the `BeautifulSoup` parsing to match the new website's HTML structure.

3. \*\*Handle Bot Detection\*\*: Adjust the proxy and user-agent settings as needed to handle the new website's bot detection mechanisms.

### Running the Flask App

1. \*\*Activate your virtual environment\*\* (if not already activated):

```sh

venv\Scripts\activate

```

2. \*\*Run the Flask app\*\*:

```sh

python app.py

```

3. \*\*Access the `/scrape/quotes` endpoint\*\*:

Open your browser and go to `http://127.0.0.1:5001/scrape/quotes`.

This setup should help you adapt the script to scrape other websites by updating the URL and selectors. If you encounter any further issues or have any additional questions, please let me know!

Similar code found with 2 license types

Choosing a reliable proxy provider largely depends on your budget and specific use case. Here are some reputable proxy providers known for stability, good performance, and options for rotating or residential proxies:

### 1. \*\*Bright Data (formerly Luminati)\*\*

- \*\*Type\*\*: Residential, data center, mobile proxies

- \*\*Pros\*\*: Highly reliable, large IP pool, rotating IPs, customizable sessions

- \*\*Cons\*\*: Higher price point

- \*\*Website\*\*: [Bright Data](https://brightdata.com/)

### 2. \*\*Smartproxy\*\*

- \*\*Type\*\*: Residential and data center proxies

- \*\*Pros\*\*: High-quality residential proxies, affordable pricing, user-friendly interface, great for web scraping

- \*\*Cons\*\*: Limited mobile proxy options

- \*\*Website\*\*: [Smartproxy](https://smartproxy.com/)

### 3. \*\*ScraperAPI\*\*

- \*\*Type\*\*: Rotating proxies for scraping

- \*\*Pros\*\*: Handles IP rotation, CAPTCHA-solving, and geotargeting automatically; affordable for small projects

- \*\*Cons\*\*: Limited for non-scraping use cases

- \*\*Website\*\*: [ScraperAPI](https://www.scraperapi.com/)

### 4. \*\*Oxylabs\*\*

- \*\*Type\*\*: Residential, data center, and mobile proxies

- \*\*Pros\*\*: Reliable, large pool of residential IPs, rotating and sticky sessions available

- \*\*Cons\*\*: Higher cost, more enterprise-oriented

- \*\*Website\*\*: [Oxylabs](https://oxylabs.io/)

### 5. \*\*ProxyMesh\*\*

- \*\*Type\*\*: Rotating proxies

- \*\*Pros\*\*: Simple pricing, easy to integrate with web scraping tools, handles IP rotation automatically

- \*\*Cons\*\*: Limited to IPs in certain locations, less comprehensive support compared to larger providers

- \*\*Website\*\*: [ProxyMesh](https://proxymesh.com/)

### 6. \*\*GeoSurf\*\*

- \*\*Type\*\*: Residential proxies

- \*\*Pros\*\*: Good global IP coverage, works well for accessing geo-blocked content, browser extensions available

- \*\*Cons\*\*: Slightly pricier than some alternatives

- \*\*Website\*\*: [GeoSurf](https://geosurf.com/)

### 7. \*\*PacketStream\*\*

- \*\*Type\*\*: Residential proxies

- \*\*Pros\*\*: Affordable, peer-to-peer residential IPs, pay-as-you-go pricing

- \*\*Cons\*\*: Slower speeds at times due to peer-to-peer nature

- \*\*Website\*\*: [PacketStream](https://packetstream.io/)

Each of these providers offers free trials or low-cost starter plans, so you can test out their proxies to see which works best for your specific needs. For web scraping, rotating residential proxies are generally recommended, as they can help avoid blocks and CAPTCHAs.

# quotes2scrape2.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.proxy import Proxy, ProxyType

from bs4 import BeautifulSoup

import random

import os

import time

import requests

# Path to your locally downloaded ChromeDriver (assuming it's extracted)

CHROME\_DRIVER\_PATH = "C:\\Program Files\\chrome\\chromedriver-win64\\chromedriver.exe"  # Adjust if different

# Verify that the ChromeDriver path is correct

if not os.path.exists(CHROME\_DRIVER\_PATH):

    raise FileNotFoundError(f"ChromeDriver not found at {CHROME\_DRIVER\_PATH}")

# List of user agents (example)

USER\_AGENTS = [

    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3",

    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_6) AppleWebKit/602.3.12 (KHTML, like Gecko) Version/10.0.3 Safari/602.3.12",

    "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110 Safari/537.36",

    # Add more user agents here

]

def get\_proxies\_from\_sslproxies():

    url = "https://www.sslproxies.org/"

    response = requests.get(url)

    if response.status\_code == 200:

        soup = BeautifulSoup(response.content, "html.parser")

        proxies = []

        for row in soup.select("table#proxylisttable tbody tr"):

            columns = row.find\_all("td")

            if len(columns) >= 8:

                ip = columns[0].text.strip()

                port = columns[1].text.strip()

                https = columns[6].text.strip()

                proxy\_type = "https" if https == "yes" else "http"

                print(

                    f"Debug - Found proxy: {proxy\_type}://{ip}:{port}"

                )  # Debug statement

                proxies.append(f"{proxy\_type}://{ip}:{port}")

        print(f"Debug - Proxies fetched: {proxies}")  # Debug statement

        return proxies

    else:

        print(

            f"Debug - Failed to fetch proxies: {response.status\_code}"

        )  # Debug statement

        raise Exception("Failed to fetch proxies from SSLProxies")

def get\_random\_proxy(proxies):

    if not proxies:

        raise ValueError("No proxies available")

    return random.choice(proxies)

def get\_random\_user\_agent():

    return random.choice(USER\_AGENTS)

def random\_delay(min\_delay=1, max\_delay=5):

    time.sleep(random.uniform(min\_delay, max\_delay))

def scrape\_data():

    url = "http://quotes.toscrape.com/"

    # Fetch proxies from SSLProxies

    proxies = get\_proxies\_from\_sslproxies()

    # Check if proxies are available

    if not proxies:

        return {"error": "No proxies available"}

    # Set up proxy

    proxy = get\_random\_proxy(proxies)

    proxy\_options = Proxy()

    proxy\_options.proxy\_type = ProxyType.MANUAL

    proxy\_options.http\_proxy = proxy

    proxy\_options.ssl\_proxy = proxy

    # Set up Chrome options

    chrome\_options = Options()

    chrome\_options.add\_argument("--headless")  # Run in headless mode

    chrome\_options.add\_argument("--disable-gpu")

    chrome\_options.add\_argument("--no-sandbox")

    chrome\_options.add\_argument("--disable-dev-shm-usage")

    chrome\_options.add\_argument(f"--proxy-server={proxy}")  # Configure the proxy

    chrome\_options.add\_argument(

        f"user-agent={get\_random\_user\_agent()}"

    )  # Rotate user agent

    # Set up Chrome driver

    service = Service(executable\_path=CHROME\_DRIVER\_PATH)

    driver = webdriver.Chrome(service=service, options=chrome\_options)

    try:

        driver.get(url)

        # Wait for page to load

        random\_delay(3, 5)  # Random delay between 3 to 5 seconds

        # Check if the quotes are loaded and print page source for debugging

        soup = BeautifulSoup(driver.page\_source, "html.parser")

        print("Debug - Page source loaded")

        print(driver.page\_source)  # Print the page source to confirm content structure

        # Attempt to locate quotes by checking for 'div' tags with class 'quote'

        quotes = [

            {

                "text": (

                    quote.find("span", class\_="text").get\_text()

                    if quote.find("span", class\_="text")

                    else ""

                ),

                "author": (

                    quote.find("small", class\_="author").get\_text()

                    if quote.find("small", class\_="author")

                    else ""

                ),

            }

            for quote in soup.find\_all("div", class\_="quote")

        ]

        # Check if any quotes were found

        if not quotes:

            print(

                "No quotes found. Check if the structure of HTML matches the selector."

            )

        return {"quotes": quotes}

    except Exception as e:

        return {"error": str(e)}

    finally:

        driver.quit()

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    result = scrape\_data()

    print(result)

If both free-proxy-list.net and geonode are not providing the necessary proxies, you can try other free proxy list providers such as:

1. **ProxyScrape**: [https://www.proxyscrape.com/free-proxy-list](vscode-file://vscode-app/c:/Users/nhari/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
2. **SSLProxies**: [https://www.sslproxies.org/](vscode-file://vscode-app/c:/Users/nhari/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
3. **Spys.one**: [https://spys.one/en/free-proxy-list/](vscode-file://vscode-app/c:/Users/nhari/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

<https://www.socks-proxy.net/> - used 192.252.211.193 4145 US United States Socks4 Anonymous Yes 1 min ago - which is working well

1. **Choose a Residential Proxy Provider**:
   * There are several residential proxy providers available, such as Luminati, Smartproxy, and Oxylabs. You will need to sign up for their services and obtain the proxy details.

# quotes2scrape2.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.proxy import Proxy, ProxyType

from bs4 import BeautifulSoup

import random

import os

import time

# Path to your locally downloaded ChromeDriver (assuming it's extracted)

CHROME\_DRIVER\_PATH = "C:\\Program Files\\chrome\\chromedriver-win64\\chromedriver.exe"  # Adjust if different

# Verify that the ChromeDriver path is correct

if not os.path.exists(CHROME\_DRIVER\_PATH):

    raise FileNotFoundError(f"ChromeDriver not found at {CHROME\_DRIVER\_PATH}")

# List of user agents (example)

USER\_AGENTS = [

    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3",

    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_6) AppleWebKit/602.3.12 (KHTML, like Gecko) Version/10.0.3 Safari/602.3.12",

    "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110 Safari/537.36",

    # Add more user agents here

]

def get\_random\_user\_agent():

    return random.choice(USER\_AGENTS)

def random\_delay(min\_delay=1, max\_delay=5):

    time.sleep(random.uniform(min\_delay, max\_delay))

def scrape\_data():

    url = "http://quotes.toscrape.com/"

    # Set up proxy

    proxy\_ip = "192.252.211.193"  # Example proxy IP

    proxy\_port = "4145"  # Example proxy port

    proxy = f"socks4://{proxy\_ip}:{proxy\_port}"

    proxy\_options = Proxy()

    proxy\_options.proxy\_type = ProxyType.MANUAL

    proxy\_options.socks\_proxy = proxy

    # Set up Chrome options

    chrome\_options = Options()

    chrome\_options.add\_argument("--headless")  # Run in headless mode

    chrome\_options.add\_argument("--disable-gpu")

    chrome\_options.add\_argument("--no-sandbox")

    chrome\_options.add\_argument("--disable-dev-shm-usage")

    chrome\_options.add\_argument(f"--proxy-server={proxy}")  # Configure the proxy

    chrome\_options.add\_argument(

        f"user-agent={get\_random\_user\_agent()}"

    )  # Rotate user agent

    # Set up Chrome driver

    service = Service(executable\_path=CHROME\_DRIVER\_PATH)

    driver = webdriver.Chrome(service=service, options=chrome\_options)

    try:

        print(f"Debug - Using proxy: {proxy}")

        driver.get(url)

        # Wait for page to load

        random\_delay(3, 5)  # Random delay between 3 to 5 seconds

        # Check if the quotes are loaded and print page source for debugging

        soup = BeautifulSoup(driver.page\_source, "html.parser")

        print("Debug - Page source loaded")

        print(driver.page\_source)  # Print the page source to confirm content structure

        # Attempt to locate quotes by checking for 'div' tags with class 'quote'

        quotes = [

            {

                "text": (

                    quote.find("span", class\_="text").get\_text()

                    if quote.find("span", class\_="text")

                    else ""

                ),

                "author": (

                    quote.find("small", class\_="author").get\_text()

                    if quote.find("small", class\_="author")

                    else ""

                ),

            }

            for quote in soup.find\_all("div", class\_="quote")

        ]

        # Check if any quotes were found

        if not quotes:

            print(

                "No quotes found. Check if the structure of HTML matches the selector."

            )

        return {"quotes": quotes}

    except Exception as e:

        print(f"Error: {e}")

        return {"error": str(e)}

    finally:

        driver.quit()

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    result = scrape\_data()

    print(result)