Asynchronous Web Scraping

Why is this interesting?

- Standard scraper using **requests** and **BeautifulSoup** is single threaded.
- Lack concurrency due to **Global Interpreter Lock** (GIL)
- Synchronous execution of URLs one after the other
- This slows down crawlers due to your scripts simply idling while they await the response from the web server
- This is problematic if you must iterate over thousands of URLs

Need Asynchronous Scraping

asyncio 3.4.3

- Ability to write **concurrent** code using the **async** and **await** syntax

aiohttp 3.6.2

- Provides high-level functionalities that streamline communication procedures between **HTTP client and server**

```python
import aiohttp
import asyncio
import nest_asyncio #allows event loop to be nested
nest_asyncio.apply()

sites = ["https://www.python.org/", "https://github.com/", "https://www.theverge.com/"]

async def get_html(session, url):
    async with session.get(url, ssl=False) as res: # make a get request for html source code
        return await res.text()

async def main():
    async with aiohttp.ClientSession() as session: # in the main coroutine, initiate an instance
        for site in sites:
            html = await get_html(session, site) # wait for this coroutine to process and return
            print(html[: 500]) # first 1000 characters

loop = asyncio.get_event_loop() # create event loop
loop.run_until_complete(main()) #run until all tasks are completed
```

What are the implications?

- Useful when downloading multiple web pages, often from different servers
- Send out requests for all pages we want to retrieve without waiting for the answer before asking for the next page
- Applications in a host of web scraping use cases

Pricing Intelligence

Reputation Management

Recruitment Analysis

What's in the future?

- Faster web scraping is an area of continuous research
- Other complexities include:

Client-side Rendering

Server-side Blacklisting

Infinite Scrolling

Handling Captchas

References:

- “Web Scraping.” *Asyncio Documentation*, https://asyncio.readthedocs.io/en/latest/webscraper.html.
- Nguyen, Q. (2018, December 17). Asynchronous Programming in Python for Web Scraping. Retrieved from https://blogs.oracle.com/datascience/asynchronous-programming-in-python-for-web-scraping.
- Welcome to AIOHTTP. (n.d.). Retrieved from https://aiohttp.readthedocs.io/en/stable/#.
- asyncio. (n.d.). Retrieved from https://pypi.org/project/asyncio/.