API query POST method (self-study)

- Apart from the GET method in previous slides, another common method for querying an API is **POST**, and basic syntax is as follows

```
>> payload = {'var1':'data1', 'var2':'data2'}
>> requests.post(url,data=payload)
```

- Always start by looking for the exact API endpoints, query method, parameters, and example queries in the API documentation

Dynamic web scraping (self-study)

- The selenium Python module lets you simulate several user actions on a browser (e.g., Firefox, Chrome) and navigate a website while you can dynamically extract information through interacting with the website like a natural user would

- Can simulate clicking a button, selecting from a dropdown list, inputting text data (e.g., log-on passwords)

- Try it yourself: ask ChatGPT (model GPT-4o or above) “give me a code sketch using Selenium to download data from [websiteURL]”

Collecting Web Data: Strategies

Step 1: formulate your business question and related data needs, identify potential data sources

- Think creatively, it's usually hard to get everything you need at once, but there are often more than one way to get some useful data
- For example, practice using ChatGPT to identify potential data sources and get new ideas about where to search and what to search for when looking for resources online

Collecting Web Data: Strategies

Step 2: design data collection strategies around the identified data sources and data needs

- For which units (companies, users, etc) do you need data, and at what frequency (just once, every year, daily, etc)
- What are your data storage needs: you can either store data locally, or put them on the cloud (e.g., AWS S3)
- Before writing any code, understand the structure of the data source (e.g., API endpoints and query structures)

Collecting Web Data: Strategies

Step 3: design the database structure and code to collect data from the identified data sources

- Most often you'll only need a small portion of the data that's available in the data source, so your code should only retrieve the information you need (and ignore information that you likely won't need), to be efficient both in terms of storage and speed
- Test the code on a small set of data, look for any errors or inconsistencies in the collected data, fix bugs in your code

Collecting Web Data: Strategies

Throughout the process: Be mindful about regulations

- Generally, do not collect user-identified individual-level data by yourself (in Europe, by GDPR): you can conduct surveys (e.g., Prolific) and get data from anonymous participants, you can collaborate with companies which will usually require you to sign non-disclosure agreements (NDAs)
- You may encounter increasing technical barriers over time due to websites blocking web crawlers

Exercise: Set up GitHub & Git Integration

1. Visit GitHub and create an account if you don't already have one.

2. Install VS Code and the GitHub Copilot extension. Follow the installation instructions to integrate Copilot with your GitHub account. Note: Copilot provides inline code completion, which will be helpful for your assignments.

3. Download and install Git from git-scm.com. Set up Git configuration by opening your terminal or command prompt and run:

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

4. Pull a GitHub repository to practice cloning and interacting with a repo locally. Clone a GitHub repository: either https://github.com/quantrocket-codeload/quant-finance-lectures or https://github.com/quantopian/research_public