**Web Scraping:**

Web scraping is the process of extracting data from websites by using automated software or bots. It involves fetching and parsing the HTML code of web pages to extract specific information, such as text, images, links, or other structured data.

Here's a simplified overview of how web scraping typically works:

1. Fetching: The web scraper sends an HTTP request to a target website and retrieves the HTML content of the web page.

2. Parsing: The HTML content is then parsed to identify and extract the desired data. This is usually done using libraries or tools like BeautifulSoup in Python.

3. Data extraction: The scraper locates specific elements within the HTML, such as HTML tags or CSS selectors, and extracts the relevant data. For example, it could extract product names, prices, or reviews from an e-commerce website.

4. Data manipulation: Once the data is extracted, it can be further processed or manipulated as per the requirements. This might involve cleaning the data, removing duplicates, or converting it into a structured format like JSON or CSV.

5. Storage or analysis: The scraped data can be stored in a database, a spreadsheet, or any other storage medium for future use. Alternatively, it can be used for immediate analysis or integration with other applications.

It's important to note that when web scraping, it's essential to respect the website's terms of service, comply with legal regulations, and be mindful of the website's server load. Some websites may have specific rules regarding scraping or may not permit it at all, so it's advisable to review a website's terms of service or contact the website owner before scraping.

**How Web Scrapers work:**

Web scrappers can extract all the data on particular sites or the specific data that a user wants. Ideally its best if you specify the data you want so that the web scraper only extracts that data quickly

What is web scrapping used for:

1. Price monitoring

2. Market Research

3. News Monitoring

4. Sentiment Analysis

5. Email Marketing

**Parser:**

* A parser is a basic tool to interpret or render information from a web document
* A parser receives input in the from of program instructions, commands & markup tags and outputs the web document as objects, methods and their attributes
* This enables you to extract the information in a meaningful way.
* A parser is also used to validate the input information before processing it.
* The extracted file can be understood and stored in the desired format only if it is parsed successfully.
* Failing to parse the data would eventually lead to failure of the entire process

There are various parsers supported by BeautifulSoup

html.parser: python based, fast and lenient

lxml html: lxml html is not built using python and it depends on C. however it is fast and lenient in nature

lxml xml: is the only xml parser available and it also depends on C

html5lib: is another python based parser; however it is slow and able to create valid HTML5

**Importance of Objects:**

* A web document gets transformed into a complex tree of objects
* A tree is defined as a collection of simple and complex objects

**Types of Objects**:

BeautifulSoup transforms a complex HTML document into a complex tree of python objects. There are 4 types of objects.

* Tag:

a tag object is an XML or HTML tag in the web document. Tags have a lot of attributes and methods

* Navigable String:

A NavigableString is a string or set of characters that corresponds to the text present within a tag

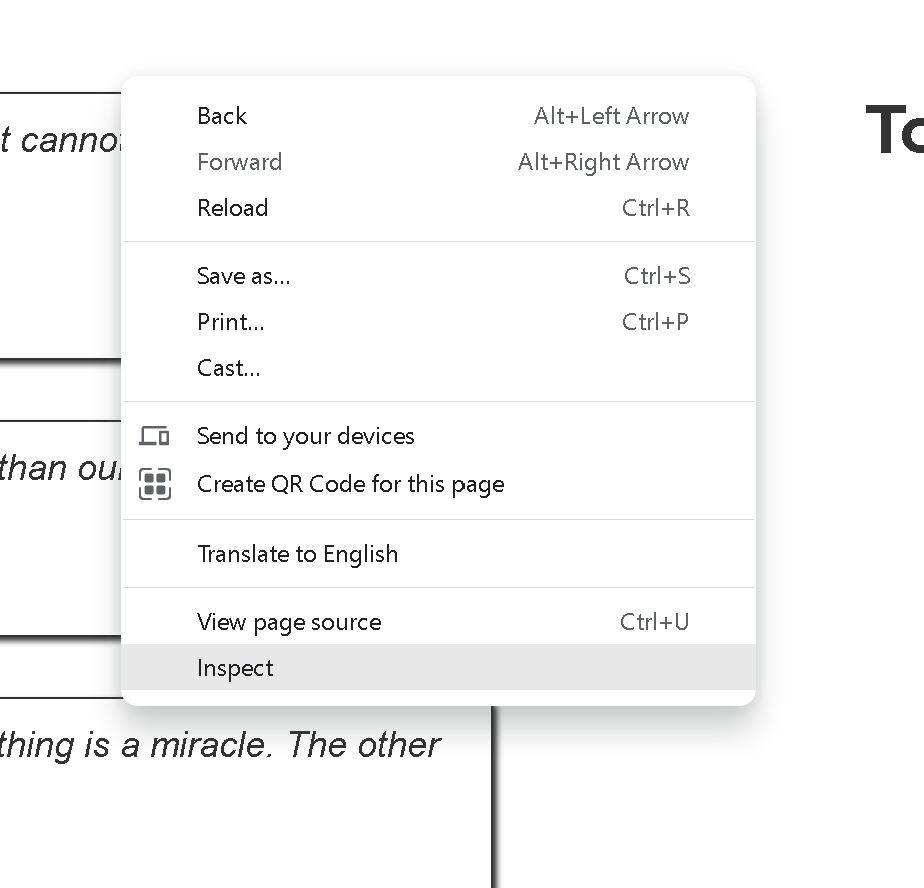
* BeautifulSoup:

A BeautifulSoup represents the entire web document and supports navigating and searching the documents tree

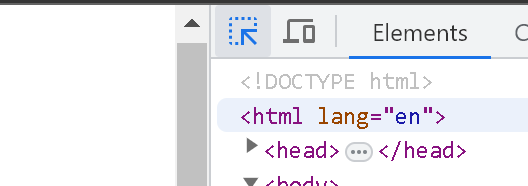
* Comment:

A comment represents the comment or information section of the document. It is a special type of NaviagableString

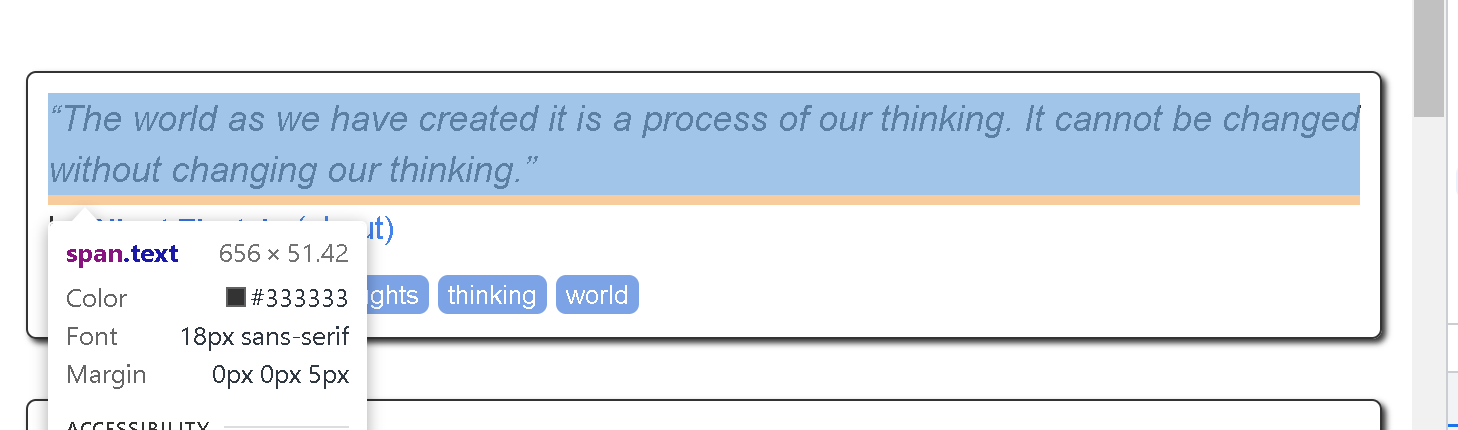
**Right Click on the web page & click inspect:**



Then click on arrow mark:



**When you select any quote it display the element, here it is “span” and class is “text”**



**For Author: element: small, class: author**

**For Tags: element: div, class: tags**

**Extract Data from API:**

**API:**

An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate and interact with each other. It acts as a messenger, defining how applications can request and exchange information.

Ex: Twitter, Facebook etc

* API generate output majorly in JSON or XML format
* Few API requires authentication

The main difference between scraping data from the web and using an API lies in the methods used to access and retrieve the data

**Web scraping vs API**

**Web Scraping:**

1. Data Source: When web scraping, you directly extract data from web pages, typically using HTML parsing techniques. You navigate through the website's structure, identify the relevant HTML elements, and extract the desired data.

2. Data Extraction: Web scraping involves parsing the HTML content of web pages, extracting specific elements or patterns, and transforming them into structured data. This process often requires techniques like XPath or CSS selectors to locate and extract the desired data.

3. Automation: Web scraping is often performed by writing custom code or using scraping frameworks/libraries. You have more control over the scraping process and can navigate through multiple pages, follow links, and handle various scenarios.

4. Data Availability: With web scraping, you can extract data from any website, regardless of whether the website provides an API or not. However, scraping without permission or against a website's terms of service may be unethical or illegal.

**API:**

1. Data Source: APIs (Application Programming Interfaces) provide a structured and controlled way to access and retrieve data from a website or online service. APIs expose specific endpoints and functionality that allow you to request and receive data in a standardized format, such as JSON or XML.

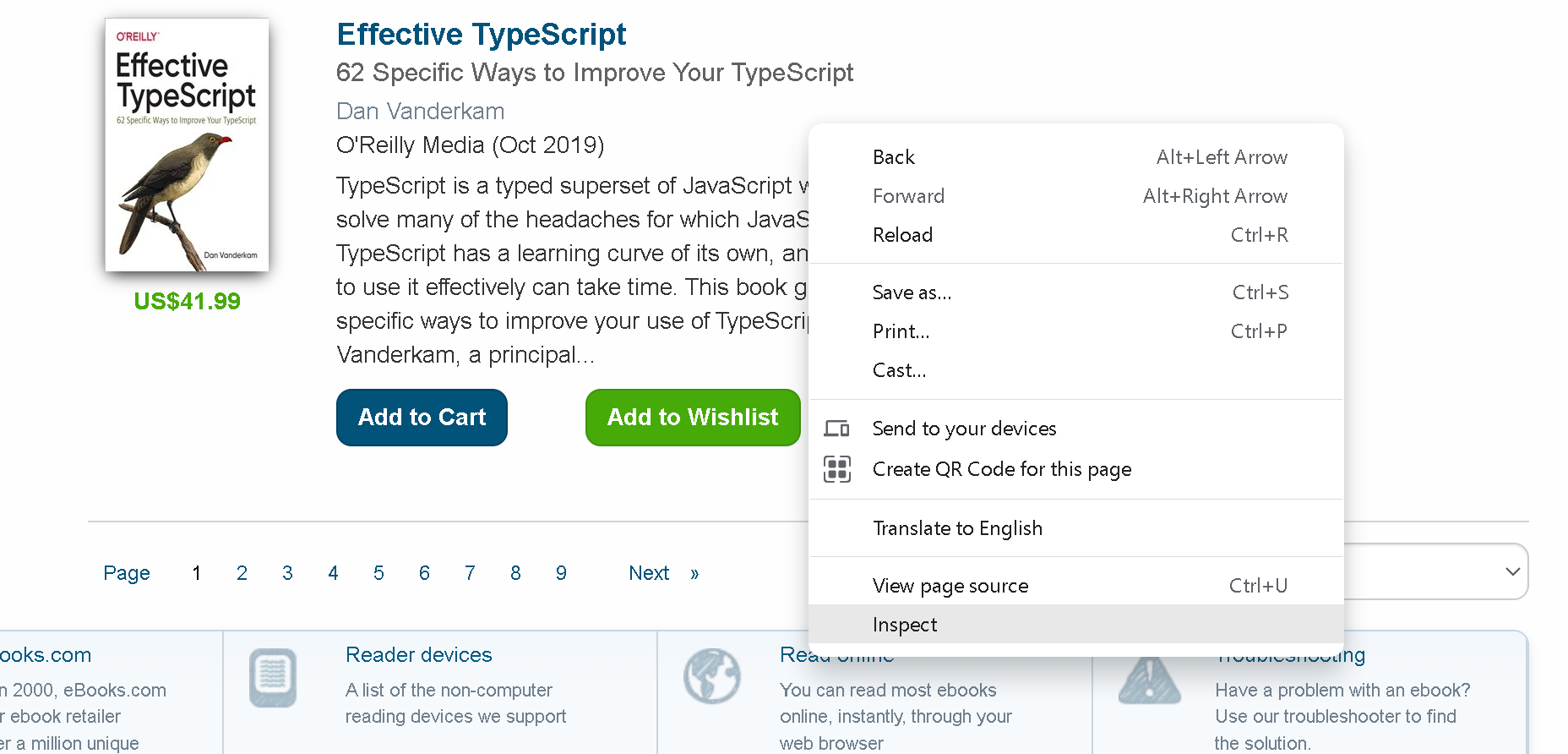
2. Data Retrieval: With an API, you make specific requests to the provided endpoints, passing any required parameters or authentication tokens. The API returns the requested data in a pre-defined format, which is usually easier to work with compared to scraping raw HTML.

3. Standardization: APIs follow predefined rules and data formats, making it easier to interact with and integrate data from different sources. APIs often provide documentation and clear guidelines on how to access and utilize the available data.

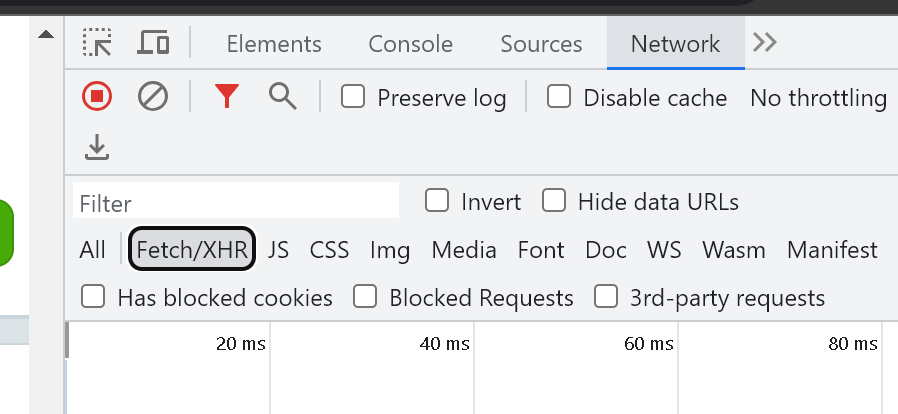
4. Security and Usage: APIs can have authentication mechanisms, rate limits, and usage policies in place. They allow controlled access to data and protect against abuse or unauthorized access. Using APIs typically ensures compliance with the terms of service and data usage policies set by the website.

In summary, web scraping involves extracting data directly from web pages using HTML parsing techniques, while APIs provide a structured and controlled way to access data from a website or service through specific endpoints and data formats. Web scraping offers more flexibility and the ability to extract data from any website, while APIs provide standardized access to data with defined rules and security measures.

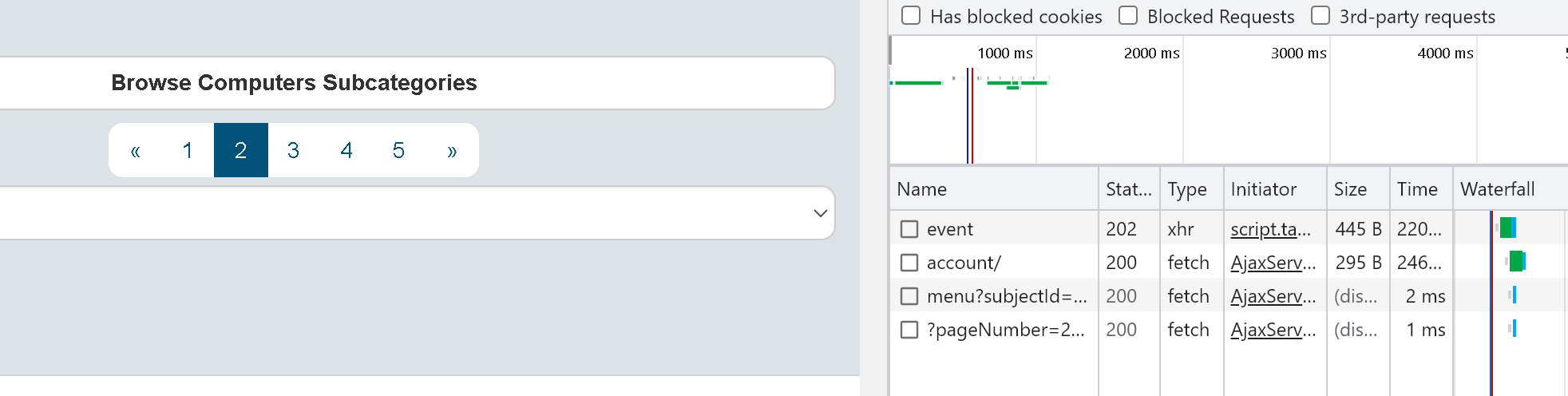
**Step\_1: Click inspect on required page**



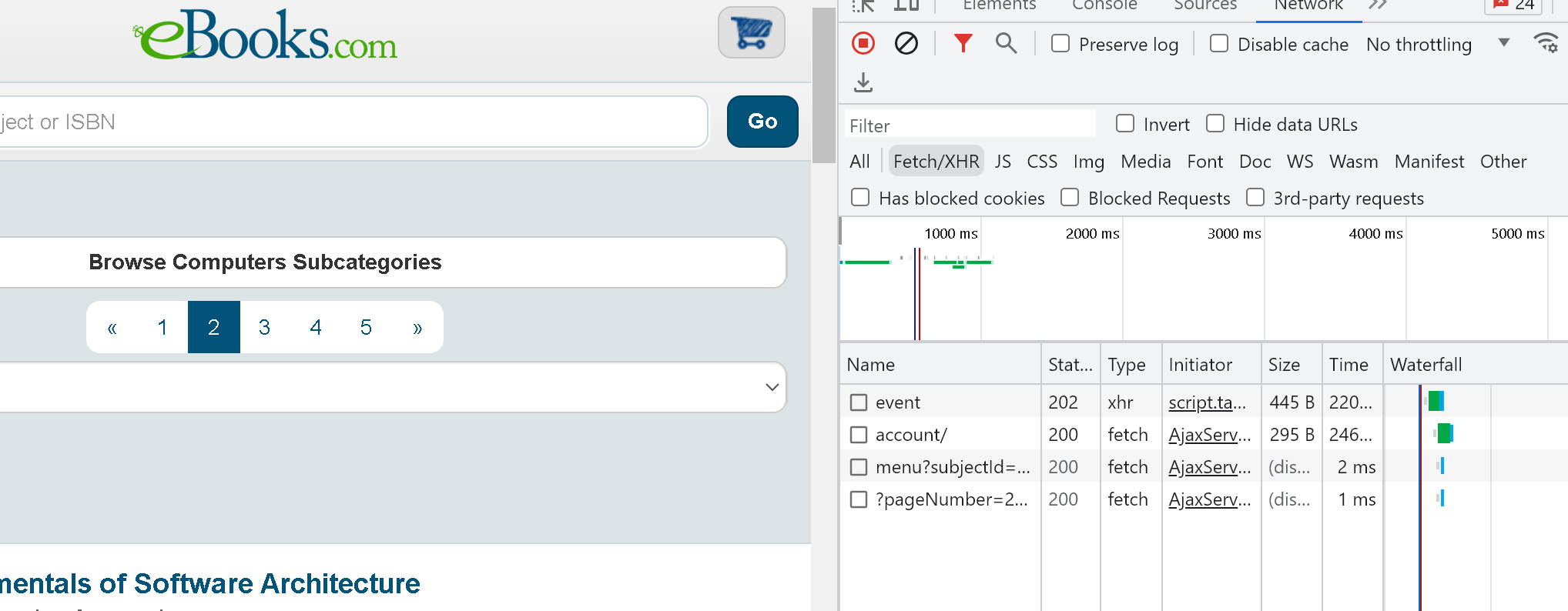
**Step\_2: Click on Network to get the request & then click Fetch/XHR**



**Step\_3: If you are not getting anything, then go the next page, here I go through the 2nd page.**

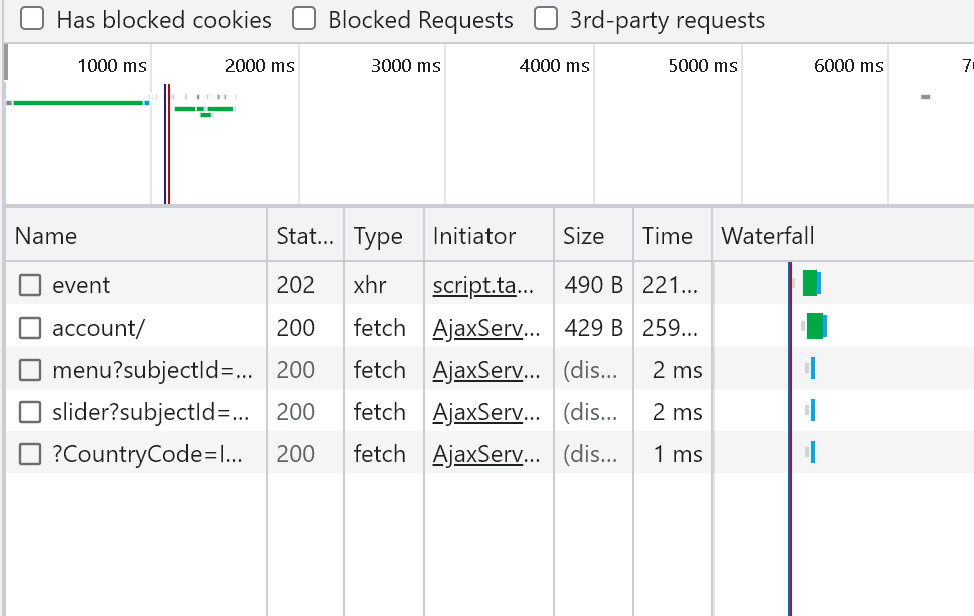


**Step\_4: Then click on clear option, which is rounded below image**

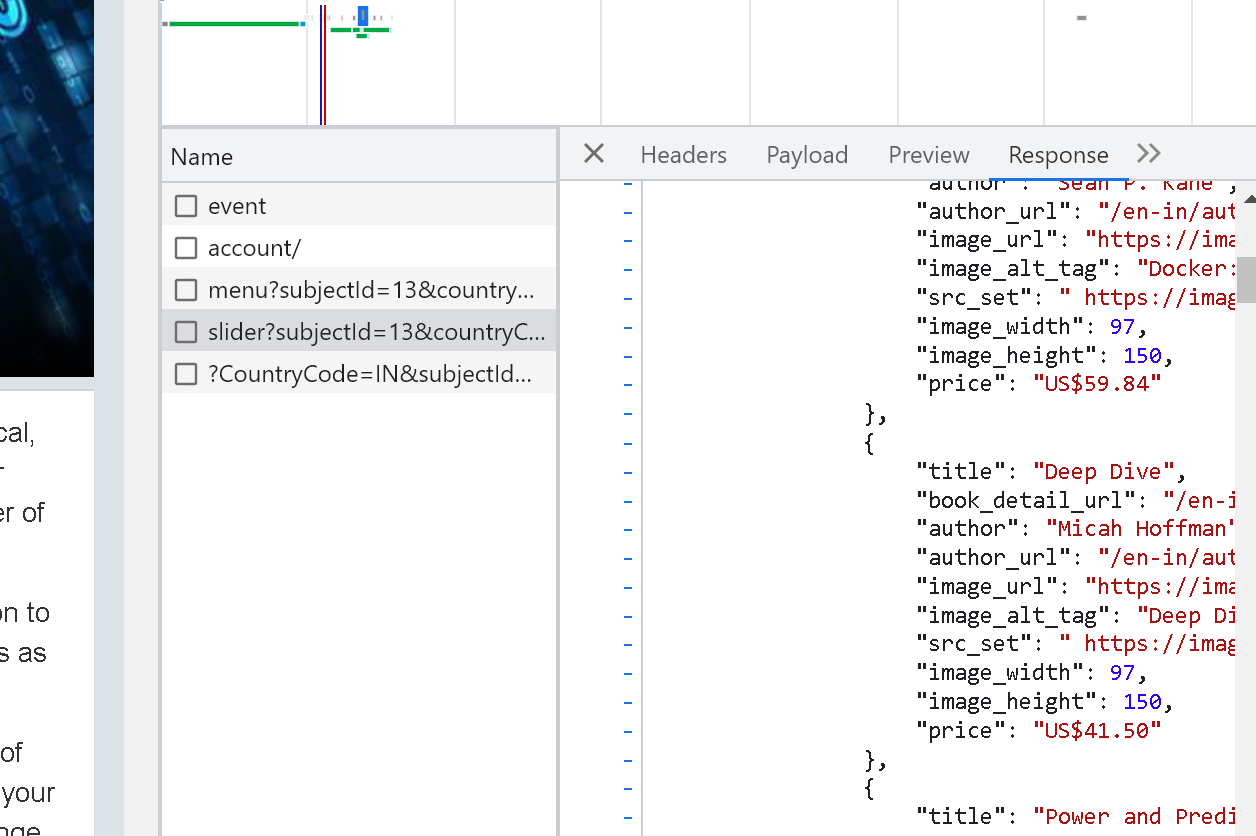




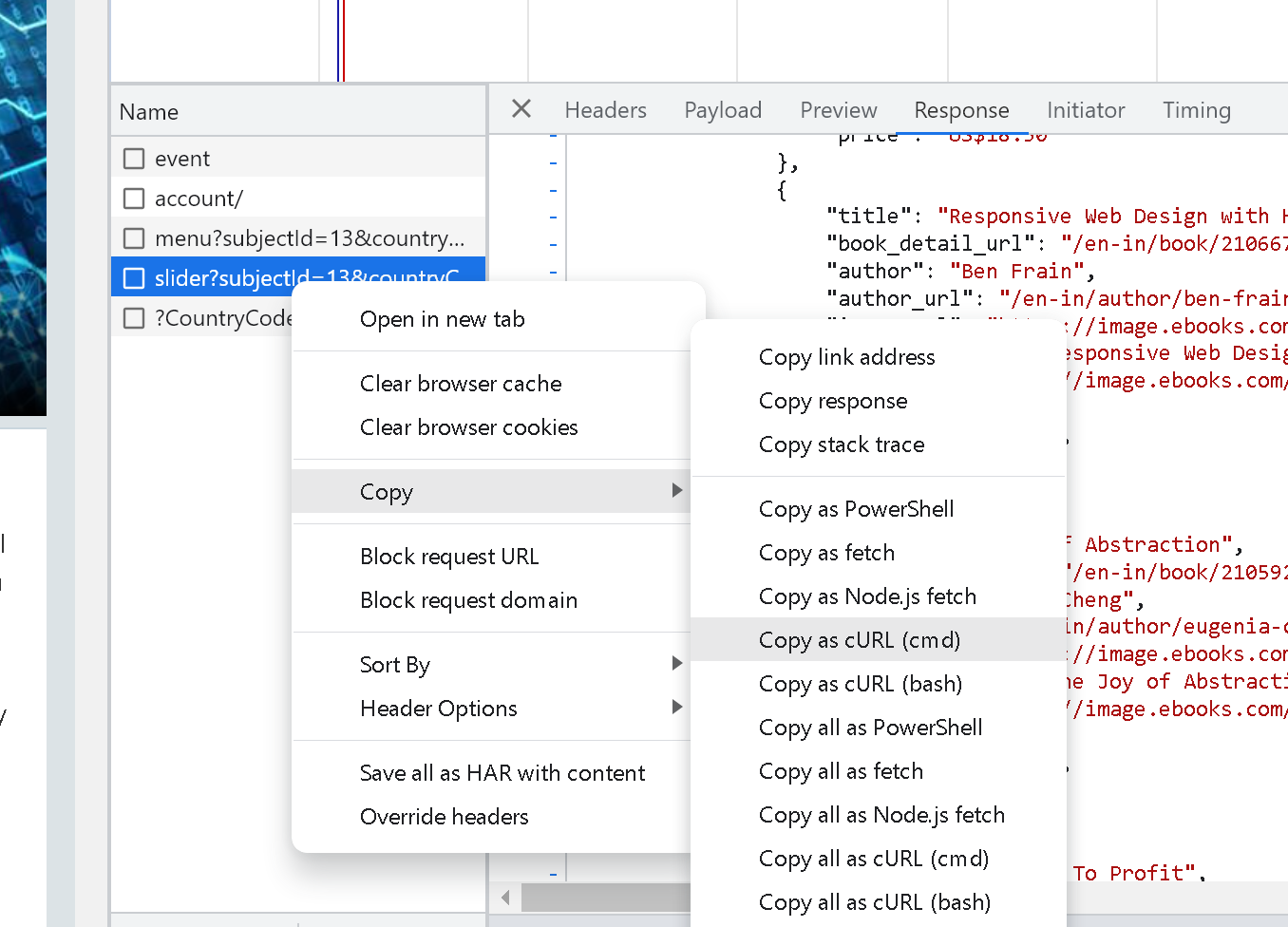
**Step\_4: Again go the first page, there u get the Network information**



**Step\_5: check once Response where you want to extract required data**



**Step\_6: Here you have to select proper name, check the preview once where the webpage information available i.e what you want to extract, Follow the path, to copy the command URL. Here my data available in 4th name.**

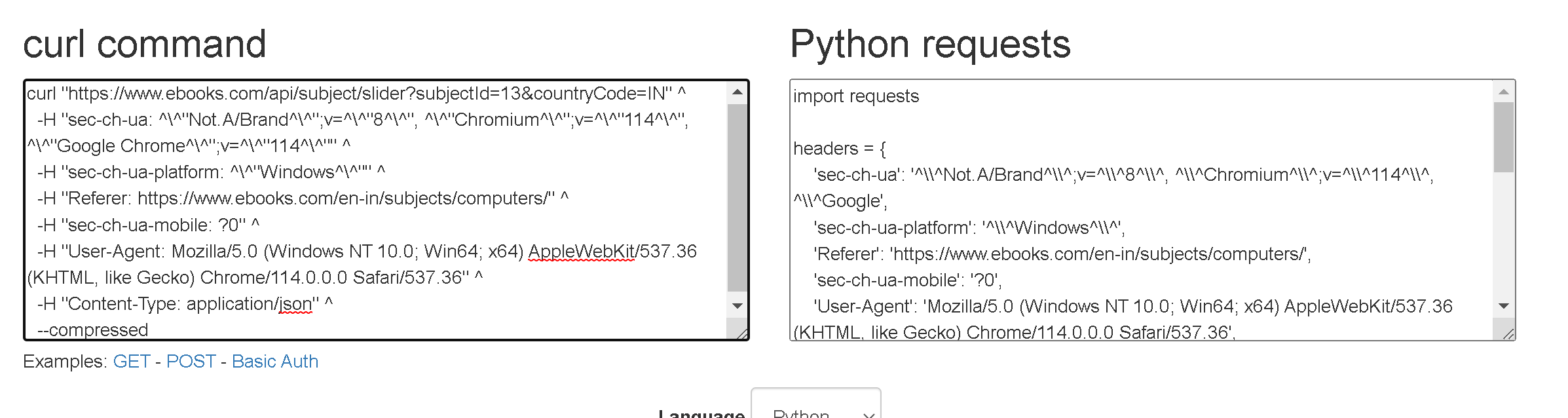


**Step\_7: Use Curl converter to convert the command URL to python code, click below page**





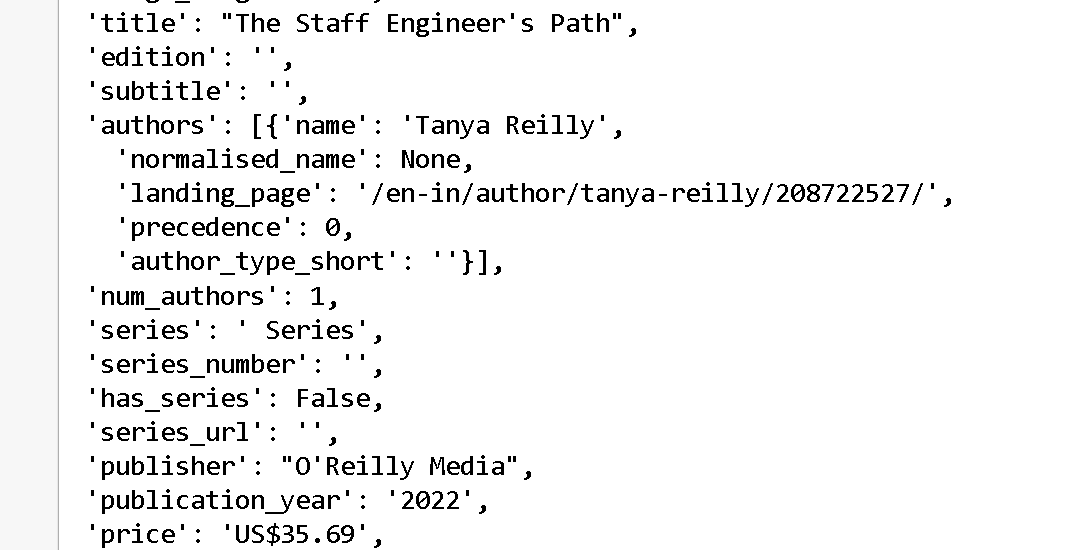
**Step\_8: Paste the copied URL in curl command box, it changed to python code**



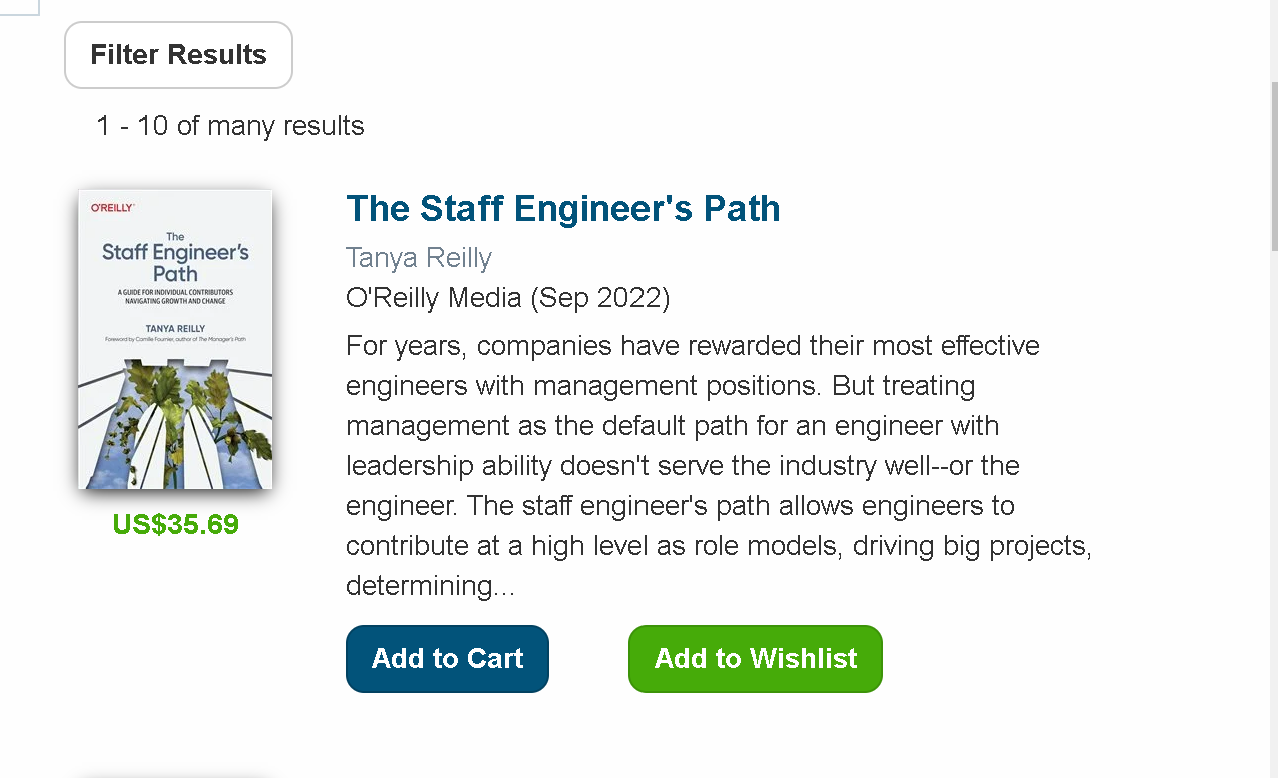
**Step\_9: Copy the Python request, Paste it in required jupyter note book, then request is created.**



**Step\_10: Results of the first book in jupyter & original webpage**









**Resources:**

ChatGPT

<https://youtu.be/kJyOp12PgP4>

<https://youtu.be/dqeuy58iQYo>

<https://youtu.be/O68xT4dE_zU>