- Search up something in arrow.com and see the search results
- Use requests to navigate to the page and extract the html from it
    - Make sure to use an appropriate header/user agent
        - i.e. `'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36'`
- Use inspect element to find html attributes from which you can pick out product pages
    - Find them with BeautifulSoup
- Take the patterns you found to collect all of the product urls
- Go into product pages and inspect element to find html attributes where you can pick the information you want
    - Find them with BeautifulSoup
    - Ensure this works across different products
- Collect the product data however you wish
- Iterate through the search results (For each page, look into each product on the page, then go to the next page)
- Store the data however you like

If search result pages do not have a common url naming scheme or has anti-scraping measures:

- We must use puppeteer
- Make sure to use puppeteer-stealth

```
const puppeteer = require('puppeteer-extra');
const StealthPlugin = require('puppeteer-extra-plugin-stealth');
puppeteer.use(StealthPlugin());
```

- Stealth is used for getting past the anti-botting measures
- We will use puppeteer to navigate to each page and retrieve the html data for the page.
- Go to the pages you want to scrape
    - If you need to click a button to go to the next page, use .click() to click on the html element
- You can collect the HTML data of the page you are on with

```
const data = await page.evaluate(() => document.querySelector('*').outerHTML);
```

- Afterwards, save it to a file so you can use it with your other scraper