Example: Web Scraping

Summary

The **demo.py** file in this repository shows how to scrape web pages using the `requests` and `bs4` (Beautiful Soup) modules along with the `.read_html()` method from Pandas. The pages scraped provide results for individual runners in the Mountain Goat, a popular annual run in Syracuse. Several years of results are collected before and after the acute phase of the Covid-19 pandemic to see whether it had a significant impact on race participation.

In addition to demonstrating the basics of web-scraping, the script also: (a) provides an example of the use of a SQLite database for storing results; (b) shows how a script can avoid re-downloading results if it is interrupted or stopped before it completes; and (c) demonstrates how to construct a stacked bar chart.

Input Data

There are no input files: key data about the pages to be scraped is built into **demo.py**.

Deliverables

None. This is an example only and there's **nothing due**.

Instructions

1. Open demo.py in Spyder or VS Code and adjust the `races` dictionary as desired to control the years of races retrieved. Then run the script and have a look at the PNG file it produces.
2. Load the resulting database `goat.db` into SQLiteStudio and browse through the tables.

Tips

- Beautiful Soup is a powerful and elegant module for extracting information from both HTML pages and XML files. Both are examples of *structured documents*, meaning that they have embedded tags indicating the logical structure of text. For example, in HTML a table begins with a `<TABLE>` "tag" and ends with a `</TABLE>` tag. To use Beautiful Soup, it helps to be familiar with the basics of HTML.
- For simple web pages that just contain one or more tables of data, `pd.read_html()` can often do the job by itself. It calls Beautiful Soup internally to find and parse the tables.
- One cautionary note: web pages that provide highly polished user interfaces often use the web language JavaScript for loading and presenting their data. In that case,the data is not present in the web page itself and can't be scraped in the usual sense. Instead, it would usually be necessary to examine the JavaScript in detail and write an alternative version that collects and stores the data.
- A second cautionary note: some web sites use session cookies (small bits of cryptographic data and other information) exchanged with client web browsers to protect against certain kinds of cyber attacks. The requests package can handle this by managing session cookies appropriately, but the details are beyond this exercise.