WORKSHOP WEB SCRAPING

OpenData Day - 07/03/26

O que é o Pyladies?

- Grupo internacional sem fins lucrativos: https://www.pyladies.com/
- Focado em aumentar a atividade e a liderança das mulheres na comunidade Python
- Missão: promover, educar e desenvolver uma comunidade de Python plural por meio de divulgação, educação, conferências, eventos e reuniões sociais.

Capítulo são paulo

- Criado em 7 de setembro de 2015
- Missão: incentivar mulheres a aprender, ensinar e motivar outras a conhecerem Python

Vamos nos conectar?

https://github.com/PyLadiesSP

www.linkedin.com/company/pyladiessp

@pyladies.saopaulo

"O pouco que você sabe pode ser muito para quem não sabe nada!"

E QUEM É VOCÊ?

Mini bio

- Olá, sou a Carol Cortez! Mãe de dois mimis.
- Sou jornalista por formação e programadora por profissão.
- Trabalhei por mais de dez anos na cobertura de economia em veículos como El País Brasil, Valor Econômico e Infomoney.
- Fiz transição de carreira na pandemia para a área de tecnologia e hoje trabalho como desenvolvedora backend no CPB (Comitê Paralímpico Brasileiro).
- Tenho 40 anos e muito a aprender e conquistar!

linkedin.com/in/ana-c-cortez

O que é webscraping?

Web scraping

- O webscraping é um termo em inglês que se refere à **raspagem de dados na internet**. Existe hoje uma infinidade de informações espalhadas pela internet, precisamos apenas aprender como extraí-las.

- Essa raspagem é feita por meio de **extração automatizada** de informações usando bots (softwares) para **coletar dados não estruturados** (HTML) e **convertê-los em formatos estruturados** (CSV, JSON, bancos de dados).

- Demanda pouco investimento. Tudo o que precisamos é de acesso à internet, conhecimento em programação e tempo.

O DOM da internet

- Para extrair dados de qualquer site na grande world wide web, precisamos antes entender como funciona a estrutura das páginas da internet.
- As páginas são estruturadas no que chamamos de DOM: Modelo de Objeto de Documento.
- Ele fornece uma representação estruturada do documento como uma árvore e define os métodos que permitem acesso a ela, para que possam alterar a estrutura, estilo e conteúdo do documento.
- Vamos, para resumir didaticamente, chamar o DOM de uma árvore de elementos, que geralmente vêm em HTML.

Inspecionar elementos

Para acessar o DOM da página, precisamos clicar na tecla F12 (ou clicar no botão direito do mouse sobre a página e em “Inspecionar”)

Selecionar elementos

- Cada elemento HTML da página ocupa um lugar no DOM. Para selecionar e interagir com esse elemento, precisamos encontrar o caminho dele nessa grande árvore.

- Muitos elementos têm características únicas que nos permite encontrá-los com mais facilidade, como um “id”.

- Se não tiver, localizamos pelo caminho parcial, com a ajuda dos elementos próximos.

Localizadores

//input[@id='search-bar']

Propriedade do elemento

Tipo do elemento


```
#Busca simples
input_pesquisa = driver.find_element("id",
"search-bar")
input_pesquisa.send_keys("material didático")
```

Elementos comuns: div, a, input, button, span, form, tb, td

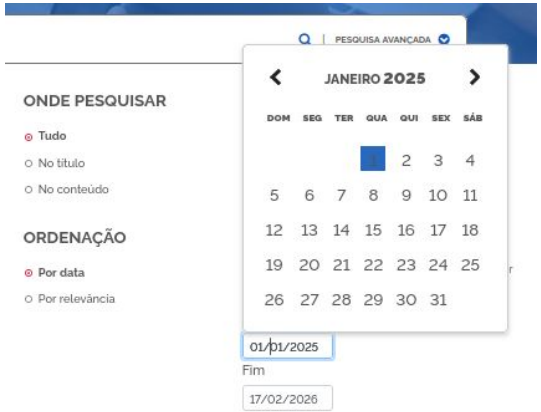
Propriedades comuns: id, class, text, name, type, placeholder

Abre caminhos - xpath

Caminhos (xpath)

- Infelizmente, nem tudo são flores: grande parte dos sites não são tão “semânticos”, principalmente com o crescimento de frameworks de front-end, como React e Angular.
- Nesses casos, precisaremos chegar aos elementos pelo “endereço” deles na árvore. O “xpath”.

```
//a[@class='ui-state-default' and normalize-space(text())='1']
```



Colinha de XPATH mais comuns

- Por atributo:
  HTML: <input name="senha">
  XPATH: //input[@name='senha']
- Pelo texto visível:
  HTML <button>Entrar</button>
  XPATH: //button[contains(text(),'Entrar')]
- Pelo class,quando não há id:
  HTML:
  XPATH: //div[contains(@class,'btn')]
- Pelos parentes próximos:
  HTML:

  <input name="email">

  XPATH: //div[@class='form']//input[@name='email']

OBS: O "//" significa que o elemento deve ser encontrado independente da posição dele na árvore. Para respeitar um caminho sequencial, utiliza-se somente "/"

Ex: //div[contains(text(), 'Resultados]]/p

Aqui, busca-se um parágrado (p) que esteja logo abaixo, na árvore do HTML, de uma div que tem o texto "Resultados" dentro dela.

Para buscar por textos, às vezes é melhor ignorar sua formatação. Para isso, tem a função normalize-space:

- Remove espaços no começo e/ou no final
- Troca vários espaços internos por apenas um

"//button[normalize-space()='PESQUISAR']"
Isso funciona em casos em que o texto dentro do elemento está assim: "\n PESQUISAR \n"

Nossas ferramentas

Python + Selenium

- Para este workshop, vamos utilizar a linguagem de programação **Python**, pois é rica em bibliotecas de coleta e análise de dados.
- O **Selenium WebDriver** é uma ferramenta que permite controlar um navegador automaticamente, como se fosse uma pessoa clicando, digitando e navegando.
- Ele abre um navegador real (Chrome, Firefox, Edge…) e executa ações na página, permitindo interações completas.

O Selenium permite:

- Abrir páginas
- Clicar em botões
- Preencher formulários
- Coletar dados da tela
- Testar sistemas

Mão na massa!!!!!!

Bora para o colab

https://colab.research.google.com/drive/1Fo2G-E9i8mcHFxXvU-uBN_DNRobHaUCy

Código no github

No repositório, você vai encontrar uma forma de rodar o código e visualizá-lo interagindo com o browser, pois utilizará uma máquina virtual com auxílio do Docker ;)

https://github.com/anacarolcortez/workshop-webscraping-docker

OBRIGADA :)