Crawler.py

1- DECISÕES:
O primeiro passo foi capturar os valores presentes em atributos das tags e depois os valores textuais. Com isso:

- *"Dificuldades surgiram, permaneceram e foram superadas." (Capt. Jack Sparrow)*

No meio do caminho tive problemas em capturar alguns valores de atributos, o que me faria mudar toda a lógica, então resolvi capturar esses valores da seguinte forma:

- A expressão **(**.title **>> nth=0)** busca exatamente a ocorrência que eu quero. Nesse exemplo, ele pega a primeira ocorrência. Eu utilizei o método **count()** pra saber quantas ocorrências existiam. Então, fiz um laço iterando até o valor de **count(**).

Em cada interação, ele incrementa o 'nth' (nth=0, nth=1, nth=2...) e faz um .append do valor do atributo que quero em **all_data**.

2 - CÓDIGO:
Variáveis que podem gerar dúvidas:

- **all_data** = Lista que armazena os dados de todos os produtos da página.
- **num** = Armazena quantas ocorrências especificadas no locator existem.
- **n** = Responsável por incrementar o valor de '>> nth=0'.
- **lenovo_data** = Lista com os itens filtrados.

No primeiro laço for, capturamos o modelo, link e estrelas (classificação) de cada ocorrência e armazenamos em **all_data** onde cada item é um dict de dados.

No segundo laço for, capturamos preço, descrição e reviews de cada ocorrência e armazenamos em **all_data**, completando as informações disponíveis do produto.

No terceiro laço for, filtramos de **all_data**, todos os dados onde o valor de model é referente a '**lenovo**' ou '**Thinkpad**', pois existem 3 resultados que satisfazem a especificação do desafio:

- Lenovo
- Thinkpad
- Lenovo Thinkpad

Então, armazenamos na lista lenovo data e inserimos no primeiro item da lista a quantidade de ocorrências encontradas. No final, formatamos e geramos um json bonitinho.

3 - MELHORIAS:
Existe muita brecha para otimização neste código:

- Eu poderia pegar os dados já filtrados, em vez de pegar todas as ocorrências e filtrar depois. Em valores altos de dados, pode exigir mais processamento. Porém, eu escolhi me atentar no contexto do desafio.
- Eu poderia ter desenvolvido uma lógica para o item “descrição” de cada ocorrência, separando por Tela, Processador, RAM, Armazenamento, SO e etc.
- Os valores de atributos poderiam ser capturados de uma maneira mais simples. Mas, desse jeito também funciona. :D
- A funcionalidade principal poderia ser feita em um módulo .py a parte é importada na minha função Flask. Eu escolhi deixar tudo em um arquivo só para ficar mais elegante.