Relatório card 8 – 8 - Prática: Web Scraping com Python p/ Ciência de Dados (II)

Leonardo José Reis Pinto

1. Introdução

O relatório aborda a raspagem de dados através de bibliotecas da linguagem Python, como o BeautifulSoup, que é muito importante na ciência de dados, pois é fundamental para a obtenção de informações da internet de forma eficiente.

2. Beautiful Soup

Uma biblioteca em Python que, através de tags HTML e XML, permite extrair dados de sites de forma eficiente. Seus principais métodos incluem find() e find_all() para localizar elementos por tag, atributos ou classe, e Select() para utilizar seletores CSS.

Eu desenvolvi um exemplo de código para extrair os nomes das placas de vídeo do Mercado Livre e seus preços. Para isso, criei duas listas para armazenar os dados e organizei essas informações em um DataFrame para melhor visualização. A seguir, compartilho alguns prints:

```
[1]  import requests
     from bs4 import BeautifulSoup
     import pandas as pd
```

Importei as bibliotecas necessária

```
# Fazendo a requisição para a página de produtos
url = 'https://lista.mercadolivre.com.br/placa-de-video#D[A:placa%20de%20video]'
response = requests.get(url)
html = response.text

# Criando o objeto Beautiful Soup
soup = BeautifulSoup(html, 'html.parser')

# Buscando todos os elementos que contêm os preços dos produtos
preco = soup.find_all('div', class_="poly-component__price")

# Extraindo e imprimindo os preços
lista_precos = []  # Lista única para armazenar todos os valores

for elemento in preco:
    lista_precos.append(elemento.text.strip())  # Adiciona o texto limpo à lista

print(lista_precos)  # Exibe a lista única com todos os valores
```

Extraindo os preços do site do Mercado Livre um por um e alocando-os na lista criada para manipulá-los depois de forma mais fácil.

```
# Fazendo a requisição para a página de produtos
url = 'https://lista.mercadolivre.com.br/placa-de-video#D[A:placa%20de%20video]'
response = requests.get(url)
html = response.text

# Criando o objeto Beautiful Soup
soup = BeautifulSoup(html, 'html.parser')

# Buscando todos os elementos que contêm os títulos dos produtos
produto = soup.find_all('h2', class_="poly-box poly-component__title")

lista_produto = []

# Extraindo e imprimindo os títulos
for produto in produto:
    lista_produto.append(produto.text.strip())

print(lista_produto)
```

Fiz a mesma coisa para os nomes dos produtos

Relatório Bootcamp Lamia

```python
# esta verificando  o menor tamanho entre as listas
tamanho_minimo = min(len(lista_produto), len(lista_precos))

# Ajusta as duas listas para o mesmo tamanho
lista_produto = lista_produto[:tamanho_minimo]
lista_precos = lista_precos[:tamanho_minimo]

# Criando um data frame para demonstrar o resultado da raspagem
df = pd.DataFrame({
    'Nome da Peça': lista_produto,
    'Preço': lista_precos
})

print(df)
```

```
                                                                                                Nome da Peça
0                                                    Placa De Vídeo R7 240 Ddr5 2gb Vga Hdmi Amd Pcie X16 3.0        R$359,90R$338,306% OFFem 12x R:
1                                                      Placa De Vídeo Gamer Amd Radeon Rx580 Gddr5 8g Hdmi Dp                  R$804,99em 12x R:
2  Placa de vídeo Nvidia MSI  Ventus XS GeForce GTX 16 Series GTX 1650 GEFORCE GTX 1650 D6 VENTUS XS OC OC Edition 4GB  R$1.148R$975,8015% OFFem 10x R$97,58 sem
3                                                   Placa De Vídeo Msi Geforce Gtx 1650 Ventus Xs Ocv3 4gb D6  R$1.247R$892,2528% OFFem 10x R$89,22 sem
4                                                    Placa de vídeo AMD Radeon RX 580 8GB DDR5 256 BIT 2048SP                   R$714em 12x R:
5                                                   Placa De Vídeo Evolut Geforce Gt610 2gb Ddr3 Dvi/hdmi/vga         R$168R$149,9010% OFFem 12x R:
6                                                  Placa De Vídeo Evolut Geforce G210 1gb Ddr3 Dvi-i/vga/hdmi        R$200,50R$120,3040% OFFem 12x R:
7                                                  Placa De Vídeo Gt610 Ddr3 2gb Vga Hdmi Nvidia Pcie X16 2.0                     R$153em 12:
```

criei um data frame com as listas criadas.

Usando o scrip do vídeo:

Aqui foi analisado um arquivo estático (`home.html`) para identificar e exibir informações específicas, como títulos e preços de cursos.

```python
from bs4 import BeautifulSoup
import requests
import time


# Utilizando open para abrir o arquivo home.html para leitura('r'), como arquivo html
with open('home.html', 'r') as html_file:
    # Atribuindo o arquivo o conteúdo para content
    content = html_file.read()

    # atribuindo a soup, o metodo BeautifulSoup do conteúdo de content no formato lxml
    soup = BeautifulSoup(content, 'lxml')

    # Imprime o código de modo agrádavel
    print(soup.prettify())

    # Localiza todas as tags h5 do site
    course_html_tags = soup.find_all('h5')

    # For para percorrer as tags h5 e imprimi-las
    for course in course_html_tags:
        print(course.text)

    # Localiza todos as tags div com a classe card
    course_cards = soup.find_all('div', class_ = 'card')

    # Percorre todos os cards
    for course in course_cards:

        course_name = course.h5.text # Course_name recebe o titulo h5
        course_price = course.a.text.split()[-1] # separa o texto e recebe a ultima string, referente ao preço
```

já aqui foi implementado um script para acesso ao site timesjobs , permitindo a coleta e filtragem de informações sobre vagas de emprego, com base em critérios definidos pelo usuário.

```
unfamiliar_skill = input('>')
print(f'Filtering out {unfamiliar_skill}')

# Acessando o código do site
html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&searchTextSrc

# Atribuindo a soup o código do site utilizando o metodo BeautifulSoup no formato lxml
soup = BeautifulSoup(html_text, 'lxml')

# Localiza todos os trabalhos, tag li e classe clearfix job-bx wht-shd-bx
jobs = soup.find('li', class_ = 'clearfix job-bx wht-shd-bx')

# Função para navegar no site e imprimir os trabalhos disponíveis e recém-postados
def find_jobs():
    # for para percorrer trabalhos
    for job in jobs:
        # Localiza a data de publicação do trabalho como texto
        published_date= job.find('span', class_ = 'sim-posted').span.text
        # if para filtrar somente as datas few
        if 'few' in published_date:

            # Localiza o nome da empresa como texto sem espaços em branco
            company_name = job.find('h3', class_ = 'joblist-comp-name').text.replace(' ', '')

            # Localiza as habilidades necessárias do trabalho
            skills = job.find('span', class_ = 'srp-skills').text.replace(' ', '')

            # Localiza o link para obter mais informções do trabalho
            more_info = job.header.h2.a['href']

            # if para imprimir somente os trabalhos que não possuem a habilidade que o usuário não possui
            if unfamiliar_skill not in skills:
                print(f"Company Name: {company_name.strip()}")
                print(f"Required Skills: {skills.strip()}")
                print(f'More info: {more_info}')
```

3. Conclusão

Com essa biblioteca de web scraping, ficou bem mais fácil e eficaz realizar extrações de dados, sendo uma ferramenta importante para diversas áreas, desde o marketing até restaurantes, permitindo coletar informações da internet e tratá-las para gerar insights poderosos.

4. Referências

https://www.crummy.com/software/BeautifulSoup/bs3/documentation.html

https://www.youtube.com/watch?v=XVv6mJpFOb0