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

Leonardo José Reis Pinto

1. Introdução

O relatório fala sobre a raspagem de dados atraves de bibliotecas da linguagem python como o beatfulsouap, muito importante na ciencias de dados, pois é fundamental para obtenção de dados da internet de forma eficiente.

2. Códigos do vídeo

O relatório aborda os scripts gerados pelo vídeo do freecodecamp sobre web scraping usando a biblioteca beatiful soap para extrair dados nas tags htmls dos sites, o scrip inicial fornecido pelo vídeo ele acessa o conteúdo de um arquivo (html.file) é carregado e processado pelo BeautifulSoup no formato lxml. A função prettify()é usada para exibir o HTML de forma estruturada e legível.posteriormente o método find_all()localiza todas as ocorrências de tags, como <h5>, permitindo o acesso ao seu conteúdo textual. Além disso, ele é usado para buscar elementos com classes específicas facilitando a remoção de informações fornecidas, como nomes e preços de cursos.

```python
# 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

        #imprime o nome e preço do curso
        print(f'{course_name} costs {course_price}')
```

```
        '
         Become a Python Machine Learning master!
        </p>
        <a class="btn btn-primary" href="#">
         Start for 100$
        </a>
       </div>
      </div>
     </body>
    </html>

    Python for beginners
    Python Web Development
    Python Machine Learning
    Python for beginners costs 20$
    Python Web Development costs 50$
    Python Machine Learning costs 100$
```

Posteriormente, o código utiliza o BeautifulSoup para processar o HTML de uma página da web obtida via requisição HTTP com a biblioteca requests. O método find()é empregado para localizar elementos únicos, como listas de vagas de emprego com classes específicas. Dentro de uma função (find_jobs()), métodos como find()e textsão usados para extrair informações como dados de publicação, nomes de empresas e habilidades obrigatórias, enquanto condições iffiltram os resultados com base em critérios definidos pelo usuário.

```python
# Texto para pedir ao usuário qual habilidade e não é familiar para que ela seja filtrada e não selecionada
print('Put some skill that you are not familiar with')
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&s

# 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']
```

```
            # 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}')

                print('')

# If para a criação da automatização, se o programa for executado na main.
if __name__ == '__main__':
    # Enquanto ele estiver sendo executado...
    while True:
        find_jobs() # Função para coletar e exivir os artistas
        time = 10 # Intervalo de execução
        print(f'Waiting {time} minutes...')
        time.sleep(time + 60) # Definindo a espera pelo intervalo de execução
```

3 - Código Prática

Agora, na parte prática, eu fiz um web scraping na página do Yahoo Finance, com o objetivo de extrair e exibir o preço atual do índice S&P 500. No início, o conteúdo HTML da página é obtido por meio de uma requisição HTTP com a biblioteca requests, utilizando um cabeçalho para simular um navegador e evitar bloqueios. Posteriormente, analise a estrutura HTML no formato lxml e localizo o elemento específico que contém o preço, identificado pela tag <fin-streamer>e o atributo data-fieldigual a 'regularMarketPrice'. Caso o preço seja encontrado, ele é extraído e exibido no console; caso contrário, uma mensagem informa que o preço não foi localizado. O código inclui uma automação na função principal que executa a busca repetidamente a cada intervalo de 1 minuto, permitindo monitorar continuamente o preço em tempo real.

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

# URL do Yahoo Finance
url = 'https://finance.yahoo.com/quote/%5EGSPC/'

# Função para buscar o preço do índice
def busca_de_preco():
    # Obtém o conteúdo da página
    html_text = requests.get(url, headers={'User-Agent': 'Mozilla/5.0'}).text

    # Analisa o HTML com BeautifulSoup
    soup = BeautifulSoup(html_text, 'lxml')

    # Localiza o preço
    sem_preco = soup.find('fin-streamer', attrs={'data-field': 'regularMarketPrice'})

    if sem_preco:
        preco = sem_preco.text.strip()
        print(f"S&P 500 Price: {preco}")
    else:
        print("preço nao encontrado.")

# If para automatização
if __name__ == '__main__':
    while True:
        busca_de_preco()  # Chama a função para buscar o preço
        intervalo_tempo = 1  # Intervalo em minutos
        print(f'Aguarde {intervalo_tempo} minuto(s)...')
        time.sleep(intervalo_tempo * 60)  # Aguarda o intervalo antes de repetir
```

OUT PUT:

```
S&P 500 Price: 5,946.33
Aguarde 1 minuto(s)...
```

4. Conclusão

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

5. Referências

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