Relatório 08 - Web Scraping com Python p/ Ciência de Dados

Guilherme Loan Schneider

Descrição da atividade

A biblioteca BeautifulSoup é usada para analisar documentos HTML e XML, permitindo que você navegue pela estrutura da página de forma simples. Ela é útil para localizar e extrair elementos específicos de uma página, como títulos, links ou tabelas. Com o BeautifulSoup, é possível buscar elementos usando seletores como tags, classes e IDs, bem como percorrer a hierarquia de elementos HTML.

Por exemplo, se você tiver uma página HTML, o BeautifulSoup pode ajudar a encontrar todos os links (tags <a>) ou textos dentro de uma tag específica. Ele possui métodos como find() e find_all() para buscas específicas, além de permitir manipulação da estrutura da página.

Outra importante biblioteca utilizada foi o “LXML”, que pode ser usado diretamente para manipulação de documentos ou como um parser para o BeautifulSoup. Além disso, o lxml suporta tanto HTML malformado quanto XML estruturado.

Na imagem abaixo, o autor utilizou a biblioteca BeautifulSoup em um arquivo HTML simples para demonstrar as formas de resgatar informações de uma página.

```python
from bs4 import BeautifulSoup

# with open('Aula 8/home.html', 'r') as file:
#     content = file.read()

#     soup = BeautifulSoup(content, 'lxml')
#     # headers = soup.find('h5')
#     courses_html_tags = soup.find_all('h5') # retorna uma lista de tags h5 que contem os nomes dos cursos

#     # Melhorando a visualização dos cursos
#     for course in courses_html_tags:
#         print(course.text)

with open('Aula 8/home.html', 'r') as file:
    content = file.read()

    soup = BeautifulSoup(content, 'lxml')
    course_cards = soup.find_all('div', class_='card')
    # A utilização do class_ é para evitar conflitos com a palavra reservada class do Python,
    # fazendo com que o BeautifulSoup entenda que estamos nos referindo a um atributo de classe do HTML

    for course in course_cards:
        course_name = course.h5.text
        course_price = course.a.text.split()[-1]
        print(f'{course_name} costs {course_price}')
```

Na aplicação em um website real, foi fundamental utilizar a blbioteca “requests”, que obtêm o HTML de uma página para posteriormente analisá-lo com o BeautifulSoup. Ela simula como se fosse um indivíduo acessando um site e requisitando informações, como pesquisa, cliques em conteúdo do site, dentre outros.

Abaixo há o exemplo de uma aplicação em um site real chamado “TimesJobs”, utilizando a biblioteca requests, que acessa o site e resgata as informações definidas n código. Em seguida a execução do código que salva as aplicações procuradas em um arquivo TXT.

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

print('Insira habilidades que você não é familiarizado: ')
unfamiliar_skills = input('>')
print(f'Filtrando vagas que não contém: {unfamiliar_skills}')

def find_jobs():
    html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&fro
    soup = BeautifulSoup(html_text, 'lxml')
    jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')

    for index, job in enumerate(jobs):
        published_date = job.find('span', class_='sim-posted').span.text
        if 'few' in published_date:
            company_name = job.find('h3', class_='joblist-comp-name').text.replace(' ', '')
            skills = job.find('div', class_='more-skills-sections').text.replace(' ', '')
            more_info = job.header.h2.a['href']
            if unfamiliar_skills not in skills:
                with open(f'posts/{index}.txt', 'w') as f:
                    f.write(f'Company Name: {company_name.strip()} \n')
                    f.write(f'Required Skills: {skills.strip()} \n')
                    f.write(f'More Info: {more_info}')
                print(f'File saved: {index}')

if __name__ == '__main__':
    while True:
        find_jobs()
        print('Filtrando vagas a cada 15 minutos...')
        time.sleep(900)
```

```
PS G:\Meu Drive\UTFPR\LAMIA\Aula 8> python .\TimesJobs.py
Insira habilidades que você não é familiarizado:
>linux
Filtrando vagas que não contém: linux
File saved: 3
File saved: 4
File saved: 5
File saved: 7
File saved: 8
File saved: 10
File saved: 11
File saved: 12
File saved: 13
File saved: 14
File saved: 15
File saved: 16
File saved: 17
File saved: 19
File saved: 21
File saved: 22
File saved: 23
File saved: 24
Filtrando vagas a cada 15 minutos...
```

Conclusões

Foi possível compreender que a utilização dessas bibliotecas permite que sejam recuperadas informações de forma rápida de websites variados, basta apenas analisar de forma rápida como estão estruturadas as informações do site escolhido. Tudo isso combinado com a utilização das três bibliotecas citadas anteriormente a BeautifulSoup, LXML e Requests.

Referencias

Web Scraping with Python - Beautiful Soup Crash Course