Relatório 8 - Web Scraping com Python p/ Ciência de Dados

Lucas Scheffer Hundsdorfer

Descrição da atividade

Nesta seção nos é ensinado um pouco de Web Scraping que é basicamente uma coleta de dados de sites da web de forma automatizada, usada para quem deseja ter acesso a mais dados para tomarem decisões mais inteligentes.

No começo do vídeo ele ensina um pouco sobre as tags do html, pois vai ser útil para a coleta de dados, e nos ensina a baixar as bibliotecas necessárias, como a 'beautifulsoup' e a 'lxml'.

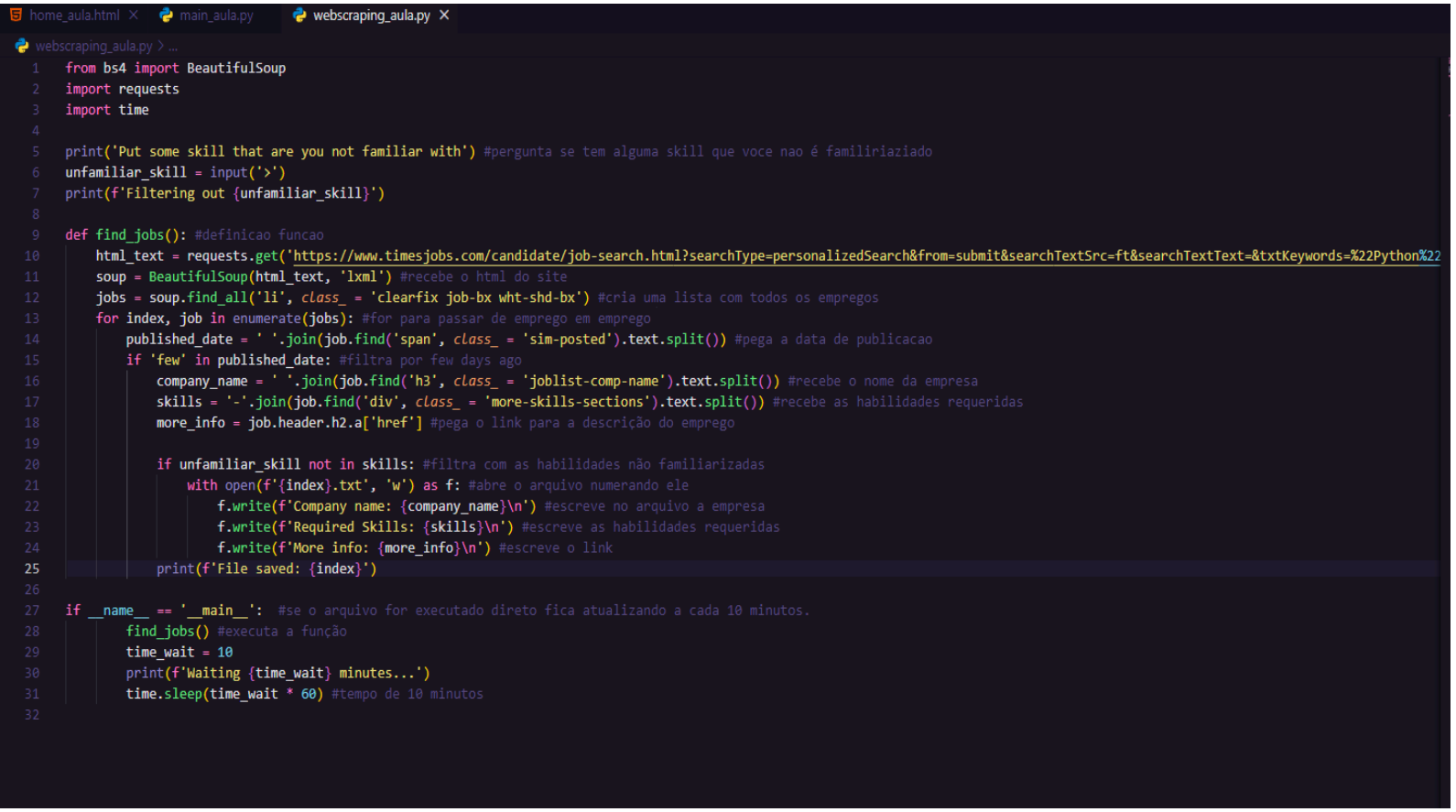
A primeira atividade ele traz um website que ele disponibiliza, bem básico apenas contém 3 'cursos' e seus preços, e vai sendo ensinado o básico de como coletar os dados dos sites utilizando as tags do html, o código final ficou assim:

home_aula.html | main_aula.py

main_aula.py > ...

```python
from bs4 import BeautifulSoup

#aqui ele abre o arquivo apenas lendo o que tem nele e printa todo o conteudo
with open('home_aula.html', 'r') as html_file:
    content = html_file.read()
    print(content)

    print('')

    #printa o html de forma mais agradável
    soup = BeautifulSoup(content, 'lxml')
    print(soup.prettify())

    print('')

    tags = soup.find('h5')  #printa apenas a primeira tag h5 que achar
    print(tags)

    print('')

    #retorna uma lista com todos os h5
    course_html_tags = soup.find_all('h5')
    for course in course_html_tags:  #for each para pegar apenas os cursos dentro da lista
        print(course.text)

    print('')

    course_cards = soup.find_all('div', class_='card') #acha toda div que tem como classe card
    for course in course_cards:
        course_name = course.h5.text #pega todos os nomes dos cursos
        course_price = course.a.text.split()[-1] #pega apenas o valor numérico do curso

        print(f'{course_name} costs {course_price}') #printa o nome do curso e quanto custa
```

A segunda atividade se baseia no site https://www.timesjobs.com/ é um código que recebe todo o arquivo html dele filtrado por python já, e faz o seguinte processo, pergunta se tem alguma habilidade que você não é familiarizado, e após isso ele filtra todos os empregos que foram publicado há alguns dias atrás e filtra pegando apenas os empregos que não solicitam a habilidade que você não é familiarizado, cria arquivos de texto enumerados com os empregos e dentro escreve o nome da empresa as habilidades necessárias e o link para maiores informações, se o arquivo for rodado diretamente, ele fica atualizando os arquivos a cada 10 minutos.



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

print('Put some skill that are you not familiar with') #pergunta se tem alguma skill que voce nao é familiriaziado
unfamiliar_skill = input('>')
print(f'Filtering out {unfamiliar_skill}')

def find_jobs(): #definicao funcao
    html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&searchTextSrc=ft&searchTextText=&txtKeywords=%22Python%22
    soup = BeautifulSoup(html_text, 'lxml') #recebe o html do site
    jobs = soup.find_all('li', class_ = 'clearfix job-bx wht-shd-bx') #cria uma lista com todos os empregos
    for index, job in enumerate(jobs): #for para passar de emprego em emprego
        published_date = ' '.join(job.find('span', class_ = 'sim-posted').text.split()) #pega a data de publicacao
        if 'few' in published_date: #filtra por few days ago
            company_name = ' '.join(job.find('h3', class_ = 'joblist-comp-name').text.split()) #recebe o nome da empresa
            skills = '-'.join(job.find('div', class_ = 'more-skills-sections').text.split()) #recebe as habilidades requeridas
            more_info = job.header.h2.a['href'] #pega o link para a descrição do emprego

            if unfamiliar_skill not in skills: #filtra com as habilidades não familiarizadas
                with open(f'{index}.txt', 'w') as f: #abre o arquivo numerando ele
                    f.write(f'Company name: {company_name}\n') #escreve no arquivo a empresa
                    f.write(f'Required Skills: {skills}\n') #escreve as habilidades requeridas
                    f.write(f'More info: {more_info}\n') #escreve o link
            print(f'File saved: {index}')

if __name__ == '__main__':  #se o arquivo for executado direto fica atualizando a cada 10 minutos.
        find_jobs() #executa a função
        time_wait = 10
        print(f'Waiting {time_wait} minutes...')
        time.sleep(time_wait * 60) #tempo de 10 minutos
```

Conclusões

O web scraping é uma ótima forma de coletar dados facilitando muito o trabalho, ele automatiza várias funções que fazendo manualmente seria muito demorado. É uma ótima forma de tratar os dados.

Referências

Web Scraping with Python - Beautiful Soup Crash Course