Relatório <8 - Pratica> - < Web Scraping com Python p/ Ciência de Dados (II)>

<Ronny Gabryel Colatino de Souza>

Descrição da atividade

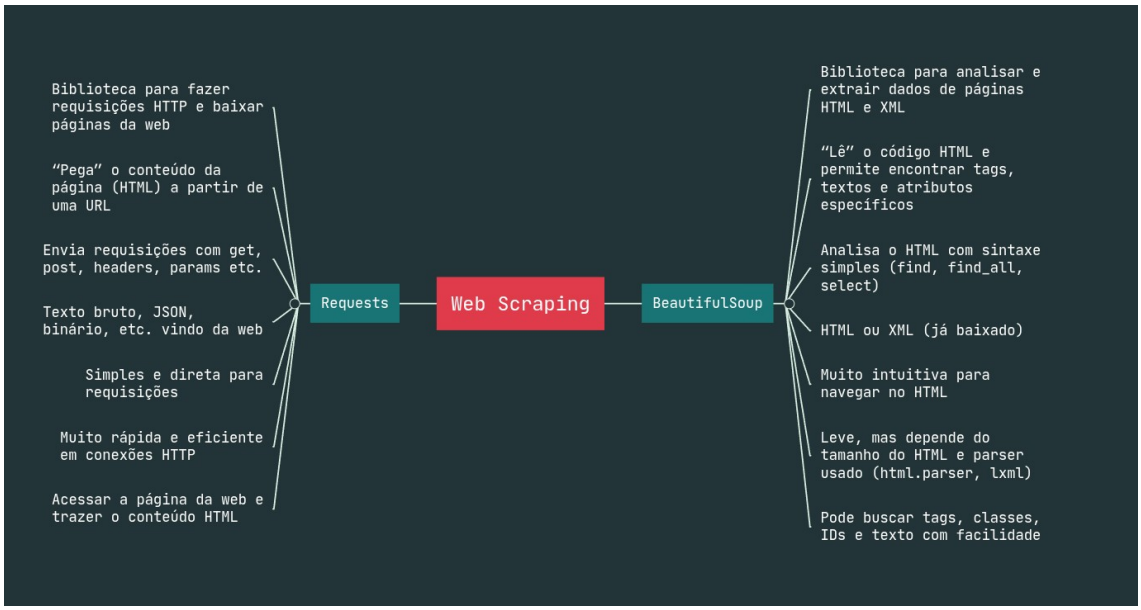
Esse card foi sobre aprender Web Scraping que basicamente é conseguir pegar informações de sites de forma estruturada pra poder usar esses dados depois. A ferramenta principal pra isso é a biblioteca BeautifulSoup que serve justamente pra extrair dados de arquivos html e xml de um jeito bem mais fácil

PyCharm

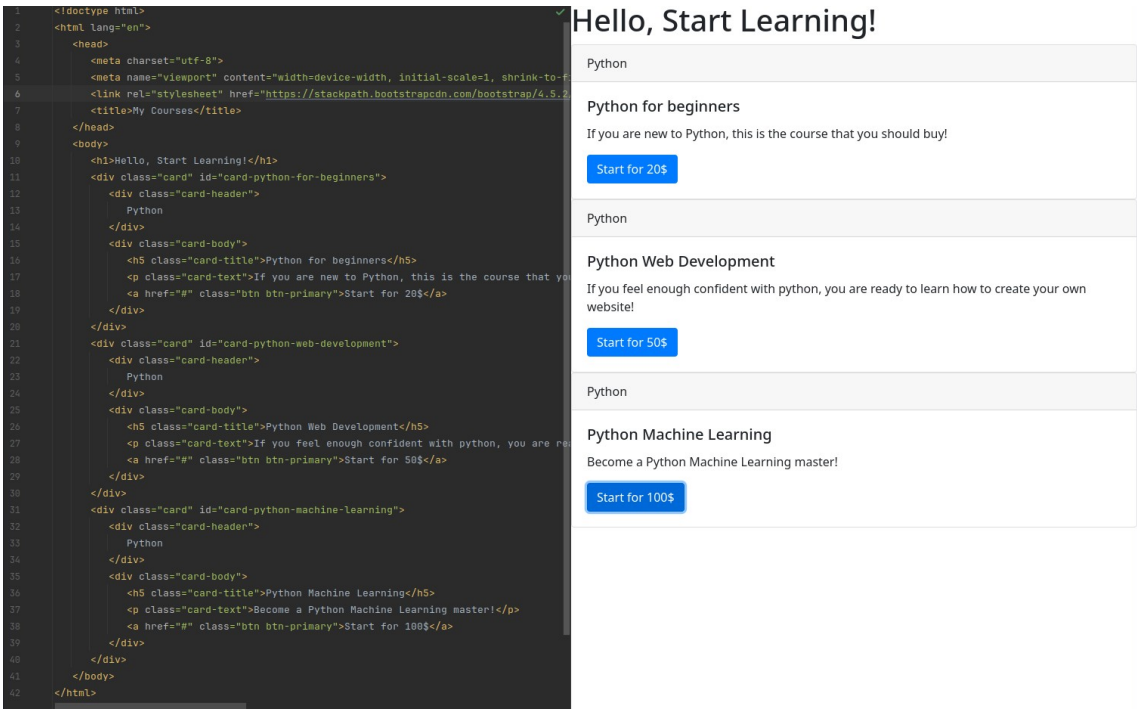
Uma coisa interessante desse card foi que acabei usando o PyCharm pela primeira vez, nem sabia da existência dessa IDE, achei ela bem mais simples comparada com o VScode tipo mais direta pro que precisa. No começo tive algumas dificuldades pra mexer nela mas nada de outro mundo, peguei uns tutoriais por fora de como navegar na IDE e em pouco tempo já tava bem melhor gostei bastante da forma que ela funciona parece bem pensada pra desenvolvimento em Python especificamente

BeautifulSoup e Requests

Logo de cara foram apresentadas duas bibliotecas principais a BeautifulSoup que faz a leitura do arquivo html e extrai os dados de forma bem otimizada, e a requests que serve pra pegar o arquivo html da página direto pela url no começo achei que seria mais complicado mas essas duas juntas resolvem tudo realmente se complementam e são bem simples de entender



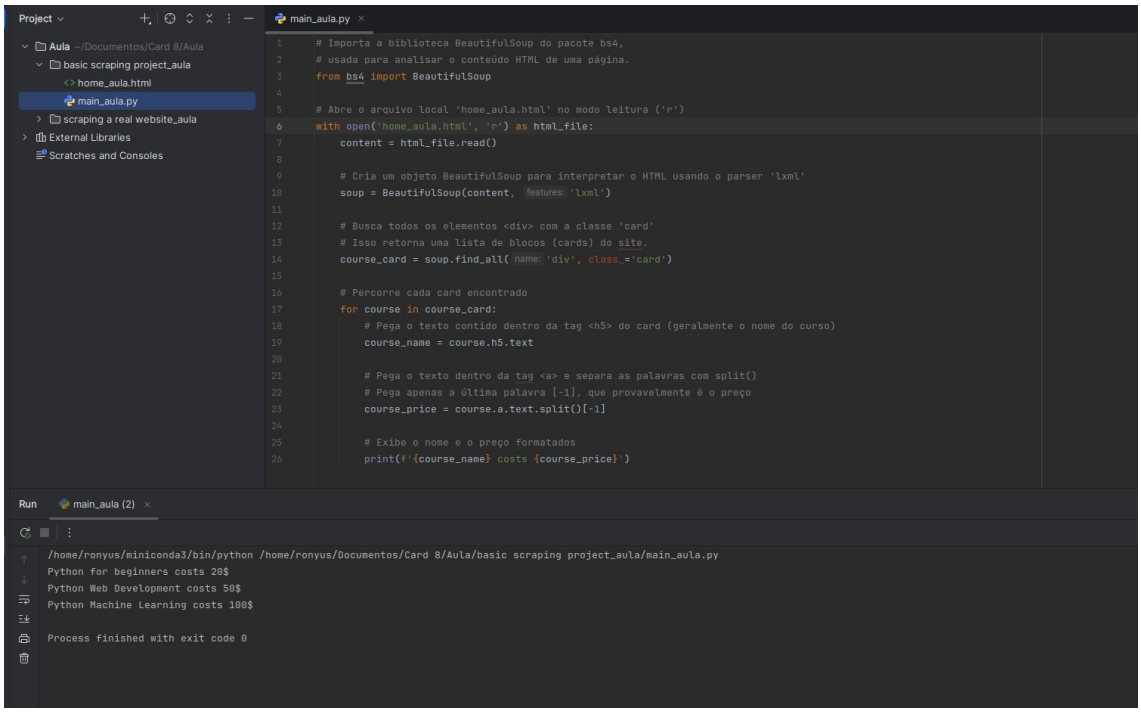
Primeiro código:



O primeiro codigo foi só com a BeautifulSoup mesmo usando um arquivo html que já esatava em minha maquina o objetivo era ler os cards de atividades que aparecem no arquivo através de divs. A parte mais importante aqui foi aprender a usar a ferramenta de inspecionar do navegador pra descobrir onde estava cada coisa

Quando você passa o mouse sobre as divs dá pra perceber que cada card de curso é uma div com a classe 'card', e dentro dessas divs tão os itens de cada curso. No código o foco foi extrair especificamente o título do curso que tá no h5 e o preço que tá no link 'a', usando o atributo .text pra pegar o conteúdo deles

Essa parte de analisar os elementos é tipo a base de tudo sem isso é impossível fazer qualquer filtragem depois então depois de achar todas as divs com a função find_all(), foi feito uma filtragem onde pra cada div a gente pega o h5 (título do curso) e usa o .split()[-1] no texto do link pra pegar só a última palavra que é o preço, e printa esses valores na tela formatados



Segundo código

O segundo código foi bem parecido mas agora usando a biblioteca requests pra pegar o html da página direto pela url a função get recebe a url do site timejobs.com como string e já usa o atributo .text pra puxar só o texto bem prático

Foi feita a leitura com beautifulsoup de novo mas dessa vez achando todas as listas (li) com a classe 'clearfix job-bx wht-shd-bx' através do find_all() essas listas eram os cards de emprego do site é claro que de novo foi necessário analisar o site pra achar a melhor forma de filtrar os elementos desejados.

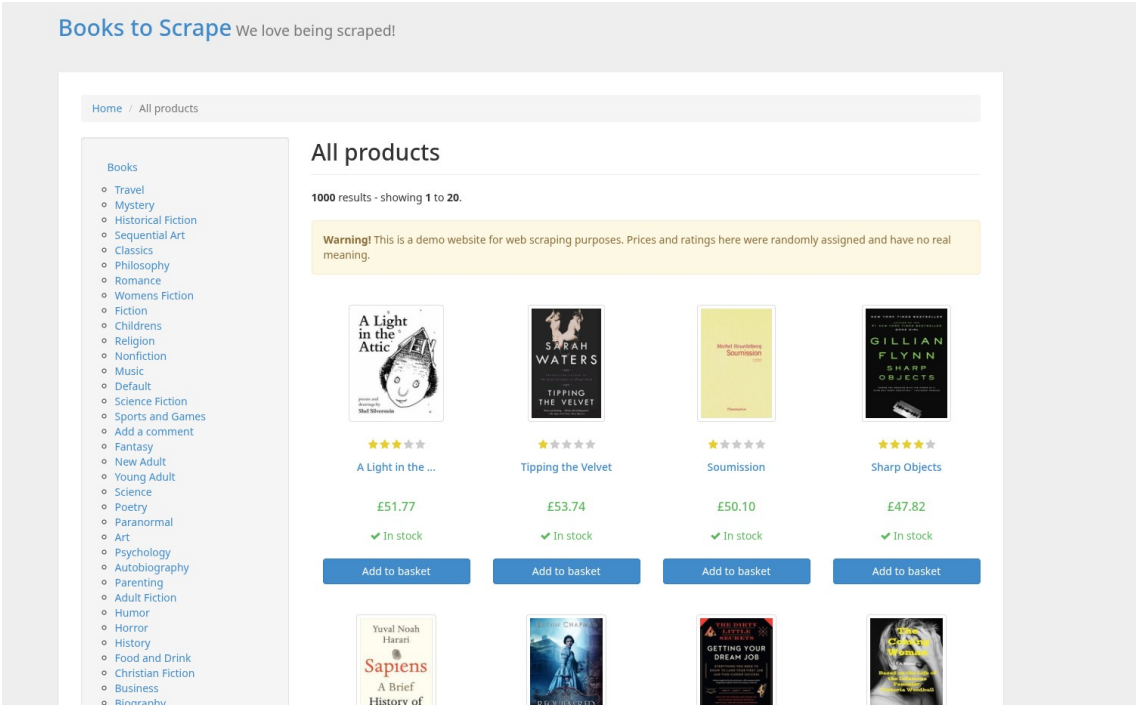
No código foi extraído o span com a data de publicação o h3 com o nome da empresa, uma div com as skills pedidas e o link através do href tratado como dicionário usando colchetes ['href']. Aqui foi usado o replace(' ', '') pra tentar limpar os espaços dos textos, embora não tenha funcionado muito bem e precisou usar.

O código pede pro usuário digitar uma skill que ele não conhece e filtra pra mostrar só vagas que não tem aquela skill. Também filtra pra pegar só vagas recentes que tem 'few' na data de publicação no final salva tudo em arquivos .txt dentro de uma pasta 'posts' numerados por índice e tem um loop com time.sleep que roda o código a cada 10 minutos automaticamente

Meu codigo

Depois de entender como funcionava o Web Scraping com os exemplos do card fui atrás de fazer meu próprio código pra praticar procurei sites com URL simples pra não complicar muito no começo pesquisei bastante e achei o books.toscrape.com que é bem

tranquilo de trabalhar A URL era bem simples e dava pra brincar com tudo sem muita dor de cabeça então escolhi fazer um buscador de livros.



A ideia era parecida com os códigos anteriores mas com algumas coisas diferentes a principal dificuldade que tive foi na parte de procurar títulos com pelo menos alguma similaridade quebrei bastante a cabeça com isso até descobrir a biblioteca difflib que mede o quão parecido um texto é com outro isso foi bem interessante porque não precisava digitar o título exato só uma parte já achava os livros parecidos e buff resolvi meu problema

Outra parte que eu gostei bastante de fazer foi o negócio das estrelas, criar um dicionário pra traduzir "One", "Two", "Three" em números foi bem legal e deixou o código mais organizado. No geral foi bem amigável de fazer não muito complexo e deu pra aplicar tudo que aprendi nos exemplos anteriores mas do meu jeito

Basicamente o código pede pro usuário digitar uma palavra ou parte do título que quer buscar ele faz a requisição pro site books.toscrape.com usando requests com um header de Usuario pra simular um navegador e usa o BeautifulSoup pra analisar o HTML.

Depois acha todos os blocos de livros através da tag article com a classe 'product_pod'. Pra cada livro encontrado ele pega o título, calcula a similaridade usando o difflib, e se a palavra buscada tiver no título ou a similaridade for maior que 0.4, ele extrai as informações

As informações extraídas são o título que tá dentro do atributo "title" da tag a, o preço que tá na classe 'price_color', a avaliação em estrelas que ele converte de texto pra número usando aquele dicionário, e o status de estoque verificando se tem "In stock" no texto. Também pega o link parcial do livro e monta a URL completa concatenando com o endereço base do site

No final salva tudo em arquivos .txt numerados dentro de uma pasta 'livros', e tem um loop que roda a cada 10 minutos pra buscar novamente igual o código das vagas de emprego

Conclusões

Num geral foi bem divertido mas o importante foi entender como inspecionar elementos de um site e usar isso pra extrair dados de forma estruturada BeautifulSoup com requests é uma combinação bem poderosa pra isso e abre várias possibilidades pra projetos futuros tipo coletar dados pra análise, monitorar preços, buscar informações específicas e assim vai. No final das contas Web Scraping não é tão complicado quanto parecia no começo é mais questão de entender a estrutura HTML do site e saber usar as ferramentas certas. E de quebra ainda conheci o PyCharm que com certeza vou usar mais vezes

Referencias

Site usado para se aprofundar: https://www.digitalocean.com/community/tutorials/como-fazer-scraping-em-paginas-web-com-beautiful-soup-and-python-3-pt

Site usado para se aprofundar: https://www.twilio.com/pt-br/blog/raspagem-dados-web-python-beautiful-soap

Card 8 – video: https://www.youtube.com/watch?v=XVv6mJpFOb0