
FIAP

PYTHON – AULA 2

WEB SCRAPING

AGENDA

Aula 1 Introdução – conceitos, arquitetura cliente – servidor, Introdução a HTML, CSS e JavaScript, URLLIB e Requests.

Aula 2 Introdução ao Beautiful Soup, Expressões Lambdas, Expressões regulares (REGEX)

Aula 3 Beautiful Soup, List Comprehension, Revisitando NUMPY e PANDAS, Introdução ao Selenium

Aula 4 Selenium 4, Scraping de Imagens, Trabalhando com inputs em pesquisas, Introdução ao Docker.

Aula 5 Selenium 4, Banco de dados SQLite, projeto final, considerações finais.

WEB SCRAPING

INTRODUÇÃO AO BEAUTIFUL SOUP,
EXPRESSÕES LAMBDAS,
EXPRESSÕES REGULARES (REGEX)

Introdução - conceitos

- Em tradução literal Web Scraping significa “raspagem da web”; sendo um outro termo também muito empregado Web Crawling, que significa “rastreamento da web”.

- É uma prática de extração automatizada de dados da web.

- Não deve considerar um programa interagindo com uma Application Programming Interface (API).

- Engloba uma variedade de técnicas de programação e tecnologias, tais como análise de dados, parsing e segurança da informação.

Introdução – benefícios

- Existe uma enorme possibilidade de explorar a internet que não seja pelo navegador.

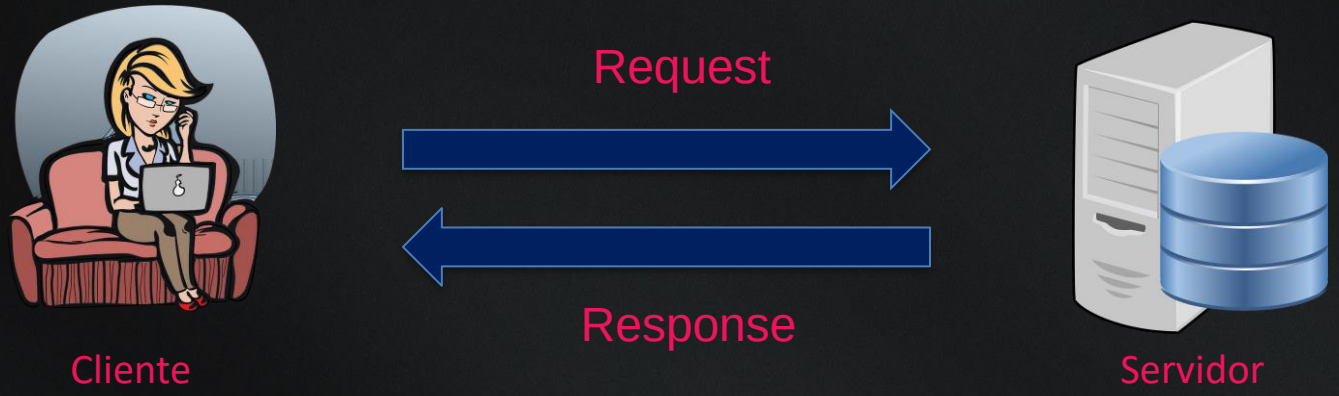
- Apesar dos navegadores serem muito bons para lidar com recursos visuais, mais intuitivos as pessoas, os web scrapers são ótimos para extrair um grande volume de dados.

- Além disso, os web scrapers podem acessar lugares que as ferramentas tradicionais, tais como buscadores não estão aptos, pois são limitados.

- Mesmo que hajam API’s para extração, os limites para volume, taxa de requisição, tipo e formato dos dados podem ser insuficientes em relação ao seus objetivos.

Arquitetura Cliente - Servidor

- O protocolo HTTP funciona por meio de requisições e respostas.
- Ao buscar por uma página web, o cliente informa o endereço no navegador e o servidor devolve uma resposta.
- Esta arquitetura é definhada por: Cliente – Servidor



Status Code

Toda requisição feita por um cliente a um servidor resulta em uma resposta representado por um código HTTP (resultado da requisição), tenha ela sido processada com sucesso ou não.

Classe	Semântica
2xx	Indica que a requisição foi processada com sucesso.
3xx	Indica ao cliente uma ação a ser tomada para que a requisição possa ser concluída.
4xx	Indica erro(s) na requisição causado(s) pelo cliente.
5xx	Indica que a requisição não foi concluída devido a erro(s) ocorrido(s) no servidor.

Verbos HTTP

Os verbos HTTP indicam qual ação está sendo requisitada pelo consumidor do serviço; os principais verbos são:

Verbos HTTP

Verbo HTTP	Ação
GET	Ler
POST	Criar
HEAD	Ler metadados
PUT	Atualizar
PATCH	Modificar parcialmente
DELETE	Excluir

User-agent

- O *user-agent* é um pedaço de software que é usado para acessar um recurso web.
- O *user-agent* pode ser um browser, terminal, web crawler, bot, script, etc.
- É o cliente por trás do cliente em uma requisição.
- Responsável por interpretar a resposta e renderizar para o usuário

Cookies

São fragmentos de dados que um servidor envia para o cliente. O cliente pode armazenar estes dados e enviá-los de volta na próxima requisição.

Normalmente, são usados para três propósitos:

- Tipificação: preferências de usuário, temas etc.
- Sessão do usuário: logins, carrinhos de compra etc.
- Rastreio: registro e análise do comportamento do usuário.

HTML, CSS e JavaScript

Quando o cliente envia requisições para o servidor, este devolve códigos que utilizam as tecnologias HTML, CSS e Javascript.

- HTML - é uma linguagem de marcação (*markup language*) utilizada para estruturar o conteúdo da página web, tais como definir seções, divisões, formulários, cabeçalhos, rodapés, links, conteúdo semântico etc.

- CSS – linguagem de estilo; define como os elementos HTML serão exibidos, tais como posições, cores, efeitos visuais etc.

- Javascript – linguagem de programação que complementa o trio, provendo funcionalidade as páginas web, tais como validações, buscas, mapas, infográficos, formulários, animações etc

Requests: HTTP for Humans™

- Requests é uma biblioteca de alto nível que proporciona um modo mais conveniente de utilizar HTTP em Python.

- Não há necessidade de adicionar strings de consulta manualmente as URLs ou codificar dados POST.

- O Pool de conexão HTTP é 100% automático.

Vamos acessar a página da biblioteca em: Pythonhttps://requests.readthedocs.io/en/latest/

BEAUTIFUL SOUP

- É uma biblioteca Python para extrair dados de arquivos HTML e XML.
- Vamos instalar a biblioteca beautifulsoup4 pelo gerenciador de pacotes
- Após a instalação, vamos criar um novo arquivo scraping_beautifulsoup.py
- Devemos importar a biblioteca BeautifulSoup do módulo bs4.
- Enviamos o conteúdo da requisição e o parâmetro html.parser

```
import requests
from bs4 import BeautifulSoup

response = requests.get("https://books.toscrape.com/")

soup = BeautifulSoup(response.content, "html.parser")
print(soup.title)
print(soup.prettify())
print(soup.text)
```

https://beautiful-soup-4.readthedocs.io/en/latest

BEAUTIFUL SOUP - Parse tree

- Páginas web são estruturadas de forma hierárquica. Desse modo, temos uma árvore com seus nós e ramificações.

- Por meio de tags podemos organizar sua estrutura, e neste caso denominamos por pais, filhos e descendentes (parents, children e descendants)

```
<ul class="breadcrumb">
  <li>
    <a href="index.html">
        Home
    </a>
  </li>
  <li class="active">
        All products
  </li>
</ul>
```

BEAUTIFUL SOUP - Parse tree

- Utilize a biblioteca BeautifulSoup para iterar com as tags. Experimente com div, conforme mostrado abaixo.

```
primeira_div = soup.div
print(primeira_div.div.div.prettify())
```

- Faça o teste com outras *tags* que você conheceu.

- Para acessar o CSS, utilize o parâmetro attrs.

```
print(primeira_div.div.attrs)
```

BEAUTIFUL SOUP - Parse tree

- Agora que podemos iterar pelas tags das páginas, vamos converter em uma lista para o Python

```
print(list(soup.ul))
```

- Perceba que os espaços vazios da página foram adicionados a nossa lista

```
['\n', <div class="row">
        <div class="col-sm-8 h1"><a href="index.html">Books to Scrape</a>
            <small>We love being scraped!</small>
        </div>
</div>, '\n']
```

BEAUTIFUL SOUP - Parse tree

- Para resolver, vamos aplicar um filtro e a biblioteca NavigableString.

```
print(list(filter(lambda x: type(x) != NavigableString, soup.ul)))
```

- A NavigableString retorna strings que estão dentro das **tags**

- Podemos aplicar este filtro para outras ***tags***. Assim, crie uma função que recebe qualquer outro elemento para ser filtrado.

BEAUTIFUL SOUP - Parse tree

```
def filtra_espaços(elemento):
   return list(filter(lambda x: type(x) != NavigableString,
elemento))

lista_filtrada = filtra_espaços(soup.ul)
print(lista_filtrada)
```

BEAUTIFUL SOUP - Parse tree

- Algumas vezes não queremos iterar de forma vertical (hierárquica) mas de forma horizontal (Siblings). Vamos analisar um item da lista

```
print(soup.ul.li)
```

```
<li>
    <a href="index.html">Home</a>
</li>
```

- Podemos utilizar **next_sibiling** para navegar de forma horizontal

```
print(soup.ul.li.next_sibling.next_sibling)
```

```
<li class="active">All products</li>
```

BEAUTIFUL SOUP - Parse tree

- Para retornar, vamos utilizar previous_sibiling.

```
print(soup.ul.li.next_sibling.next_sibling.previous_sibling.previous_sibling)
```

```
<li>
    <a href="index.html">Home</a>
</li>
```

BEAUTIFUL SOUP - Parse tree

- Para obter textos das **tags**, podemos simplesmente utilizar o atributo **text.**

```
print(soup.ul.text)
```

- Ou a função **get_text**, mais versátil e que possibilita a passagem de parâmetros para separar o texto e eliminar espaços vazios.

```
print(soup.ul.get_text(separator=", ", strip=True))
```

BEAUTIFUL SOUP - Parse tree

- Se a intenção for extrair todo o texto da página e converter em uma lista de strings para o python, podemos utilizar stripped_strings

```
strings = list(soup.stripped_strings)
print("Numero de strings: ", len(strings))
print(strings)
```

BEAUTIFUL SOUP - Parse tree

- Quando fazemos scraping da web, normalmente queremos algo específico, desse modo, precisamos refinar nossas buscas, por meio de pesquisas.

- Vamos utilizar a função find_all para filtrarmos os preços dos livros.

```
print(soup.find_all("p", attrs={"class":"price_color"}))
```

- Podemos encontrar os parâmetros, acessando o site no modo de inspeção.

[£51.77 , £53.74 , £50.10 , £47.82 , £54.23 , £22.65 , ...]

BEAUTIFUL SOUP - Parse tree

- O resultado é uma lista com as tags contendo os preços. Vamos refinar iterando por meio de um FOR e utilizando a conhecida função get_text.

```
precos = soup.find_all("p", attrs={"class":"price_color"})
for preco in precos:
    print(preco.get_text())
```

- Perceba a diferença.

- Nota: dentro de uma lista podemos utilizar a função find para iterar com elementos aninhados.

BEAUTIFUL SOUP – Exercícios

1.

- Acesse a página **Books to Scrape** e inspecione os elementos
- Faça uma busca para retornar os dados de todos os livros da página (título, preço e classificação).
- Elabore uma função que retorna os dados de um livro específico.
- Utilize a função **find** para iterar na lista de livros.
- Ex:

```
titulo = livros[2].find("h3").find("a")["title"]
```

BEAUTIFUL SOUP – Exercícios

1.

```
def extracao_livros(livros):
    avaliacao = livros.find("p", attrs={"class":"star-rating"})["class"][-1]
    preco = livros.find("p", attrs={"class": "price_color"}).get_text()
    titulo = livros.find("h3").find("a")["title"]
    return avaliacao, preco, titulo
```

BEAUTIFUL SOUP – Exercícios

2.

- Execute a função em um laço para extrair e adicionar todos os elementos em uma lista Python.
- Faça a exibição da lista completa.
- Faça a exibição da lista iterando por cada elemento.
- Faça uma busca na lista exibindo somente os livros com 4 e 5 estrelas.

BEAUTIFUL SOUP – Exercícios

2.

```
lista_livros = []
for elemento in livros:
    for livro in extracao_livros(elemento):
        lista_livros.append(livro)
print(lista_livros)
```

BEAUTIFUL SOUP

- Conseguimos extrair os dados do site e transforma-los em uma lista de strings no Python.
- Entretanto, pode acontecer de precisarmos limpar alguns dados e converte-los em valores numéricos.
- Veja o preço dos livros acompanhados da representação monetária (libras)

'£37.59'

BEAUTIFUL SOUP

- Uma solução interessante é utilizar um padrão denominado Regular Expression (Regex).
- As expressões regulares são padrões para selecionar caracteres em uma string.
- Desse modo, podemos emprega-las para remover caracteres inválidos.

Antes de aplicar Regex no python vamos praticar um pouco.

Acesse: https://regex101.com/

BEAUTIFUL SOUP

Algumas classes

- [0-9] todos os números devem ser destacados
- [a-z0-9] busca letras e números
- [A-z] todas as letras do alfabeto;
- [a-z] busca todas as letras minúsculas;
- [A-Z] busca todas as letras maiúsculas;
- [^as] ignora todas as combinações que tenham “as”
- W reduz a busca em caracteres que não sejam alfanuméricos, como espaços e símbolos.

BEAUTIFUL SOUP

- **Vamos utilizar a biblioteca Regex no Python.**

```
import re
```

- **Utilizamos a expressão regular para filtrar apenas números e pontos.**

```
def limpa_preco(preco):
    return float(re.sub("[^0-9.]", "", preco))

livro = extracao_livros(livros[2])
print(limpa_preco(livro[1]))
```

BEAUTIFUL SOUP – Exercícios

3.

- Utilize Regex para limpar e converter os preços em valores numéricos.
- Ajuste a função que retorna os dados dos livros para devolver os preços sem a representação monetária.

BEAUTIFUL SOUP – Exercícios

3.

```
def limpa_preco(preco):
    return float(re.sub("[^0-9.]","", preco))

def extracao_livros(livro):
    avaliacao = livro.find("p", attrs={"class":"star-rating"})["class"][-1]
    preco = livro.find("p", attrs={"class": "price_color"}).get_text()
    titulo = livro.find("h3").find("a")["title"]

    return titulo, limpa_preco(preco), avaliacao
```

BEAUTIFUL SOUP – Exercícios

4.

- Crie uma função que recebe uma avaliação e converte em valores numéricos. Utilize dicionários.

- Ajuste a função de extração de livros para retornar a função que converte a avaliação.

BEAUTIFUL SOUP – Exercícios

4.

```
def converte_avaliacao(avaliacao):
    classificacao={
        "One": 1,
        "Two": 2,
        "Three": 3,
        "Four": 4,
        "Five": 5
    }
    return classificacao[avaliacao]
```

```
def extracao_livros(livro):
    avaliacao = livro.find("p", attrs={"class":"star-rating"})["class"][-1]
    preco = livro.find("p", attrs={"class": "price_color"}).get_text()
    titulo = livro.find("h3").find("a")["title"]

    return titulo, limpa_preco(preco), converte_avaliacao(avaliacao)
```

BEAUTIFUL SOUP – Exercícios

5.

- Modifique a função de extração de livros para retornar um dicionário.

- **Desafio:** faça um laço para iterar com esta função e agregar os dicionários a uma lista (lista de dicionários).

BEAUTIFUL SOUP – Exercícios

5.

```
def extracao_livros_dict(livro):
    avaliacao = livro.find("p", attrs={"class":"star-rating"})["class"][-1]
    preco = livro.find("p", attrs={"class": "price_color"}).get_text()
    titulo = livro.find("h3").find("a")["title"]

    return {
        "titulo": titulo,
        "preço": limpa_preco(preco),
        "avaliação": converte_avaliacao(avaliacao)
    }
print(extracao_livros_dict(livros[4]))
```

BEAUTIFUL SOUP – Exercícios

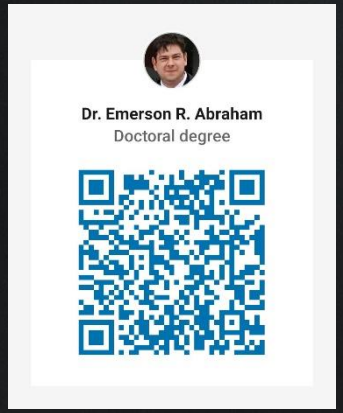
5. Desafio

```
lista_livros = [extracao_livros_dict(livro) for livro in livros]
print(lista_livros)
```

BIBLIOGRAFIA BÁSICA

- BEAZLEY, David. Python Essential Reference,. 2009.
- BEK, ANDY. The Ultimate Web Scraping With Python Bootcamp 2023.
- BHARGAVA, ADITYA Y. Entendendo Algoritmos. Um guia ilustrado para programadores e outros curiosos. São Paulo: Ed. Novatec, 2017
- DOWNEY, ALLEN B. Pense em Python. Pense como um cientista da computação. São Paulo: Ed. Novatec, 2016
- GRANATYR, Jones; PACHOLOK, Edson. IA Expert Academy. Disponível em: https://iaexpert.academy/
- KOPEC, DAVID. Problemas clássicos de ciência da computação com Python. São Paulo: Ed. Novatec, 2019
- MCKINNEY, WILLIAM WESLEY. Python para análise de dados. Tratamento de dados com Pandas, Numpy e Ipython. São Paulo: Ed. Novatec, 2018
- MITCHELL, RYAN. Web Scraping com Python. Coletando mais dados na web moderna. São Paulo: Ed. Novatec, 2019

OBRIGADO



FIAP


FIAP