

Web scraping

Exemplo 1. De um html fictício

```
import json
from bs4 import BeautifulSoup

# --- SIMULAÇÃO DA PÁGINA WEB ---
# Em vez de baixar da internet, criamos um HTML falso para testar
html_simulado = """
<html>
  <head><title>Página de Teste</title></head>
  <body>
    <h1 class="headline">Como aprender Web Scraping em 2026</h1>
    <p class="lead">O Web Scraping é uma técnica poderosa para coletar
dados da internet de forma automatizada.</p>
    <p class="lead">Com bibliotecas como BeautifulSoup, o processo se
torna simples e estruturado.</p>
    <p class="normal-text">Este parágrafo não tem a classe 'lead',
então será ignorado.</p>
  </body>
</html>
"""

# Simulamos que a requisição deu certo e encontramos o HTML
html_bruto = html_simulado
url_ficticia = 'https://exemplo.edu'

# --- 2. Inicializando o Parser DOM ---
soup = BeautifulSoup(html_bruto, 'html.parser')

# --- 3. Extraíndo através de Seletores ---
# Procura o h1 com a classe 'headline'
titulo = soup.find('h1', class_='headline').get_text(strip=True) if
soup.find('h1', class_='headline') else "Título não encontrado"

# Procura todos os parágrafos com a classe 'lead'
paragrafos = [p.get_text(strip=True) for p in soup.find_all('p',
class_='lead')]

# --- 4. Estruturando o resultado com metadados ---
```

```

dados_extraídos = {
    'url_origem': url_ficticia,
    'titulo': titulo,
    'texto': " ".join(paragrafos)
}

# Exibe o resultado final estruturado
print("Dados extraídos com sucesso:")
print(json.dumps(dados_extraídos, indent=4, ensure_ascii=False))

```

Exemplo 2. Raspando do Wikipedia

```

import requests
import csv
from bs4 import BeautifulSoup

def raspar_dados_wikipedia(nome_arquivo="populacao_estados_brasil.csv"):
    # 1. URL da página oficial da lista de estados por população
    url_wiki =
    "https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_
    _popula%C3%A7%C3%A3o"

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
        AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
    }

    print("Conectando de forma segura aos servidores da Wikipédia...")
    resposta = requests.get(url_wiki, headers=headers, timeout=15)

    if resposta.status_code != 200:
        print(f"❌ Falha ao acessar a página. Código de erro:
        {resposta.status_code}")
        return

    # 2. Faz o parse do HTML puro da página
    soup = BeautifulSoup(resposta.text, 'html.parser')

```

```
# 3. Localiza a tabela de dados (na Wikipédia, tabelas de dados usam a
classe 'wikitable')
tabela = soup.find('table', class_='wikitable')

if not tabela:
    print("✗ A tabela com a classe 'wikitable' não foi localizada na
página.")
    return

print("Tabela encontrada! Extraíndo dados geográficos...")

dados_estados = []
# Pega todas as linhas da tabela (tags <tr>)
linhas = tabela.find_all('tr')

for linha in linhas:
    # Pega todas as colunas de dados de cada linha (tags <td>)
    colunas = linha.find_all('td')

    # Se a linha tiver colunas preenchidas, extraímos as informações
    if colunas:
        # Captura a posição no ranking
        posicao = colunas[0].get_text().strip()

        # Captura o nome do Estado (geralmente dentro de um link <a>)
        estado = colunas[1].get_text().strip()

        # Captura a População (remove espaços e caracteres especiais
do texto)
        populacao = colunas[2].get_text().strip()

        # Captura a porcentagem da população total do país
        porcentagem = colunas[3].get_text().strip()

        # Adiciona na nossa lista organizada
        dados_estados.append([posicao, estado, populacao,
porcentagem])

# 4. Salva os dados capturados diretamente em um arquivo CSV
estruturado
```

```

if dados_estados:
    with open(nome_arquivo, mode="w", newline="", encoding="utf-8") as
file:
        writer = csv.writer(file)
        # Cabeçalho da planilha
        writer.writerow(["Posição", "Estado (Unidade Federativa)",
"População", "% da Pop. Total"])
        writer.writerows(dados_estados)

    print(f"\n 🎉 SUCESSO! {len(dados_estados)} estados mapeados e
salvos em '{nome_arquivo}'.")
else:
    print("⚠️ Nenhuma linha de dados pôde ser tratada de dentro da
tabela.")

# --- EXECUÇÃO DO PROJETO ---
raspar_dados_wikipedia()

```

Código acima explicado

1. Importação de Bibliotecas

- `import requests`
Importa a biblioteca HTTP Requests. Ela é responsável por fazer o Python conversar com a internet, permitindo enviar requisições para servidores e baixar o código de páginas web.
- `import csv`
Importa o módulo nativo do Python para manipulação de arquivos CSV. Ele fornece as ferramentas necessárias para criar, escrever e organizar dados em formatos de planilhas.
- `from bs4 import BeautifulSoup`
Importa a classe BeautifulSoup do pacote bs4. Essa biblioteca funciona como um "tradutor" de código de páginas. Ela lê o HTML bruto (que vem como um texto gigante e confuso) e o transforma em um objeto organizado que o Python consegue pesquisar através de tags.

2. Definição da Função e Configuração da Requisição

- `def`
`raspar_dados_wikipedia(nome_arquivo="populacao_estados_brasil.csv"):`
Cria a função principal do programa. Ela aceita um parâmetro opcional chamado `nome_arquivo` que já define por padrão qual será o nome da planilha gerada no final.
 - `url_wiki = "https://pt.wikipedia.org/..."`
Armazena o endereço web em uma variável. É a URL da página exata da Wikipédia que contém a tabela de populações das unidades federativas do Brasil.
 - `headers = { "User-Agent": "..." }`
Cria um dicionário com os cabeçalhos da requisição. O parâmetro *User-Agent* simula um navegador real (neste caso, o Google Chrome rodando no Windows 10). Isso avisa o servidor que a conexão parte de um navegador comum, e não de um robô genérico, evitando bloqueios automáticos.
 - `print("Conectando de forma segura...")`
Imprime uma mensagem informativa no terminal para que o usuário saiba que o programa iniciou e está tentando se conectar à internet.
 - `resposta = requests.get(url_wiki, headers=headers, timeout=15)`
Dispara a requisição HTTP de fato. O método `.get()` vai até o site informado, envia a nossa identidade disfarçada (`headers`) e define um limite de tempo máximo de 15 segundos (`timeout`). Se o site não responder nesse período, o programa encerra para não travar o computador.
 - `if resposta.status_code != 200:`
Verifica se a conexão deu certo. O código `200` significa "Sucesso/OK" no protocolo da internet. Qualquer número diferente disso (como o famoso erro 404 de página não encontrada) indica uma falha.
 - `print(f"❌ Falha ao acessar...")`
Caso a conexão falhe, imprime o código de erro na tela para diagnóstico do desenvolvedor.
 - `return`
Interrompe imediatamente a execução da função se houver um erro de conexão, impedindo que o script continue e quebre nas linhas seguintes.
-



3. Processamento do Código da Página (Parsing)

- `soup = BeautifulSoup(resposta.text, 'html.parser')`
Ativa o motor de tradução do BeautifulSoup. O `resposta.text` contém o HTML bruto baixado do site. Passamos o parâmetro `'html.parser'` para indicar ao interpretador nativo do Python que ele deve organizar esse código com base nas regras de tags da web.
 - `tabela = soup.find('table', class_='wikitable')`
Busca a tabela de dados na página. O método `.find()` vasculha o código do início ao fim e captura a primeira tag `<table>` que possua a classe CSS chamada `wikitable` (que é o padrão de tabelas de dados públicos da Wikipédia).
 - `if not tabela:`
Cria uma trava de segurança. Se a Wikipédia mudar o layout e a classe deixar de existir, a variável `tabela` ficará vazia (`None`).
 - `print("❌ A tabela com a classe...")`
Informa o erro ao usuário caso a tabela não seja localizada.
 - `return`
Interrompe o código caso a tabela não exista, evitando erros de leitura nas etapas seguintes.
 - `print("Tabela encontrada!...")`
Exibe uma mensagem de sucesso no terminal confirmando que o container principal de dados foi capturado e o escaneamento vai começar.
-



4. Extração de Linhas e Colunas (Looping)

- `dados_estados = []`
Cria uma lista vazia na memória do Python. Ela servirá como um "balde" para acumular os dados limpos de cada estado antes de salvarmos tudo na planilha.
- `linhas = tabela.find_all('tr')`
Mapeia as linhas da tabela. O método `.find_all('tr')` captura todas as tags `<tr>` (que significam *Table Row* ou Linha da Tabela) contidas dentro do bloco da tabela e cria uma lista com elas.

- `for linha in linhas:`
Inicia um laço de repetição (loop). O programa passará de linha em linha, executando os comandos internos para cada uma das linhas capturadas na tabela da Wikipédia.
 - `colunas = linha.find_all('td')`
Mapeia as colunas de dados da linha atual. O método `.find_all('td')` busca as células de dados (*Table Data*) existentes especificamente dentro daquela linha.
 - `if colunas:`
Filtra linhas de cabeçalho. Linhas que contêm títulos (tags `<th>`) não possuem tags `<td>`. Esse comando garante que o Python só processe as linhas que realmente tenham colunas com dados numéricos.
 - `posicao = colunas[0].get_text().strip()`
Extrai o ranking do estado. Acessa a primeira coluna (índice 0), usa `.get_text()` para extrair apenas a informação em formato de texto e aplica o `.strip()` para apagar espaços em branco inúteis ou quebras de linha (`\n`) nas pontas.
 - `estado = colunas[1].get_text().strip()`
Extrai o nome do Estado/Unidade Federativa acessando a segunda coluna disponível da linha (índice 1).
 - `populacao = colunas[2].get_text().strip()`
Extrai a quantidade de habitantes acessando a terceira coluna disponível da linha (índice 2).
 - `porcentagem = colunas[3].get_text().strip()`
Extrai a representação percentual daquele estado no território nacional acessando a quarta coluna disponível da linha (índice 3).
 - `dados_estados.append([posicao, estado, populacao, porcentagem])`
Guarda as informações extraídas. O comando `.append()` agrupa as quatro variáveis limpas em uma mini lista e a joga para dentro do nosso "balde" principal (`dados_estados`).
-

5. Gravação no Arquivo Planilha (CSV)

- `if dados_estados:`

Verifica se o balde de dados não está vazio antes de tentar abrir ou criar um arquivo físico no computador.

- `with open(nome_arquivo, mode="w", newline="", encoding="utf-8") as file:`

Abre ou cria o arquivo CSV de forma segura.

- `mode="w"` ativa o modo de escrita (*write*).
- `newline=""` impede o Python de pular linhas em branco de forma errada no Windows.
- `encoding="utf-8"` garante que letras com acentos ou caracteres especiais da nossa língua (como "São Paulo", "Pará", "ã") sejam gravados de forma legível.
- O operador `with` garante que o arquivo seja fechado automaticamente pelo sistema operacional assim que a gravação terminar, liberando memória RAM.

- `writer = csv.writer(file)`

Cria o objeto de escrita do CSV. Ele pega o arquivo aberto (`file`) e o prepara para receber comandos de estruturação em formato de planilha separada por vírgulas ou pontos e vírgulas.

- `writer.writerow(["Posição", "Estado...", "População", "% da Pop. Total"])`

Escreve a primeira linha da planilha (o Cabeçalho). O método `.writerow()` (no singular) aceita uma lista de termos e os grava como os títulos das colunas na primeira linha do arquivo Excel.

- `writer.writerows(dados_estados)`

Grava todas as informações coletadas de uma só vez. O método `.writerows()` (no plural) recebe o nosso balde de dados e escreve todas as linhas acumuladas dos 27 estados em lote abaixo do cabeçalho.

- `print(f"\n 🎉 SUCESSO! {len(dados_estados)}...")`

Exibe a mensagem final de confirmação no terminal, utilizando o comando `len()` para contar dinamicamente quantos estados foram salvos com sucesso na planilha.

- `else:`

Caso o balde de dados tenha terminado vazio, o programa ignora a gravação e pula para a linha de aviso abaixo.

- `print("⚠️ Nenhuma linha de dados...")`

Informa o usuário que o HTML da página foi modificado de tal forma que nenhum dado válido pôde ser extraído e limpo.

6. Disparo do Programa

- `raspar_dados_wikipedia()`

Executa a função de fato. Até aqui o código era apenas um manual de instruções guardado na memória. Esta linha ordena o Python a iniciar a leitura, passando por todas as etapas descritas acima.