Coletando dados na internet

Aula 16

Ana Júlia Lima

Desenvolvido para
Young 1 - sexta #5228

Ctrl Play
Brasil
Outubro 2025

Sumário

1 Coletando Dados na Internet — Web Scraping

1.1 Introdução

O **Web Scraping** é a técnica de coletar informações de sites de forma automática, usando código Python. Com ele, é possível capturar textos, links, tabelas e outros dados exibidos em uma página web.

1.2 Por que aprender Web Scraping?

- Automatiza a coleta de dados;
- Permite construir bases personalizadas;
- É útil para análises de preços, notícias, estatísticas;
- Ensina a compreender a estrutura das páginas HTML.

1.3 Entendendo o HTML

Antes de extrair dados, é importante compreender a estrutura do HTML, que é composta por **tags**, **atributos** e **conteúdo**.

```
<html>
  <head>
    <title>Exemplo de Página</title>
  </head>
  <body>
    <h1>Bem-vindo!</h1>
    <p class="descricao">Este é um exemplo simples de HTML.</p>
  </body>
</html>
```

1.4 Instalando bibliotecas

Para acessar páginas e analisar HTML, instale as bibliotecas:

```
pip install requests
pip install beautifulsoup4
```

1.5 Acessando páginas web com Python

```
import requests

url = "https://www.example.com"
resposta = requests.get(url)

print(resposta.status_code)  # código HTTP
print(resposta.text)         # HTML da página
```

Códigos HTTP comuns:

- 200 — Sucesso;
- 404 — Página não encontrada;
- 500 — Erro no servidor.

1.6 Usando o Beautiful Soup

```
from bs4 import BeautifulSoup

html = resposta.text
sopa = BeautifulSoup(html, "html.parser")
```

1.7 Funções principais do BeautifulSoup

`prettify()` Formata o HTML para visualização:

```
print(sopa.prettify())
```

`find()` Retorna a primeira ocorrência da tag especificada:

```
titulo = sopa.find("h1")
print(titulo.text)
```

`find_all()` Retorna uma lista com todas as ocorrências da tag:

```
paragrafos = sopa.find_all("p")
for p in paragrafos:
    print(p.text)
```

1.8 Exemplo completo de Web Scraping

```
import requests
from bs4 import BeautifulSoup

url = "https://www.example.com"
resposta = requests.get(url)

sopa = BeautifulSoup(resposta.text, "html.parser")

titulo = sopa.find("h1")
print("Título da página:", titulo.text)

paragrafos = sopa.find_all("p")
for p in paragrafos:
    print("Parágrafo:", p.text)
```

1.9 Resumo prático

- `requests`: acessa páginas web;
- `BeautifulSoup`: interpreta e navega no HTML;
- `find()`: retorna a primeira tag encontrada;
- `find_all()`: retorna todas as tags encontradas;
- `prettify()`: mostra o HTML formatado.

2 Pandas — Trabalhando com Dados em Python

2.1 Introdução

Depois de coletar informações com Web Scraping, é necessário organizar e analisar os dados. A biblioteca **Pandas** permite trabalhar com tabelas de dados (DataFrames) de forma simples e eficiente.

2.2 O que é o Pandas?

O Pandas (Python Data Analysis Library) é usado para:

- Organizar dados em formato de tabela;
- Importar e exportar arquivos (CSV, Excel, JSON etc.);
- Tratar valores nulos e duplicados;
- Filtrar, ordenar e agrupar informações;
- Fazer análises estatísticas com facilidade.

2.3 Instalação e importação

```
pip install pandas
```

```
import pandas as pd
```

2.4 Criando um DataFrame

```
import pandas as pd

dados = {
    "Nome": ["Ana", "Bruno", "Carla"],
    "Idade": [23, 30, 27],
    "Cidade": ["São Paulo", "Rio de Janeiro", "Curitiba"]
}

df = pd.DataFrame(dados)
print(df)
```

Saída:

```
    Nome  Idade        Cidade
0    Ana     23     São Paulo
1  Bruno     30  Rio de Janeiro
2  Carla     27       Curitiba
```

2.5 Relacionando Pandas e Web Scraping

Após coletar dados com BeautifulSoup, podemos armazená-los em um DataFrame para análise.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = "https://quotes.toscrape.com/"
resposta = requests.get(url)
sopa = BeautifulSoup(resposta.text, "html.parser")
```

```python
citacoes = sopa.find_all("span", class_="text")
autores = sopa.find_all("small", class_="author")

lista_citacoes = [c.text for c in citacoes]
lista_autores = [a.text for a in autores]

df = pd.DataFrame({
    "Citação": lista_citacoes,
    "Autor": lista_autores
})

print(df.head())
```

2.6 Salvando dados em arquivo

```python
df.to_csv("citacoes.csv", index=False, encoding="utf-8")
```

2.7 Lendo arquivos com Pandas

```python
dados = pd.read_csv("citacoes.csv")
print(dados.head())
```

2.8 Operações úteis com DataFrames

- `df.head()` — mostra as 5 primeiras linhas;
- `df.info()` — mostra informações gerais;
- `df.describe()` — exibe estatísticas básicas;
- `df.shape` — número de linhas e colunas;
- `df.columns` — nomes das colunas;
- `df.sort_values()` — ordena os dados;
- `df.dropna()` — remove valores nulos;
- `df.fillna()` — substitui valores nulos.

2.9 Resumo prático

- **requests**: acessa páginas web;
- **BeautifulSoup**: interpreta HTML;
- **pandas**: organiza e analisa os dados coletados.