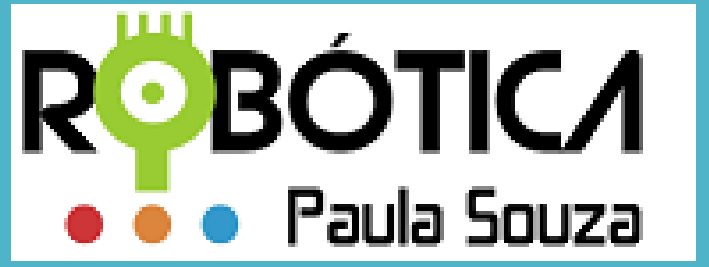
Prof. Me. Jean

Workshop de Web Scraping

https://robotica.cps.sp.gov.br/

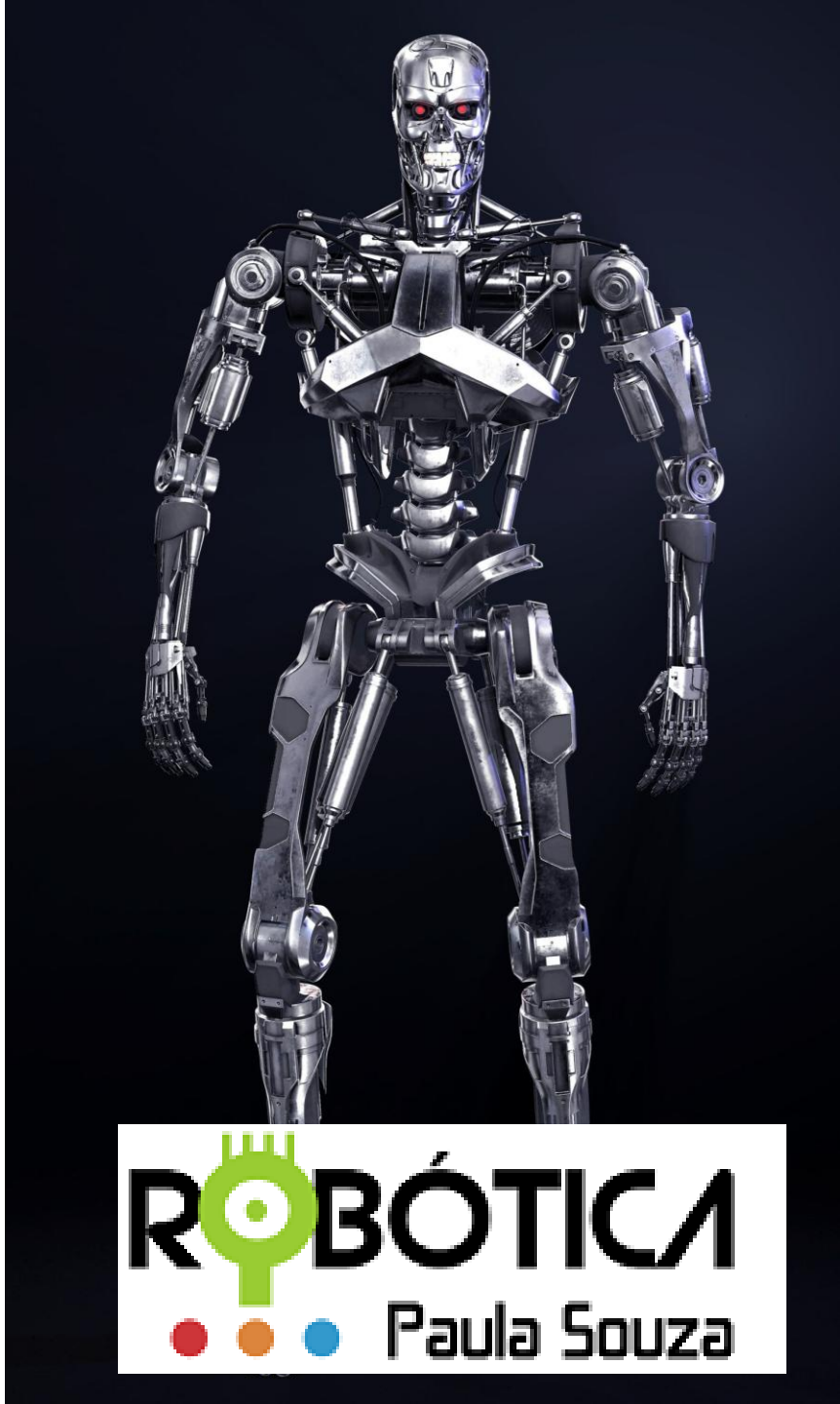


Tipos de Robôs

Quando ouvimos a palavra **"robô"**, geralmente imaginamos máquinas com:

- Braços metálicos,
- Sensores,
- Rodas,
- Corpo,
- Pernas...

como os que vemos na indústria, em filmes e laboratórios.



Robôs que não têm corpo

Mas existem também **robôs invisíveis**.

Robôs que **não têm corpo**, mas têm uma função bem definida:

executar tarefas automaticamente dentro de ambientes digitais,

como um site, uma rede social ou um sistema.

Os bots

Robôs sem corpo são chamados de **bots** .

São programas criados para navegar, interagir, buscar e coletar informações na internet de forma automatizada.

Entre esses bots, um dos mais úteis é o **WebScraper.**

ROBÓTICA
Paula Souza

Curiosidade: 🤔

O Google começou com Web Scraping?

A resposta é Sim.

O **Google** nasceu **como um robô que visitava** páginas da web automaticamente, copiava seu conteúdo e armazenava em servidores para organizar e indexar as informações.

Exatamente o que chamamos hoje de Web Crawling, uma forma de Web Scraping.

Esse mecanismo permitiu ao Google montar seu buscador, classificando os sites e exibindo os resultados de forma útil para o usuário.

O que fazemos com Web Scraping hoje tem raízes na mesma lógica que deu origem ao maior buscador do mundo.

Exemplos de sites que usam ou usaram Web Scraping

- **Google** – para indexar e organizar páginas da web (Googlebot)
- **Buscapé** – coleta e comparação de preços em lojas online
- **Trivago / Kayak / Decolar** – comparação de preços de hotéis e passagens
- **WebMotors / OLX / Zap Imóveis** – monitoramento de anúncios e mercado
- **LinkedIn** – alvo comum de scraping por empresas (possui restrições legais)

Mais:

- **Portais de Notícias** – coleta de manchetes e dados para agregadores
- **Sites Governamentais / de Licitações** – extração de dados públicos e relatórios
- **Sites Acadêmicos** – coleta de artigos, teses, citações e indicadores

O que é Web Scraping ou robô: WebScraper?

WebScraper: um robô digital que visita páginas da web, **lê e coleta informações**, como se fosse um ser humano navegando e anotando os dados, só que de forma muito mais rápida, precisa e automática.

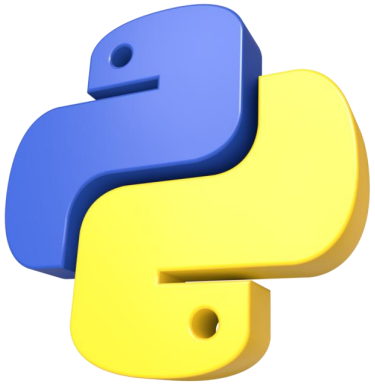
Web scraping ou Raspagem Web é uma técnica que permite a coleta de informações de websites de forma automática.

Imagine que você deseja reunir dados de várias páginas, como preços de produtos, horários de eventos ou informações sobre notícias ou artigos científicos.

Fazer isso manualmente pode ser muito trabalhoso e demorado.

Web Scraping - Python

Com o Web Scraping, usamos programas de computador (neste caso o Python) para acessar essas páginas e extrair as informações desejadas.



É como ter um assistente virtual que lê as páginas da web para você e coleta os dados que precisa.

Essa técnica é muito útil para pesquisadores, desenvolvedores e empresas que precisam de grandes quantidades de dados de maneira rápida e eficiente.

ROBÓTICA
Paula Souza

10 tipos de usos para Web Scraping

1. **Comparação de Preços**: Coleta de preços em sites de e-commerce.
2. **Agregação de Notícias**: Reunião de artigos de várias fontes em um único lugar.
3. **Análise de Sentimento**: Coleta de avaliações de redes sociais.
4. **Pesquisa Acadêmica**: Obtenção de dados para estudos.
5. **Monitoramento de Vagas**: Agregação de oportunidades de emprego de diferentes sites.
6. **Avaliação Imobiliária**: Rastreio de preços de imóveis para análises.
7. **Dados Meteorológicos**: Coleta de previsões climáticas.
8. **Identificação de Tendências**: Descoberta de produtos/assuntos populares.
9. **Análise de Concorrentes**: Monitoramento nos sites de concorrentes.
10. **Pesquisa de Mercado**: Coleta de dados para estratégias de negócios e economia.

Como Funciona o Web Scraping?

1. Acessa uma página da WEB.
2. Obtém e Interpreta a resposta (HTML).
3. Extrai as informações desejadas.
4. Salva essas informações.

Ferramentas de Web Scraping

As ferramentas básicas que vamos utilizar:

- Visual Studio Code
- Python 3.11 instalado

Guia de Instalação do VS Code e Pyhton 3

Passo 1: Instalar o Visual Studio Code (VS Code)

1.Acesse o site do VS Code: Download VS Code.

2.Baixe e execute o instalador.

3.Durante a instalação, marque a opção 📌 **"Add to PATH"** 📌 para poder usar o terminal.

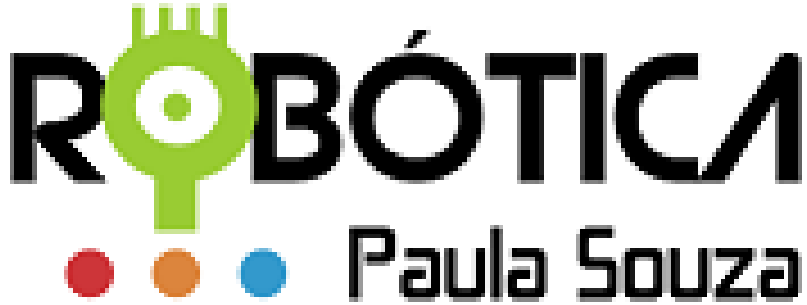
4.Complete a instalação seguindo as instruções do instalador.

ROBÓTICA
Paula Souza

Guia de Instalação do VS Code e Pyhton 3

Passo 2: Instalar Python 3.11

1. Acesse o site oficial do Python: Download Python 3.11.
2. Baixe o instalador do Python 3.11 para Windows.
3. Execute o instalador. Importante marcar a opção 📌 **"Add Python 3.11 to PATH"** 📌 para facilitar o uso do Python no terminal.
4. Escolha **"Customize installation"** e depois **"Install Now"** para concluir a instalação.



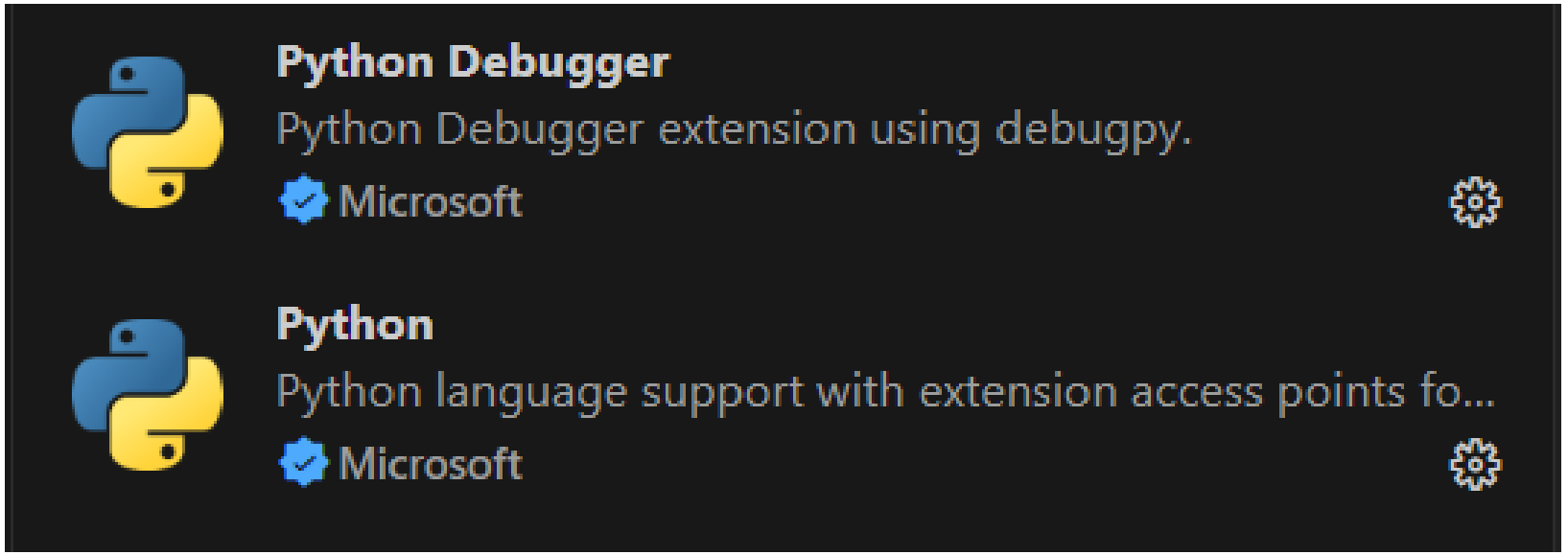
Guia de Instalação

Passo 3: Instalar a Extensão Python no VS Code

1.Abra o VS Code.

2.Vá ao **Marketplace de Extensões** (ícone de quadrado no menu lateral) e procure por "Python".

3.Instale as extensões **Python da Microsoft** para suporte a recursos Python (imagem).



ROBÓTICA Paula Souza

Guia de Instalação

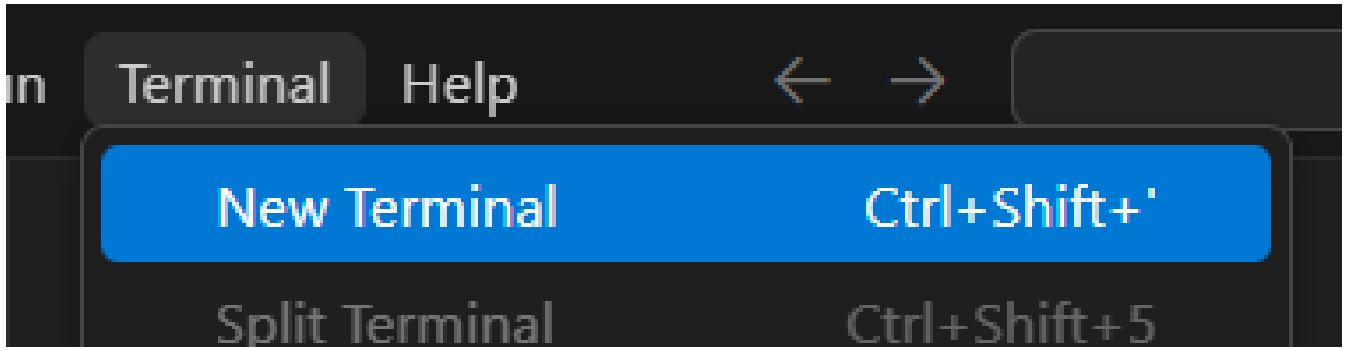
Passo 4: Verificar a Instalação do Python e pip

1.Abra o terminal (Prompt de Comando, PowerShell, ou Terminal no VS Code).

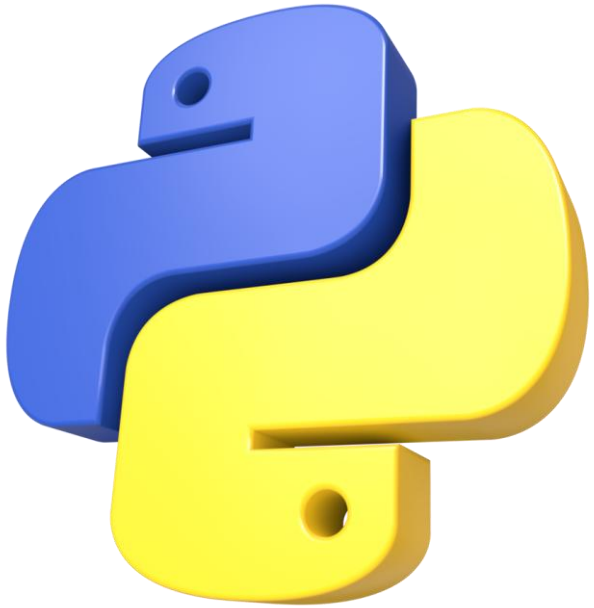
2.Digite o comando para verificar a instalação do Python:

3.**python –version** ou **python3 –version**

4.Se o comando funcionar, está pronto para uso.



Para acessar o Terminal no VS Code.



Iniciando um novo projeto

Criar uma nova Pasta para o projeto

Abra a Pasta no Visual Studio Code

Criar um arquivo chamado: main.py

A requests permite que seu código Python se conecte a sites, envie formulários, baixe conteúdos, acesse APIs ou colete páginas HTML – tudo via internet.

Instalação das Bibliotecas

Abra o Terminal no Visual Studio Code e aplique o comando abaixo:

```
pip install requests
```

A biblioteca **requests** permite que seu código Python se conecte a sites, envie formulários, baixe conteúdos, ou colete páginas HTML usando a Internet.

ROBÓTICA Paula Souza

Instalação da Biblioteca beautifulsoup4

Abra o Terminal no Visual Studio Code e aplique o comando abaixo:

```
pip install bs4
```

ou

```
pip install beautifulsoup4
```

BeautifulSoup é uma biblioteca Python usada para extrair dados de arquivos HTML e XML de forma simples e intuitiva.
Ela não faz a requisição da página (isso é com requests), mas lê o conteúdo HTML recebido e permite navegar, buscar e extrair dados

ROBÓTICA Paula Souza

Exemplo de Requisição HTTP (requests)

```
import requests

url = 'https://exemplo.com'
response = requests.get(url)
print(response.text)  # imprime o HTML da página
```

O que esse código faz:

- **import requests:** importa a biblioteca responsável por fazer requisições HTTP.
- **url = 'https://exemplo.com':** define o endereço da página que será acessada.
- **requests.get(url):** envia uma requisição do tipo GET para o site e armazena a resposta.
- **response.text:** acessa o conteúdo da resposta (o código HTML da página).
- **print(...):** exibe o HTML da página no terminal.

Analisando o HTML com BeautifulSoup

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(response.text, 'html.parser')
print(soup.prettify())  # Exibe o HTML legível
```

O que esse código faz:

- **from bs4 import BeautifulSoup:** importa a biblioteca usada para analisar e manipular HTML.
- **BeautifulSoup(response.text, 'html.parser'):** cria um **"objeto sopa HTML"** a partir da página.
- **soup:** agora é um objeto que representa a estrutura da página (é a página Web).
- **soup.prettify():** formata o HTML com indentação para facilitar a leitura.
- **print(...):** exibe o HTML organizado no terminal.

Como obter os conteúdos das páginas

Antes de começar a programar, **é fundamental inspecionar a estrutura** da página no navegador (com o botão direito → "Inspecionar elemento") para identificar:

- **As tags que contêm os dados** (ex: <h1>, <p>, <a>, <img>, , etc.)
- **Os ids e classes usados**
- **Os atributos relevantes** (ex: href, src, alt, title, etc.)

Isso facilita muito a criação do código, pois você saberá exatamente onde e como buscar as informações.

Exemplo: Obter Todos os Elementos pela Tag

Neste exemplo vamos obter e listar todos os conteúdos das Tags <p> encontradas

```
paragrafos = soup.find_all('p')
for p in paragrafos:
    print(p.get_text())
```

Exemplo: Obter Elemento pelo ID

Neste exemplo vamos obter o conteúdo da Tag especifica através de seu ID

```
elemento = soup.find(id='id_da_tag')
print(elemento.get_text())
```

Exemplo: Obter Elementos por Classe CSS

Neste exemplo vamos obter o conteúdo de todas Tag que possuam uma classe CSS

```
elementos = soup.find_all(class_='classe-da-tag')
for el in elementos:
    print(el.get_text())
```

Exemplo: Obter o **href** das Tags <a>

Neste exemplo vamos obter o conteúdo do atributo **href** de todas Tag <a>

```
links = soup.find_all('a')
for link in links:
    print(link.get('href'))
```

python™

Exemplo: Obter **src** das Imagens

Neste exemplo vamos obter o conteúdo do atributo **src** de todas Tag <img>

```
imagens = soup.find_all('img')
for img in imagens:
    print(img.get('src'))
```

Outras dicas avançadas

Navegando com Paginação:

```
while next_page is not None:
    # Faz a requisição e extrai dados da próxima página
```

Salvando os Dados em CSV

```
import csv

with open('dados.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Campo1', 'Campo2'])  # Cabeçalhos
    for dado in dados:
        writer.writerow([dado['campo1'], dado['campo2']])
```

Boas Práticas no Web Scraping (Ética)

Boas práticas no Web Scraping é essencial para formar programadores **conscientes, éticos e técnicos**.

🚫 Ignorar as regras éticas podem gerar bloqueio de IP ou até problemas legais 🚫.

Como?

✅ Sempre verifique se o site permite scraping ✅.

ROBÓTICA Paula Souza

Boas Práticas – Como saber se posso?

✅ Sempre verifique se o site permite scraping ✅ .

1. Respeite os Termos de Uso dos sites (faça a leitura para saber não é bloqueado o Scraping)

2. Muitos sites possuem um arquivo chamado robots.txt (ex: https://exemplo.com/robots.txt) que define o que pode ou não ser acessado automaticamente. (faça a leitura para saber não é bloqueado o Scraping)

ROBÓTICA Paula Souza

Boas Práticas – Evitar sobrecarregar os servidores

Evite sobrecarregar os servidores.

1. Sites não foram feitos para serem acessados por robôs em alta velocidade.
2. Use delays (um tempo) entre as requisições (ex: time.sleep(5)).
3. Em projetos maiores, considere usar uma fila de execução com controle de tempo.
4. Servidores não suportam muitos acessos simultâneos em intervalos de tempo muito pequenos.
5. Seu IP pode ser bloqueado se exagerar.

Conclusão

- Agora que você tem uma base sólida sobre Web Scraping, está pronto para explorar sites e coletar dados de forma prática e responsável.
- Sempre respeite as boas práticas: consulte o robots.txt, evite sobrecarregar servidores e siga as regras do site.
- Com Python e bibliotecas como requests e BeautifulSoup, é possível transformar páginas da web em fontes ricas de dados — seja para estudos, projetos ou análises de mercado.

https://robotica.cps.sp.gov.br/

ROBÓTICA Paula Souza

Dúvidas ???

Acesse

https://robotica.cps.sp.gov.br/

ROBÓTICA
Paula Souza