Guia de Extração de Dados com Python

Índice

1. Extração Básica de Dados

Para acessar um site, extrair informações e salvar em uma planilha, você pode usar as bibliotecas `requests`, `BeautifulSoup` e `pandas`. Aqui está um exemplo básico:

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

# Passo 1: Acessar o site
url = 'https://exemplo.com'
response = requests.get(url)

# Verifica se a requisição foi bem-sucedida
if response.status_code == 200:
    # Passo 2: Extrair as informações usando BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')

    # Exemplo de extração de dados: Encontrando todas as tags <h2>
    titles = soup.find_all('h2')
    data = []

    for title in titles:
        data.append(title.get_text())

    # Passo 3: Organizar os dados em um DataFrame
    df = pd.DataFrame(data, columns=['Titles'])

    # Passo 4: Salvar os dados em uma planilha Excel
    df.to_excel('dados_extraidos.xlsx', index=False)

    print("Dados extraídos e salvos em 'dados_extraidos.xlsx'")
else:
    print(f"Erro ao acessar o site: {response.status_code}")
```

2. Autenticação

Para sites que exigem autenticação, você pode usar `requests.Session()` para gerenciar a sessão. Exemplo:

```
login_url = 'https://exemplo.com/login'
session = requests.Session()
payload = {'username': 'seu_usuario', 'password': 'sua_senha'}
session.post(login_url, data=payload)
```

3. Paginação

Se os dados estão em várias páginas, você pode iterar sobre as páginas. Exemplo:

```
for page in range(1, total_pages+1):
    url = f'https://exemplo.com/page={page}'
    response = session.get(url)
    # Processar cada página
```

4. Manipulação de JavaScript Dinâmico

Para sites que carregam dados via JavaScript, você pode usar `Selenium`. Exemplo:

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get('https://exemplo.com')
# Interagir com a página e extrair dados
```

5. Tratamento de Dados Complexos

Se os dados são complexos, você pode precisar navegar na estrutura HTML. Exemplo:

```
soup = BeautifulSoup(response.content, 'html.parser')
complex_data = soup.find('div', {'class': 'complex-structure'})
```

6. Manejo de Erros e Exceções

Para lidar com erros, implemente tratamento de exceções. Exemplo:

```
try:
    response = requests.get(url)
    response.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(f"HTTP error occurred: {err}")
```

7. Exportação Avançada

Para exportar dados em formatos avançados:

```
with pd.ExcelWriter('dados.xlsx') as writer:
    df1.to_excel(writer, sheet_name='Aba1')
    df2.to_excel(writer, sheet_name='Aba2')
```

8. Requisitos Legais e Éticos

Respeite os Termos de Serviço do site e considere pedir permissão ou usar APIs públicas.