Manipulação de Planilhas Excel com Python

Manipular planilhas do Excel usando Python é uma tarefa bastante comum e pode ser feita com bibliotecas como **openpyxl**, **pandas**, ou **xlrd** para leitura, e **openpyxl** ou **xlsxwriter** para escrita. Abaixo, vou te mostrar como fazer isso usando a biblioteca *openpyxl*, que é uma das mais populares para este propósito.

1. Instalação da Biblioteca

Primeiro, você precisa instalar a biblioteca *openpyxl*:

```
pip install openpyxl
```

2. Abrindo uma Planilha do Excel

Para abrir uma planilha, você usa o método `load_workbook`:

```
from openpyxl import load_workbook

# Carregar a planilha
workbook = load_workbook('caminho/para/sua/planilha.xlsx')

# Selecionar uma aba específica
sheet = workbook['NomeDaAba']
```

3. Lendo Dados de uma Célula

Você pode ler dados de uma célula específica acessando-a pelo endereço da célula:

```
# Ler o valor de uma célula específica
valor = sheet['A1'].value
print(valor)
```

4. Escrevendo Dados em uma Célula

Para inserir dados em uma célula, basta atribuir um valor a ela:

```
# Escrever um valor em uma célula específica
sheet['B2'].value = 'Novo Valor'
```

5. Salvando as Alterações

Após modificar os dados, você deve salvar o arquivo:

```
# Salvar o arquivo
workbook.save('caminho/para/sua/planilha_modificada.xlsx')
```

6. Iterando Sobre Células

Você também pode iterar sobre linhas ou colunas:

```
# Iterar sobre as linhas de uma coluna específica
for row in sheet['A']:
    print(row.value)

# Iterar sobre todas as células de uma planilha
for row in sheet.iter_rows(min_row=1, max_row=10, min_col=1, max_col=5):
    for cell in row:
        print(cell.value)
```

7. Criando uma Nova Planilha

Você pode criar uma nova planilha dentro do mesmo arquivo:

```
new_sheet = workbook.create_sheet(title='NovaAba')

# Escrever em uma célula da nova aba
new_sheet['A1'].value = 'Hello, World!'
```

8. Excluindo uma Planilha

Para excluir uma planilha:

```
# Excluir uma aba do workbook
workbook.remove(workbook['NomeDaAba'])
```

9. Fechando o Arquivo

Embora o *openpyxl* não exija explicitamente que você feche o arquivo, você pode fazê-lo para garantir que os recursos sejam liberados:

```
# Fechar o workbook (opcional)
workbook.close()
```

Resumo

Com *openpyxl*, você pode realizar uma ampla gama de operações em planilhas do Excel, desde abrir e ler dados, até modificar células, salvar arquivos e até mesmo criar novas planilhas. Se você precisa trabalhar com grandes volumes de dados ou realizar operações mais complexas, a integração com *pandas* também é uma boa opção para facilitar a manipulação dos dados.

Manipulação de Tags HTML com Python

Manipular tags HTML com Python pode ser feito usando bibliotecas como **BeautifulSoup** (parte do pacote `bs4`) ou **lxml**. Essas bibliotecas permitem analisar, modificar, e navegar por documentos HTML e XML. Aqui está um guia sobre como manipular tags HTML com Python usando *BeautifulSoup*:

1. Instalando a Biblioteca

Primeiro, você precisa instalar o BeautifulSoup e o parser `lxml` (opcional):

```
pip install beautifulsoup4 lxml
```

2. Carregando e Analisando HTML

Aqui está um exemplo básico de como carregar e analisar um arquivo HTML:

```
from bs4 import BeautifulSoup

# Exemplo de HTML
html_doc = """


Aqui é um título
```

```
Era uma vez uma história curta.


Primeira história
Segunda história

"""

# Criando um objeto BeautifulSoup
soup = BeautifulSoup(html_doc, 'lxml')

# Imprimindo o HTML formatado
print(soup.prettify())
```

3. Navegando pelo Documento

Você pode navegar pelo documento HTML usando métodos e atributos do BeautifulSoup:

```
 # Acessando o título da página
print(soup.title.string)

# Acessando o primeiro parágrafo com classe "title"
print(soup.find('p', class_='title'))

# Acessando todos os links (tags )
for link in soup.find_all('a'):
    print(link.get('href'))
```

4. Modificando o HTML

Você pode modificar o HTML adicionando, removendo ou alterando tags e atributos:

```
 # Adicionando um novo link
new_link = soup.new_tag('a', href='http://example.com/story3', class_='link')
new_link.string = 'Terceira história'
soup.body.append(new_link)

# Modificando o texto de um parágrafo
p = soup.find('p', class_='story')
p.string = 'Era uma vez uma história diferente.'

# Removendo uma tag
soup.a.decompose()  # Remove o primeiro link
```

5. Salvando as Modificações

Após modificar o HTML, você pode salvar as alterações em um novo arquivo:

```
with open('novo_arquivo.html', 'w', encoding='utf-8') as file:
    file.write(str(soup))
```

6. Exemplos Avançados

Adicionar atributos a uma tag:

```
tag = soup.find('a')
tag['style'] = 'color:red;'
```

Inserir nova tag em um lugar específico:

```
# Inserir antes de uma tag existente
soup.body.insert(1, new_link)
```

Remover todas as tags de um tipo específico:

```
for tag in soup.find_all('a'):
    tag.decompose()
```

Resumo

Com o *BeautifulSoup*, você pode facilmente acessar e manipular qualquer parte de um documento HTML, permitindo automatizar tarefas como web scraping, geração de HTML dinâmico, e muito mais.