

Excel com Python: A Revolução das Planilhas



O Poder da Automação nas Planilhas

Potencialize suas Planilhas Excel com Python

Usar Python no Excel é uma maneira incrível de otimizar suas planilhas e automatizar tarefas repetitivas. Com as bibliotecas certas, como openpyxl e pandas, você pode transformar uma simples planilha em uma ferramenta poderosa.

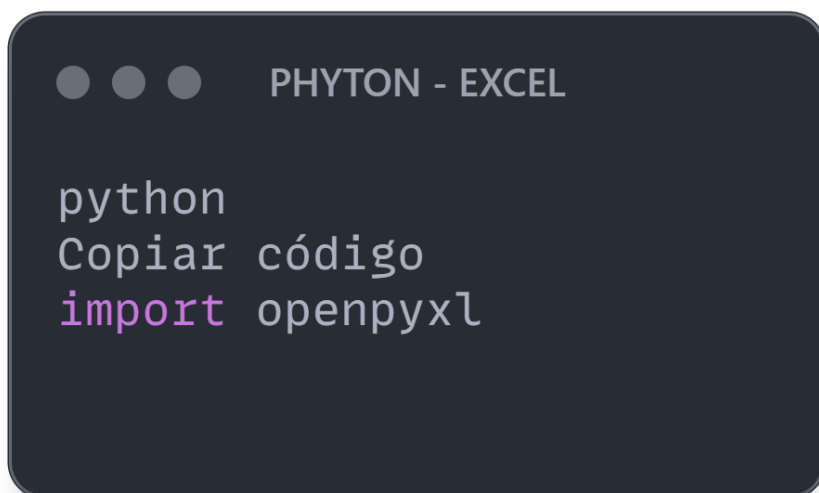
Neste eBook, vamos explorar como aplicar Python no seu Excel com exemplos simples e práticos.

01

Automatizando Tarefas Repetitivas

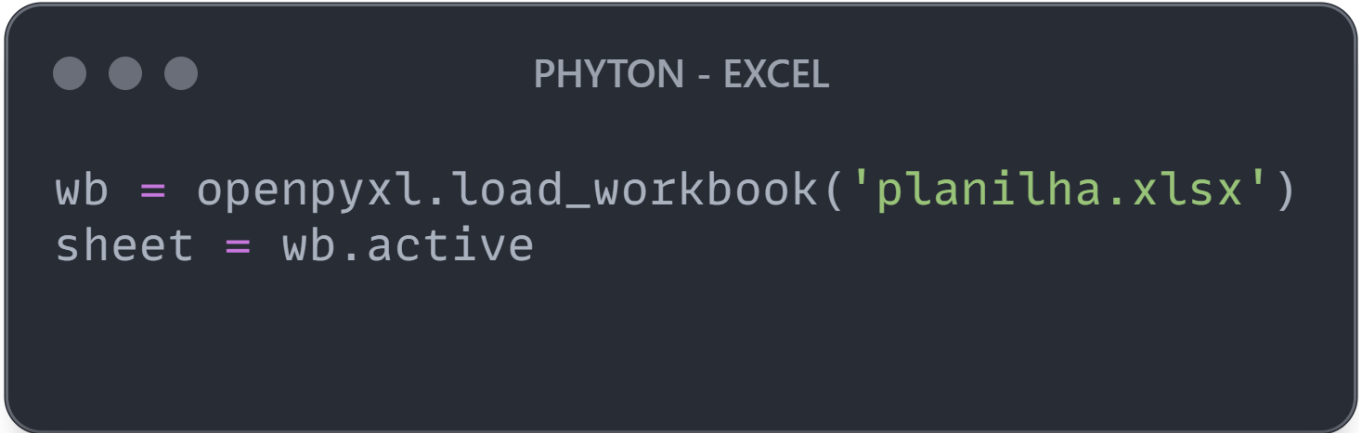
Se você costuma realizar a mesma tarefa várias vezes em suas planilhas, Python pode automatizar isso de maneira rápida.

Exemplo: Limpar uma Coluna de Dados




```
python
Copiar código
import openpyxl
```

Carregar a planilha



```
wb = openpyxl.load_workbook('planilha.xlsx')  
sheet = wb.active
```

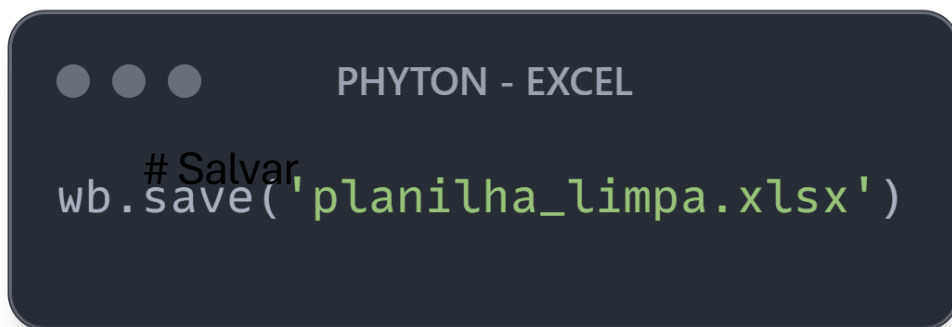
Limpar os dados da coluna A



PHYTON - EXCEL

```
for row in sheet['A']:  
    row.value = None
```

Salvar a planilha modificada



```
● ● ● PHYTON - EXCEL
wb.save('# Salvar
        'planilha_limpa.xlsx')
```

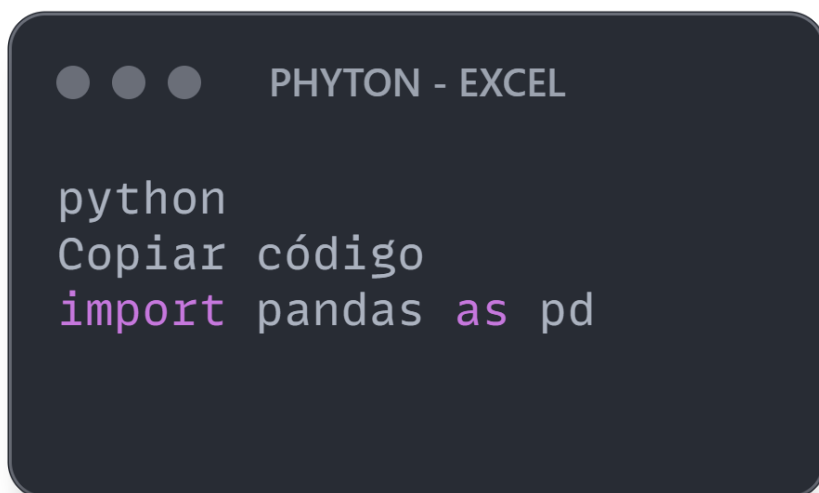
Com esse código, você limpa a coluna A de uma planilha, algo que seria tedioso fazer manualmente. A automação é simples, mas poderosa.

02

Analizando Dados de Forma Rápida

Com Python, você pode fazer cálculos e análises complexas em segundos, algo que no Excel poderia levar mais tempo.

Exemplo: Calcular a Média de Vendas



```
python
Copiar código
import pandas as pd
```

Carregar a planilha

A dark-themed code editor window with a title bar that says "PHYTON - EXCEL". Inside the window, the following Python code is written:

```
df = pd.read_excel('vendas.xlsx')
```

```
df = pd.read_excel('vendas.xlsx')
```

Calcular a média das vendas

PHYTON - EXCEL

```
media_vendas = df['Vendas'].mean()  
print(f'Média das vendas: {media_vendas}')
```

Este código calcula rapidamente a média das vendas de uma planilha, uma tarefa comum que pode ser feita com apenas uma linha de código.

03

Criando Gráficos para Visualizar Dados

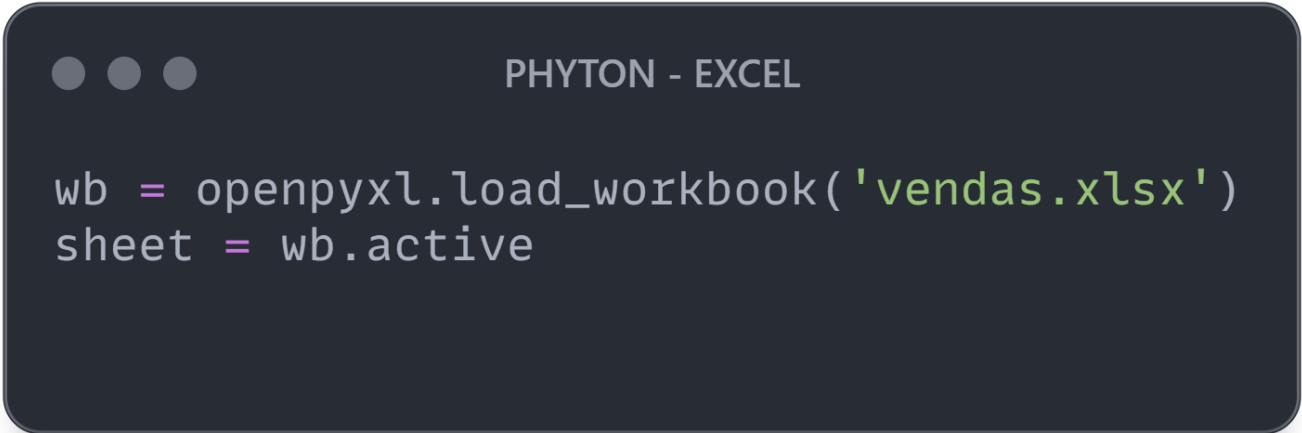
Gráficos são essenciais para entender os dados. Com Python, você pode criar gráficos diretamente no Excel, sem precisar usar o próprio Excel manualmente.

Exemplo: Gerar um Gráfico de Barras

PHYTON - EXCEL

```
python
Copiar código
import openpyxl
from openpyxl.chart import BarChart, Reference
```

Carregar a planilha



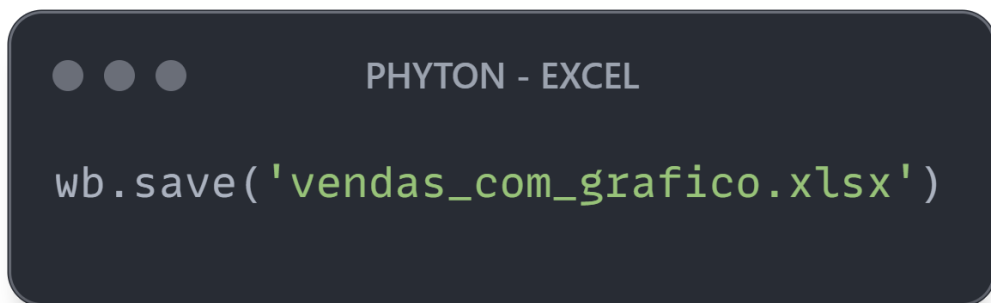
```
wb = openpyxl.load_workbook('vendas.xlsx')  
sheet = wb.active
```

Criar um gráfico de barras

```
PHYTON - EXCEL

grafico = BarChart()
dados = Reference(sheet, min_col=2, min_row=1, max_row=6, max_col=2)
grafico.add_data(dados, titles_from_data=True)
sheet.add_chart(grafico, 'E5')
```

Salvar a planilha com o gráfico



```
PHYTON - EXCEL  
wb.save('vendas_com_grafico.xlsx')
```

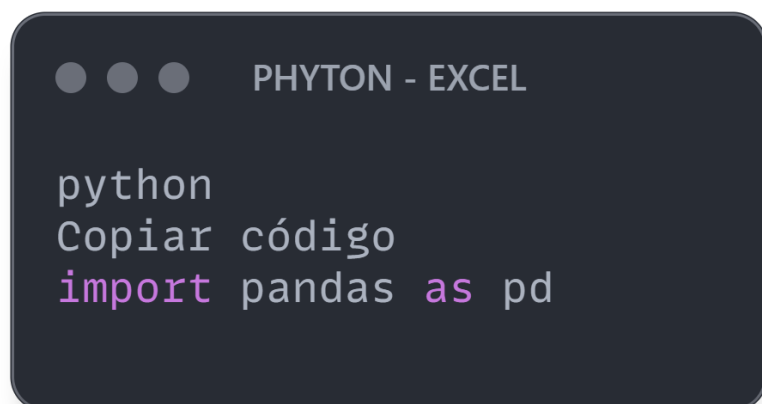
Esse código gera um gráfico de barras diretamente na planilha, facilitando a visualização dos dados sem precisar sair do Excel.

04

Filtrando e Organizando Dados


Filtrar dados de acordo com certos critérios é uma tarefa comum no Excel. Python torna isso ainda mais rápido e fácil.

Exemplo: Filtrar Vendas Acima de R\$ 1.000



```
python  
Copiar código  
import pandas as pd
```

Carregar os dados



```
df = pd.read_excel('vendas.xlsx')
```

Filtrar vendas acima de R\$ 1.000



PHYTON - EXCEL

```
vendas_acima_1000 = df[df['Vendas'] > 1000]
```

Salvar o resultado

PHYTON - EXCEL

```
vendas_acima_1000.to_excel('vendas_filtradas.xlsx', index=False)
```

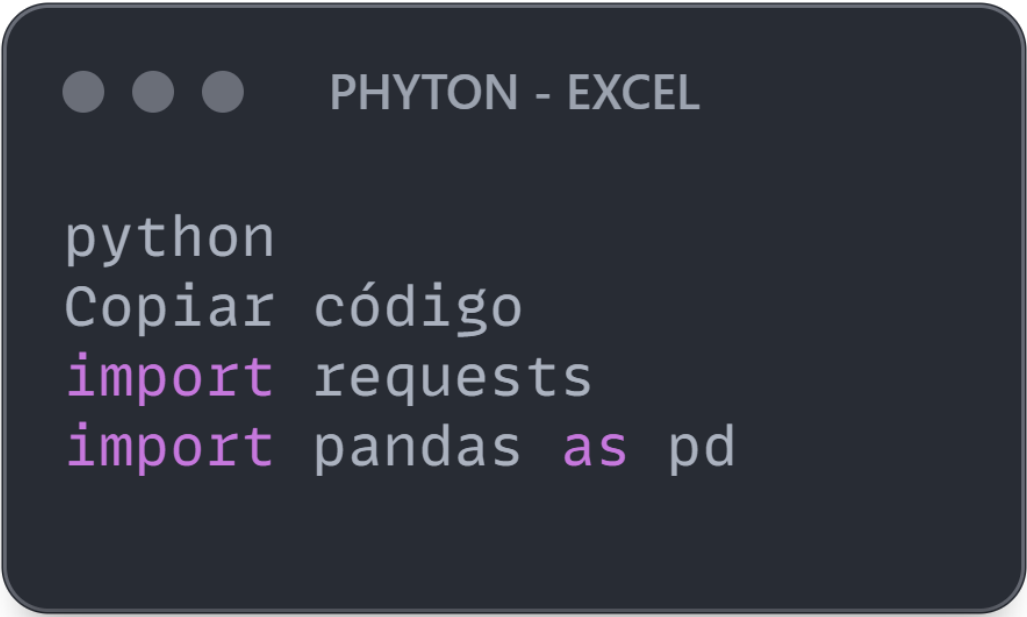
Aqui, o código filtra todas as vendas que são maiores que R\$ 1.000 e salva esses dados em uma nova planilha.

05

Integração com APIs para Dados em Tempo Real

Você pode conectar seu Excel com APIs para atualizar dados em tempo real, como taxas de câmbio, dados financeiros, ou outras informações externas.

Exemplo: Atualizar Taxa de Câmbio



```
python
Copiar código
import requests
import pandas as pd
```

Obter a taxa de câmbio USD para BRL



PHYTON - EXCEL

```
url = 'https://api.exchangerate-api.com/v4/latest/USD'  
response = requests.get(url)  
dados = response.json()
```


Carregar dados do Excel



PHYTON - EXCEL

```
df = pd.read_excel('cambio.xlsx')
```

Atualizar a coluna de Taxa de Câmbio



PHYTON - EXCEL

```
df['Taxa de Câmbio'] = dados['rates']['BRL']
```

Salvar a planilha com a taxa atualizada



PHYTON - EXCEL

```
df.to_excel('cambio_atualizado.xlsx', index=False)
```

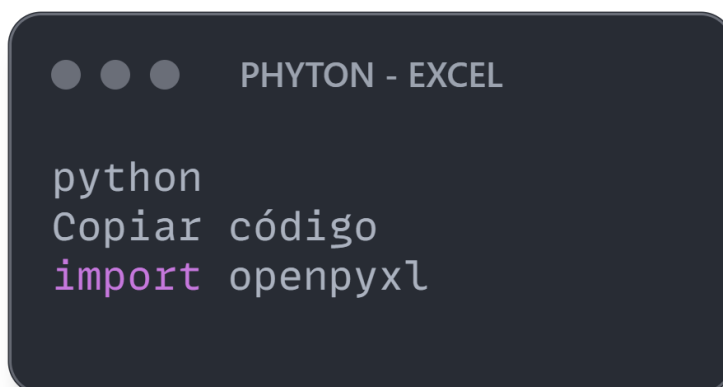
Com esse código, a planilha é automaticamente atualizada com a taxa de câmbio mais recente, sem que você precise buscar essa informação manualmente.

06

Criando Funções Personalizadas para Seus Cálculos


Python permite criar funções personalizadas, o que pode ser muito útil para cálculos específicos que você usa frequentemente.

Exemplo: Função de Cálculo de Desconto



```
python
Copiar código
import openpyxl
```

Função de desconto

A dark-themed code editor window with the title 'PHYTON - EXCEL'. It contains a Python function definition for applying a discount. The function is named 'aplicar_desconto' and takes two parameters: 'preco' (price) and 'desconto' (discount). It returns the price multiplied by (1 minus the discount).

```
def aplicar_desconto(preco, desconto):  
    return preco * (1 - desconto)
```

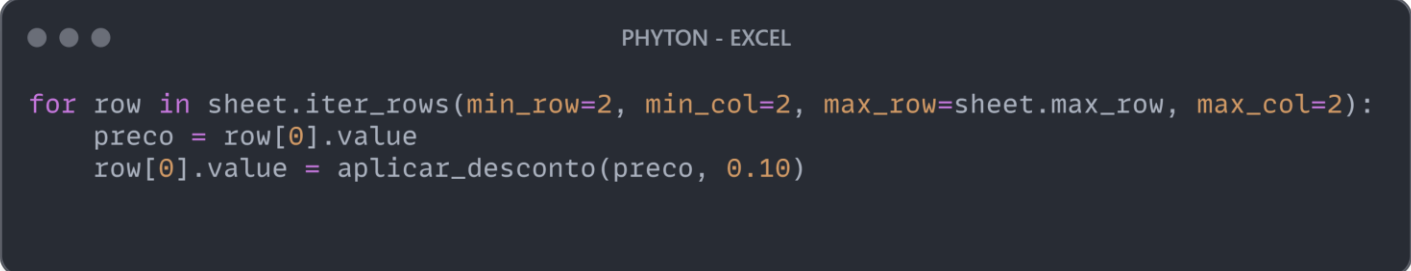
Carregar a planilha



PHYTON - EXCEL

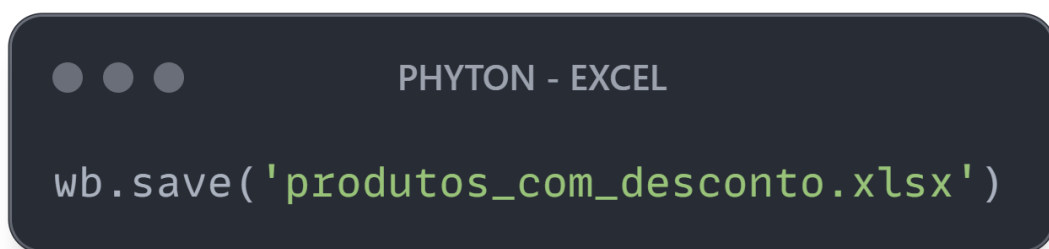
```
wb = openpyxl.load_workbook('produtos.xlsx')  
sheet = wb.active
```

Aplicar 10% de desconto nos preços

A dark-themed code editor window titled "PHYTON - EXCEL" with three window control buttons (red, yellow, green) in the top-left corner. The code inside is a Python loop that iterates over rows in an Excel sheet starting from row 2, column 2, to the maximum row and column. It retrieves the price value from the first cell of each row and applies a 10% discount using a function named "aplicar_desconto".

```
for row in sheet.iter_rows(min_row=2, min_col=2, max_row=sheet.max_row, max_col=2):  
    preco = row[0].value  
    row[0].value = aplicar_desconto(preco, 0.10)
```


Salvar a planilha com desconto

A dark-themed code editor window with a title bar that says "PHYTON - EXCEL". Inside the window, there is a single line of Python code: `wb.save('produtos_com_desconto.xlsx')`. The code is written in a light green color on a dark background.

```
wb.save('produtos_com_desconto.xlsx')
```

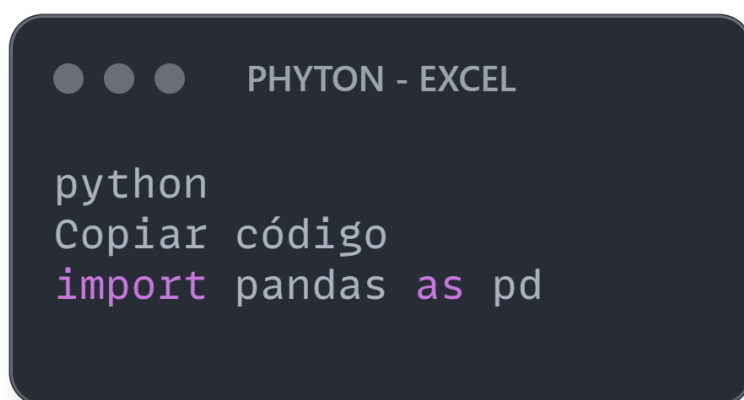
Aqui, criamos uma função para aplicar um desconto nos preços dos produtos e atualizamos a planilha com o valor descontado.

07

Gerando Relatórios Automáticos


Gerar relatórios de vendas, faturamento ou outros dados pode ser feito de forma totalmente automática com Python.

Exemplo: Função de Cálculo de Desconto



```
python
Copiar código
import pandas as pd
```

Carregar os dados

A terminal window with a dark background and rounded corners. At the top, there are three small gray circles followed by the title 'PYTHON - EXCEL'. Below the title, a single line of Python code is displayed: `df = pd.read_excel('vendas.xlsx')`. The code uses syntax highlighting: `df` is purple, `=` is pink, `pd.read_excel` is light gray, and the string `'vendas.xlsx'` is green.

```
df = pd.read_excel('vendas.xlsx')
```

Adicionar coluna de mês



PHYTON - EXCEL

```
df['Mês'] = df['Data'].dt.month
```

Gerar relatório de vendas por mês

PHYTON - EXCEL

```
relatorio = df.groupby('Mês')['Vendas'].sum()
```

Salvar o relatório



PHYTON - EXCEL

```
relatorio.to_excel('relatorio_vendas.xlsx')
```

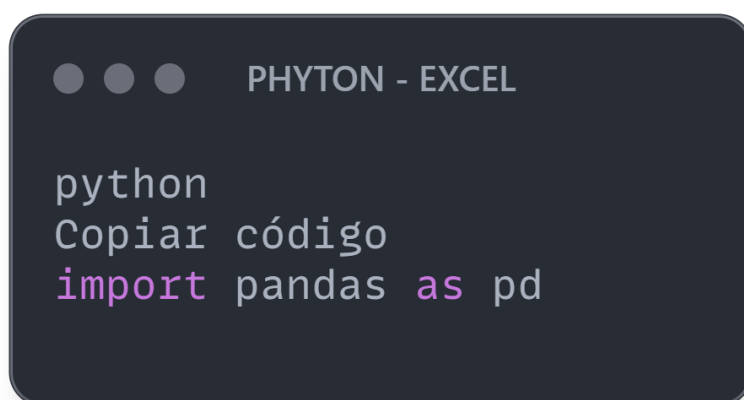
Com este código, você pode gerar um relatório mensal de vendas sem esforço, agrupando e somando os valores por mês.

08

Acelerando o Processamento de Grandes Volumes de Dados

Se você trabalha com grandes volumes de dados, o Python pode tornar o processamento muito mais rápido do que o Excel tradicional.

Exemplo: Processando Dados em Blocos



```
python
Copiar código
import pandas as pd
```

Processar dados em blocos

```
chunksize = 10000 # Processar 10.000 linhas de cada vez
for chunk in pd.read_excel('grandes_dados.xlsx', chunksize=chunksize):
    # Processar e salvar cada parte
    resultado = chunk[chunk['Vendas'] > 500]
    resultado.to_excel('dados_processados.xlsx', mode='a', header=False)
```

Esse código lê os dados em blocos menores, o que facilita o processamento de grandes volumes de informações, sem travar o Excel.

09

Conclusão

Integrar Python com Excel oferece um leque de possibilidades que tornam suas planilhas muito mais potentes e eficientes. Desde a automação de tarefas simples até a execução de análises complexas, Python ajuda a otimizar seu tempo e a obter resultados mais rápidos e precisos. Com os exemplos que você aprendeu, agora você pode começar a turbinar suas planilhas e explorar ainda mais o potencial dessa combinação.

10

Conclusão

ESTE EBOOK FOI CRIADO POR IA, E
DIAGRAMADO POR HUMANO.

ESTE CONTEÚDO FOI CRIADO COM FINS DE
APRENDIZADO DE USO DAS FERRAMENTAS
DE IA, SEM QUE TENHA SIDO REALIZADA A
REVISÃO/VALIDAÇÃO CUIDADOSA DAS
INFORMAÇÕES GERADAS POR IA, PODENDO
CONTER ERROS.