Excel em Pandas

CSV parece ser o formato mais popular para armazenar dados entre cientistas de dados. E isso é compreensível, ele faz o trabalho feito e é um formato bastante simples; em Python, mesmo sem biblioteca, pode-se construir um simples analisador CSV em menos de 10 linhas de código.

Lendo arquivos do Excel

A maneira mais simples de ler arquivos Excel em DataFrames Pandas é usando a seguinte função:

```
import pandas as pd

df = pd.read_excel('path_to_excel_file', sheet_name='…')
```

Onde pode ser o nome da folha que queremos ler, é índice ou uma lista com todas as folhas que queremos ler; os elementos da lista podem ser misturados: nomes de folhas ou índices. Se quisermos todos os lençóis, podemos usar. No caso em que queremos que mais folhas sejam lidas, elas serão devolvidas como um dicionário de quadros de dados. As chaves de tal dicionário serão o índice ou o nome de uma folha, dependendo de como especificamos.

Pacote xlrd

Pandas não podem ler arquivos excel por conta própria, então precisamos instalar outro pacote python para fazer isso.

Assim, vamos instalar o pacote **xlrd.**

```
pip install xlrd

pd.read_excel(
```

Agora, se tentarmos ler os mesmos dados novamente:

```
df = pd.read_excel('crimes_data_england_and_wales.xlsx', sheet_name='Table D1')
df
```

	Table D1: Proportion of adults who were victims of all CSEW crime and personal crime (including fraud and computer misuse), by personal characteristics, year ending March 2020 CSEW1,2	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	England and Wales	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Adults aged 16 and over
2	Personal characteristic3	All CSEW crime (including fraud and computer m...	Personal crime (including fraud and computer m...	Unweighted base - number of adults	NaN	Personal characteristic3	All CSEW crime (including fraud and computer m...	Personal crime (including fraud and computer m...	Unweighted base - number of adults
3	NaN	Percentage, victims once or more	NaN	NaN	NaN	NaN	Percentage, victims once or more	NaN	NaN
4	ALL ADULTS	19.2891	11.0121	33735	NaN	Respondent's employment status	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
77	7. The definition of disability used is consis...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
78	8. The terminology used to label this data has...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
79	9. 'Gender identity the same as sex registered...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
80	10. A question on gender identity was added to...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
81	"-" indicates that data are not reported becau...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

82 rows × 9 columns

Funciona.

Mas os arquivos do Excel podem ser um pouco mais confusos. Além dos dados, eles podem ter outros comentários/explicações na primeira e/ou últimas filas.

Para dizer aos pandas para começar a ler uma folha do Excel a partir de uma linha específica, use o argumento **header=0**, indexada por onde começar a ler. Por padrão, **header=0** e a primeira linha desse tipo é usada para dar os nomes das colunas de quadro de dados.

Para pular linhas no final de uma folha, use **skipfooter = número de linhas para pular**

Por exemplo:

```
df = pd.read_excel('crimes_data_england_and_wales.xlsx', sheet_name='Table D1', header=3, skipfooter=12)
df
```

	Personal characteristic3	All CSEW crime (including fraud and computer misuse)4,5	Personal crime (including fraud and computer misuse)6	Unweighted base - number of adults	Unnamed: 4	Personal characteristic3.1	All CSEW crime (including fraud and computer misuse)4,5.1	Personal crime (including fraud and computer misuse)6.1	Unweighted base - number of adults.1
0	NaN	Percentage, victims once or more	NaN	NaN	NaN	NaN	Percentage, victims once or more	NaN	NaN
1	ALL ADULTS	19.2891	11.0121	33735.0	NaN	Respondent's employment status	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	In employment	21.7593	12.440790	19374.0
3	16-24	23.74	14.6502	2103.0	NaN	Unemployed	23.9912	14.636249	536.0
4	25-34	22.253	12.7872	4836.0	NaN	Economically inactive	14.5918	8.269260	13760.0
...	...	...	...	...	...	...	...	...	...
62	Cohabiting	21.7998	11.788	3505.0	NaN	1-3 times a month	28.3309	17.867971	1428.0
63	Single	22.1638	13.6448	7575.0	NaN	4+ times a month	27.5407	18.237359	203.0
64	Separated	22.6005	13.8343	953.0	NaN	NaN	NaN	NaN	NaN
65	Divorced/legally dissolved partnership	21.2843	12.3225	2959.0	NaN	NaN	NaN	NaN	NaN
66	Widowed	9.38007	5.68372	3330.0	NaN	NaN	NaN	NaN	NaN

67 rows × 9 columns

Isto é um pouco melhor. Ainda existem algumas questões específicas para esses dados. Dependendo do que queremos alcançar, também podemos precisar reorganizar os valores de dados para outra maneira. Mas neste artigo, vamos focar apenas na leitura e escrita de e para quadros de dados.

Outra maneira de ler arquivos Excel além do acima é usando um objeto. Tal objeto pode ser construído usando o construtor. Um objeto pode ser usado de algumas maneiras. Em primeiro lugar, ele tem um atributo que é uma lista de todos os nomes de folhas dentro do arquivo Excel aberto.

```
xlsx = pd.ExcelFile('crimes_data_england_and_wales.xlsx')
```

```
xlsx.sheet_names
```

```
['Notes',
 'Table D1',
 'Table D2',
 'Table D3',
 'Table D4',
 'Table D5',
 'Table D6',
 'Table D7',
 'Table D8',
 'Table D9',
 'Table D10',
 'Table D11',
 'Table D12',
 'Table D13',
 'Table D14',
 'Table D15']
```

Em seguida, este objeto também tem um método que pode ser usado para analisar uma folha do arquivo e retornar um quadro de dados.

O **primeiro parâmetro** deste método pode ser **o índice da folha que queremos analisar ou seu nome**. O resto dos parâmetros são os mesmos da função.

Um exemplo de análise da segunda folha (**índice 1**):

```
df = xlsx.parse(1, header=3, skipfooter=12)
df
```

	Personal characteristic3	All CSEW crime (including fraud and computer misuse)4,5	Personal crime (including fraud and computer misuse)6	Unweighted base - number of adults	Unnamed: 4	Personal characteristic3.1	All CSEW crime (including fraud and computer misuse)4,5.1	Personal crime (including fraud and computer misuse)6.1	Unweighted base - number of adults.1
0	NaN	Percentage, victims once or more	NaN	NaN	NaN	NaN	Percentage, victims once or more	NaN	NaN
1	ALL ADULTS	19.2891	11.0121	33735.0	NaN	Respondent's employment status	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	In employment	21.7593	12.440790	19374.0
3	16-24	23.74	14.6502	2103.0	NaN	Unemployed	23.9912	14.636249	536.0
4	25-34	22.253	12.7872	4836.0	NaN	Economically inactive	14.5918	8.269260	13760.0
...	...	...	...	...	...	...	...	...	...
62	Cohabiting	21.7998	11.788	3505.0	NaN	1-3 times a month	28.3309	17.867971	1428.0
63	Single	22.1638	13.6448	7575.0	NaN	4+ times a month	27.5407	18.237359	203.0
64	Separated	22.6005	13.8343	953.0	NaN	NaN	NaN	NaN	NaN
65	Divorced/legally dissolved partnership	21.2843	12.3225	2959.0	NaN	NaN	NaN	NaN	NaN
66	Widowed	9.38007	5.68372	3330.0	NaN	NaN	NaN	NaN	NaN

67 rows × 9 columns

... e aqui analisamos a mesma folha **usando seu nome em vez de um índice**:

```
df = xlsx.parse('Table D1', header=3, skipfooter=12)
df
```

	Personal characteristic3	All CSEW crime (including fraud and computer misuse)4,5	Personal crime (including fraud and computer misuse)6	Unweighted base - number of adults	Unnamed: 4	Personal characteristic3.1	All CSEW crime (including fraud and computer misuse)4,5.1	Personal crime (including fraud and computer misuse)6.1	Unweighted base - number of adults.1
0	NaN	Percentage, victims once or more	NaN	NaN	NaN	NaN	Percentage, victims once or more	NaN	NaN
1	ALL ADULTS	19.2891	11.0121	33735.0	NaN	Respondent's employment status	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	In employment	21.7593	12.440790	19374.0
3	16-24	23.74	14.6502	2103.0	NaN	Unemployed	23.9912	14.636249	536.0
4	25-34	22.253	12.7872	4836.0	NaN	Economically inactive	14.5918	8.269260	13760.0
...	...	...	...	...	...	...	...	...	...
62	Cohabiting	21.7998	11.788	3505.0	NaN	1-3 times a month	28.3309	17.867971	1428.0
63	Single	22.1638	13.6448	7575.0	NaN	4+ times a month	27.5407	18.237359	203.0
64	Separated	22.6005	13.8343	953.0	NaN	NaN	NaN	NaN	NaN
65	Divorced/legally dissolved partnership	21.2843	12.3225	2959.0	NaN	NaN	NaN	NaN	NaN
66	Widowed	9.38007	5.68372	3330.0	NaN	NaN	NaN	NaN	NaN

67 rows × 9 columns

pd.ExcelFile() também pode ser usado em declarações internas, e se você quiser fazer algo um pouco mais elaborado, como analisar apenas folhas com 2 palavras em seu nome, você pode fazer algo como **as:**

```
crime_data = {}
with pd.ExcelFile('crimes_data_england_and_wales.xlsx') as xlsx:
    for sheet_name in xlsx.sheet_names:
        if len(sheet_name.split(' ')) == 2:
            crime_data[sheet_name] = xlsx.parse(sheet_name)
```

A mesma coisa que você pode fazer usando em vez de método, assim:

```
crime_data2 = {}
with pd.ExcelFile('crimes_data_england_and_wales.xlsx') as xlsx:
    for sheet_name in xlsx.sheet_names:
        if len(sheet_name.split(' ')) == 2:
            crime_data2[sheet_name] = pd.read_excel(xlsx, sheet_name)
```

... ou, se você simplesmente quer todas as folhas, você pode fazer:

```
: crime_data3 = pd.read_excel('crimes_data_england_and_wales.xlsx', None)
```

Escrevendo arquivos excel

Agora que sabemos ler arquivos excel, o próximo passo para nós é ser capaz de também escrever um quadro de dados para um arquivo excel. Podemos fazer isso usando o método de data frame.

Vamos primeiro criar um quadro de dados simples para escrever em um arquivo excel:

```
data_to_write = [
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12]
]
df = pd.DataFrame(data_to_write, columns=['A', 'B', 'C', 'D'])
df
```

	A	B	C	D
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Agora queremos escrevê-lo em um arquivo excel:

```
df.to_excel('output.xlsx', sheet_name='sheet1')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-5-d11fe1de4685> in <module>
----> 1 df.to_excel('output.xlsx', sheet_name='sheet1')

~\AppData\Roaming\Python\Python37\site-packages\pandas\core\generic.py in to_excel(self, excel_writer, sheet_name, na_rep, float_format, columns, header, index, index_label, startrow, startcol, engine, merge_cells, encoding, inf_rep, verbose, freeze_panes)
   2179             startcol=startcol,
   2180             freeze_panes=freeze_panes,
-> 2181             engine=engine,
   2182         )
   2183

~\AppData\Roaming\Python\Python37\site-packages\pandas\io\formats\excel.py in write(self, writer, sheet_name, startrow, startcol, freeze_panes, engine)
    724             need_save = False
    725         else:
--> 726             writer = ExcelWriter(stringify_path(writer), engine=engine)
    727             need_save = True
    728

~\AppData\Roaming\Python\Python37\site-packages\pandas\io\excel\_openpyxl.py in __init__(self, path, engine, mode, **engine_kwargs)
     16     def __init__(self, path, engine=None, mode="w", **engine_kwargs):
     17         # Use the openpyxl module as the Excel writer.
---> 18         from openpyxl.workbook import Workbook
     19
     20         super().__init__(path, mode=mode, **engine_kwargs)

ModuleNotFoundError: No module named 'openpyxl'
```

... e temos um erro.

Mais uma vez, pandas não podem escrever para superar arquivos por conta própria; precisamos de outro pacote para isso. As principais opções que temos são:

- **xlwt**

  — funciona apenas com arquivos Excel 2003 (.xls); modo apêndice não suportado

- **xlsxwriter**

  — funciona apenas com arquivos Excel 2007+ (.xlsx); modo apêndice não suportado

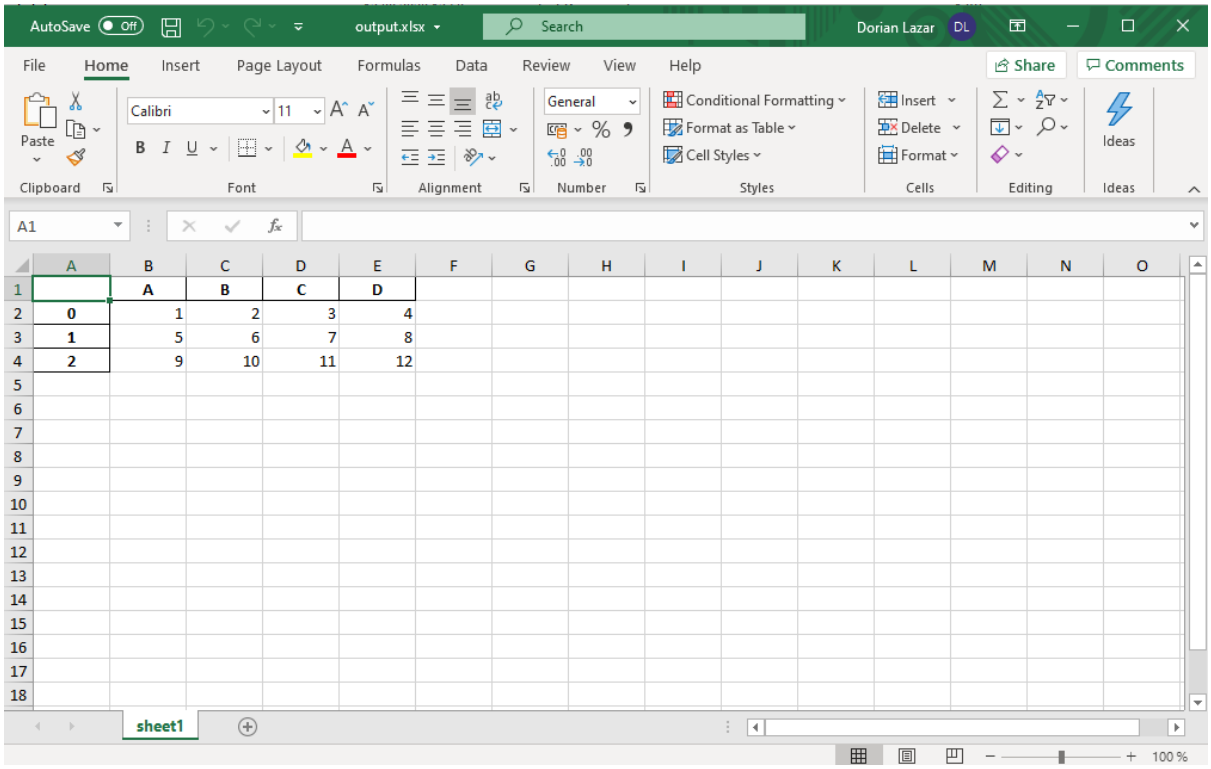
- **openpyxl**

  — funciona apenas com arquivos Excel 2007+ (.xlsx); suporta o modo apêndice

Se quisermos ser capazes de escrever para o formato de .xls antigo devemos instalar, pois é o único que lida com esses arquivos. Para .xlsx arquivos, vamos escolher, pois ele também suporta o modo de apêndice:

```
pip install xlwt openpyxl
```

Agora, se executarmos novamente o código acima, ele funciona; um arquivo excel foi criado:

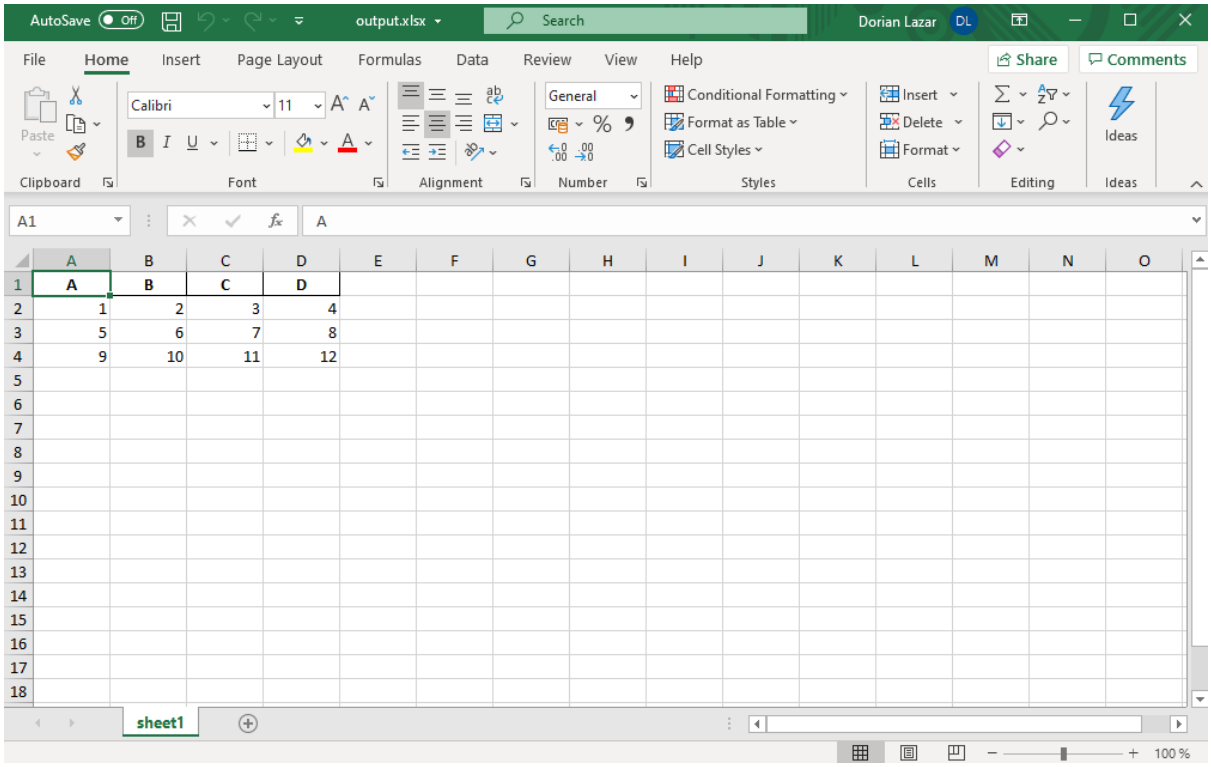


	A	B	C	D	E
1		A	B	C	D
2	0	1	2	3	4
3	1	5	6	7	8
4	2	9	10	11	12

Por padrão, pandas também escreve a coluna de índice junto com nossas colunas. Para se livrar dele, use como no código abaixo:

```
df.to_excel('output.xlsx', sheet_name='sheet1', index=False)
```

A coluna de índice não está lá agora:



E se quisermos escrever mais planilhas? Se quisermos adicionar uma segunda folha ao arquivo anterior, você acha que o código abaixo funcionará?

```
df.to_excel('output.xlsx', sheet_name='sheet2', index=False)
```

A resposta é **não**. Ele apenas substituirá o arquivo com apenas uma folha: folha2.

Para escrever mais folhas em um arquivo Excel, precisamos usar um objeto como mostrado abaixo. Primeiro, criamos outro quadro de dados para folha2, depois abrimos um arquivo Excel como um objeto no qual escrevemos os 2 quadros de dados:

```
df2 = pd.DataFrame(data=[[1,2,3],[4,5,6]], columns=['A', 'B', 'C'])
df2
```

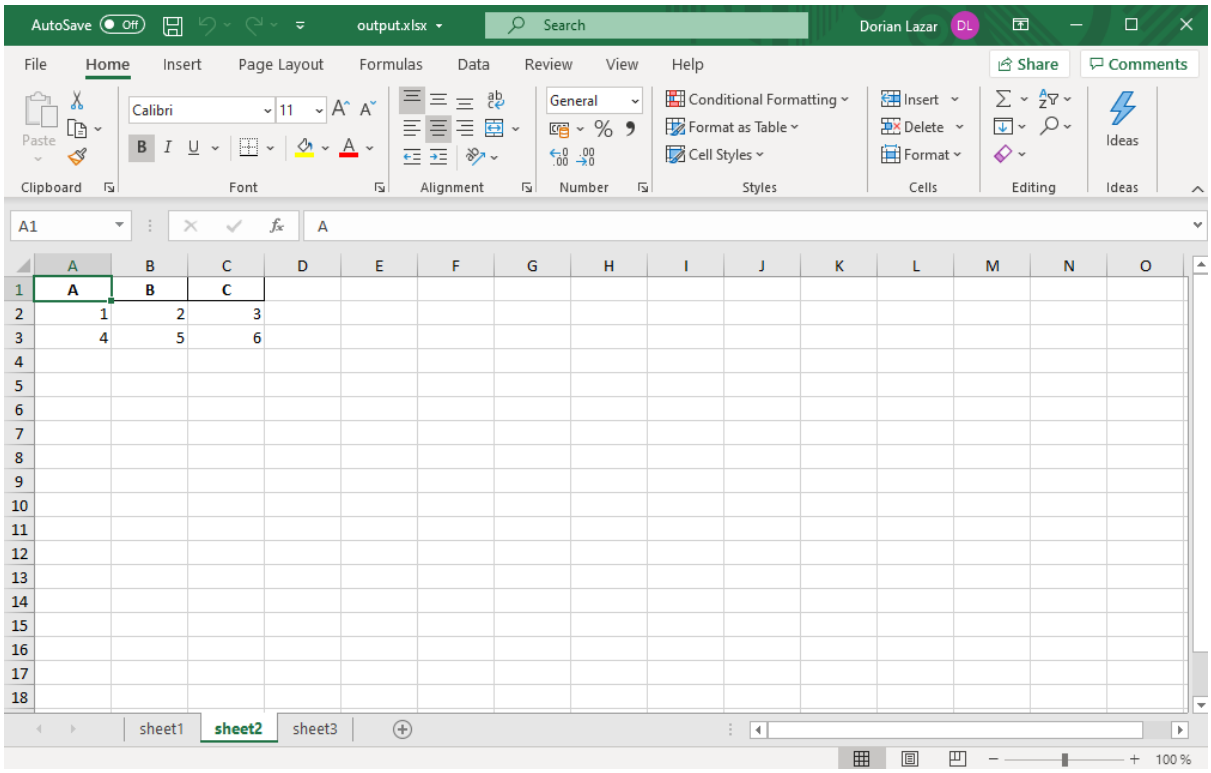
	A	B	C
0	1	2	3
1	4	5	6

```
with pd.ExcelWriter('output.xlsx') as writer:
    df.to_excel(writer, sheet_name='sheet1', index=False)
    df2.to_excel(writer, sheet_name='sheet2', index=False)
```

Agora, nosso arquivo Excel deve ter 2 folhas. Se quisermos adicionar outra folha a ele, precisamos abrir o arquivo no modo de apêndice e executar código semelhante ao anterior. Por exemplo:

```
with pd.ExcelWriter('output.xlsx', mode='a') as writer:
    df.to_excel(writer, sheet_name='sheet3', index=False)
```

Nosso arquivo Excel, agora, tem 3 folhas e se parece com isso:



Trabalhando com Fórmulas Excel

Provavelmente você está se perguntando, neste momento, sobre fórmulas Excel. E quanto a eles? Como ler a partir de arquivos que têm fórmulas? Como escrevê-los nos arquivos do Excel?

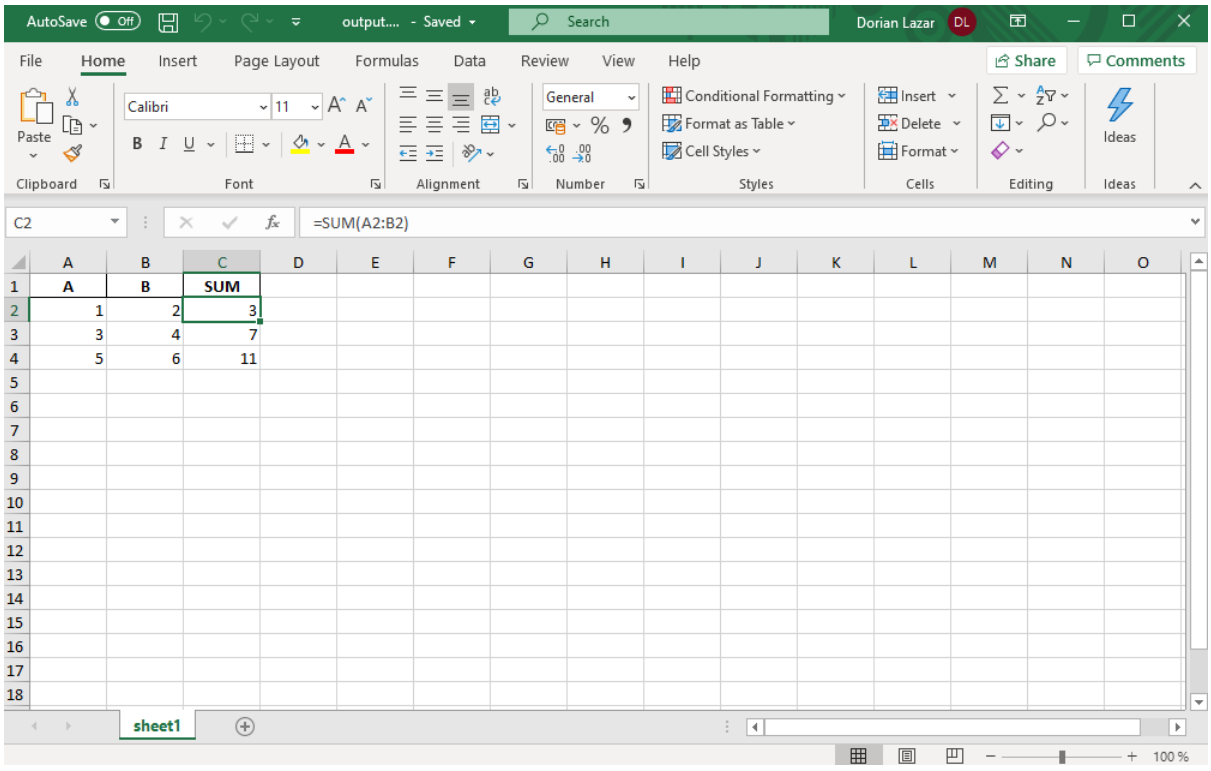
É muito fácil. Escrever fórmulas para arquivos Excel é tão simples quanto apenas escrever a sequência da fórmula, e essas strings serão automaticamente interpretadas pelo Excel como fórmulas. Como exemplo:

```
df = pd.DataFrame(data=[[1, 2, '=SUM(A2:B2)'], [3, 4, '=SUM(A3:B3)'], [5, 6, '=SUM(A4:B4)']],
                  columns=['A', 'B', 'SUM'])
df
```

	A	B	SUM
0	1	2	=SUM(A2:B2)
1	3	4	=SUM(A3:B3)
2	5	6	=SUM(A4:B4)

```
df.to_excel('output.xlsx', sheet_name='sheet1', index=False)
```

O arquivo Excel produzido pelo código acima é:



Agora, se quisermos ler um arquivo Excel com fórmulas nele, os pandas lerão em quadros de dados o resultado dessas fórmulas.

Por exemplo, vamos ler nosso arquivo criado anteriormente:

```
df_read = pd.read_excel('output.xlsx', sheet_name='sheet1')
df_read
```

	A	B	SUM
0	1	2	3
1	3	4	7
2	5	6	11

Às vezes, você precisa salvar o arquivo Excel manualmente para que isso funcione e não obter zeros em vez do resultado de fórmulas (aperte CTRL+S antes de executar o código acima).