clase5

December 12, 2023

1 Clase 5 - XlsxWriter, estructura de un proyecto y ejemplos prácticos

1.1 XlsxWriter

Se instala con `pip install XlsxWriter` o `conda install XlsxWriter`

Es una libreria de python que sirve para crear hojas de calculo de formato "Excel" de forma automatizada y con una capacidad de configuración enorme. Con XlsxWriter vamos a poder crear archivos excel de múltiples hojas a partir de muchos dataframes. Además nos va a dar la posibilidad de crear archivos excel usando casi todo lo que excel nos ofrece. Vamos a poder escribir formulas de excel, dar formato a las celdas, automatizar la creación de gráficos de excel o hasta escribir macros programáticamente. Muchas veces por más automatizados y complejos que sean nuestros procesos es necesario compartirlos en forma de hojas de cálculo simplemente por la facilidad y familiaridad que tiene esa herramienta. Cuándo queremos exportar proyectos complejos a un excel es usual que un excel de una sola hoja con una tabla plana no sea suficiente satisfacer el objetivo del proyecto, para eso podemos usar esta libreria para sacarle más provecho al formato de excel.

¡Vean la documentación acá!: Documentación XlsxWriter

Veamos ejemplos:

[ ]:
```
import xlsxwriter

workbook = xlsxwriter.Workbook('hello.xlsx')
worksheet = workbook.add_worksheet()

worksheet.write('A1', 'Hello world')

workbook.close()
```

En este ejemplo ven como crea un nuevo archivo de excel y explícitamente en escribe algo en la celda 'A1'. Quiero que noten como debe usar el método `.close()` para dejar de trbajar con el archivo. Es super importante no olvidarse de cerrar los archivos porque al igual que cuando uno usa un excel, ninguna otra persona ni proceso puede modificarlo mientras esté abierto. No cerrarlo va a causar que no sea seguro que los procesos que quisimos plasmar en el excel se guarden y tambien puede generar que su programa no pueda terminar correctamente o que directamente se quede abierto imposibilitando usar el archivo.

Por esa razón, para no olvidarse de cerrar el archivo, recomiendo mucho esta otra forma de escribir

el mismo código:

```
[ ]: with xlsxwriter.Workbook('hello.xlsx') as workbook:
         worksheet = workbook.add_worksheet()
         worksheet.write('A1', 'Hello world')
```

De esa forma abrimos o creamos el documento con `with` y una vez que salimos de ese bloque de código, el archivo se cierra de manera segura. Tengan en cuenta este patrón cada vez que en Python encuentren este tipo de objetos que deban ser cerrados por un método como `.close()` ya que posiblemente funcionen con `with` y de esa forma dejemos un código más claro y seguro.

1.1.1 El objeto `Workbook`, `Worksheet` y otros

Fijense como para usar esta librería debemos usar los objetos que nos provee. Para hacer la hoja de cálculo más simple debemos llamar por lo menos a dos objetos. Al objeto `Workbook` que va a representar en el Python a nuestro archivo de excel y al objeto `WorkSheet` que representa a una hoja del excel.

Noten esto: primero llamamos a Workbook con el nombre del archivo que queremos modificar o leer (`workbook = xlsxwriter.Workbook('hello.xlsx')` o de la otra forma equivalente `with xlsxwriter.Workbook('hello.xlsx') as workbook:`). Luego es a partir del objeto que llamamos workbook que podemos traer nuevos objetos que como `Worksheet` a través de los métodos `.add_????()`que van a representar otras funcionalidades del excel. En el ejemplo de arriba creo solo una hoja a la que le escribe algo.

En el siguiente ejemplo escribimos dos hojas y le ponemos formato a las celdas con el objeto `Format`:

```
[ ]: with xlsxwriter.Workbook('hello.xlsx') as libro2:
         hoja1 = libro2.add_worksheet('hoja1')
         hoja2 = libro2.add_worksheet('hoja1')
         formato_celda_resaltada = libro2.add_format({'bold':True,"bg_color":
      ↪"yellow", 'font_color': 'red'})
         hoja1.write('A1', 'Hola mundo', formato_celda_resaltada)
         hoja2.write(0,0,'Chau')
```

Tengamos en cuenta que de la misma manera que puse un texto en una celda, tambien podemos escribir fórmulas de excel. Y asi como puedo agregar hojas al proyecto también puedo agregar gráficos si quisiese.

1.1.2 Pandas y XlsxWriter

Esta libreria funciona muy bien integrada con Pandas, para ello debemos aprovechar el objeto `ExcelWriter` que es la herramienta de pandas para integrarse muchas librerias para exportar los datos. Van a ver que se una muy parecido a los objetos que estabamos viendo antes, en este caso tenemos que indicarle a pandas que estamos usando como motor de exportación a **xlsxwriter**

Recordemos que posteriormente podiamos exportar a excel de la siguiente forma `df.to_excel('tabla.xlsx')`. Pero de esta forma nunca podremos crear hojas de cálculo de varias hojas, ni darle formato, ni usar otras herramientas de excel.

[ ]:
```
import pandas as pd

df1 = pd.DataFrame({'uno':[1,2,3], 'dos':[3,2,1]})
df2 = pd.DataFrame({'hola':['a','b','c'], 'chau':[9,8,7]})

with pd.ExcelWriter('pandas.xlsx', engine='xlsxwriter') as writer:
    # Escribimos los datos en dos hojas distintas del excel
    df1.to_excel(writer,sheet_name='hoja1',index=False)
    df2.to_excel(writer,sheet_name='hoja2',index=False)

    #Usamos métodos de xlsxwriter para editarlo
    # Primero recuperamos los objetos que modelan al excel (libro, hojas,␣
 ↪formato, etc)
    workbook: xlsxwriter.Workbook = writer.book
    hoja1:xlsxwriter.Workbook.worksheet_class = writer.sheets["hoja1"]
    hoja2:xlsxwriter.Workbook.worksheet_class = writer.sheets["hoja2"]

    formato = workbook.add_format({'bold':True,"bg_color":"yellow",␣
 ↪'font_color': 'red'})

    # pintemos con este formato a la primer columna de cada hoja
    hoja1.set_column('A:A', None, formato)
    hoja2.set_column('A:A', None, formato)
```

En este ejemplo de arriba se ve como funcionan perfectamente las dos librerias. Con `pandas` imprimimos los datos y podemos usar a `xlsxwriter` para darle formato sin alterarlos. Vean como usa `None` para indicar a xlsxwriter que no tiene que modificar los datos de las celdas, con `.set_column(<rango>, <datos>,<formato>, etc.)` podemos usar la notación de rangos de columnas de excel como `A:A`

Acá abajo les dejo un ejemplo en el que no usamos `pd.to_excel()` de pandas para imprimir nuestras tablas sino que armamos dos bucles anidados para que recorra la tabla celda por celda. Tengan en cuenta este ejemplo porque en muchos casos que trabajemos con tablas y nos falte un método necesario, podemos hacer este barrido de la tabla como último recurso. Si bien no es lo más eficiente, hace que las cosas funcionen.

[ ]:
```
df = pd.DataFrame({'col1':[1,2,3], 'col2':[3,4,5]})


with pd.ExcelWriter('desde_pandas.xlsx',engine='xlsxwriter') as writer:

    workbook: xlsxwriter.Workbook = writer.book
    hoja1 = workbook.add_worksheet("hoja1")

    formato = workbook.add_format({"color":"red","bold":True})
```

```
    (max_filas, max_columnas)= df.shape


    print('Miren en que orden barre la tabla.')
    for fila in range(max_filas):
        for columna in range(max_columnas):
            print(f'Fila: {fila}, columna: {columna}, valor: {df.
 ↪iloc[fila,columna]}')
            hoja1.write(fila,columna, df.iloc[fila,columna],formato)
```

```
Miren en que orden barre la tabla.
Fila: 0, columna: 0, valor: 1
Fila: 0, columna: 1, valor: 3
Fila: 1, columna: 0, valor: 2
Fila: 1, columna: 1, valor: 4
Fila: 2, columna: 0, valor: 3
Fila: 2, columna: 1, valor: 5
```

Les dejo otro ejemplo más completo. Por ejemplo damos un formato monetario a lsa columnas numéricas. Tengan en cuenta que cuando pandas ya abrio la hoja para escribir los datos, la podemos recuperar esa hoja usando `writer.sheets[<nombre de hoja>]`

[ ]:
```
df1 = pd.DataFrame({'col1': [1.1, 2.2, 3.3], 'col2': [4.4, 5.5, 6.6]})
df2 = pd.DataFrame({'col1': [7.7, 8.8, 9.9], 'col2': [10.10, 11.11, 12.12]})

with pd.ExcelWriter('formato_numerico.xlsx', engine='xlsxwriter') as writer:

    df1.to_excel(writer, sheet_name='sheet1', index=None)
    df2.to_excel(writer, sheet_name='sheet2', index=None)
    workbook:xlsxwriter.Workbook = writer.book

    formato_dinero = workbook.add_format({'num_format': '#.##0,00 [$ARS];-#.
 ↪##0,00 [$ARS]'})
    formato_bordes = workbook.add_format({'border': 1})
    formato_encabezado = workbook.add_format({'bold': True, "bg_color":
 ↪"yellow", 'font_color': 'red'})
    formato_encabezado.set_border(3)
    formato_encabezado.set_align("left")



    for worksheet in writer.sheets.values():
        hoja:xlsxwriter.Workbook.worksheet_class = worksheet
        hoja.set_row(0, None, formato_encabezado)
```

```
        hoja.set_column(0,0, None, formato_bordes)
        hoja.conditional_format('A2:D4', {'type': 'no_blanks', 'format':␣
↪formato_dinero})
```

Hagan la prueba, van a ver que este código funciona casi al 100% pero van a ver que el encabezado no se pintó correctamente. Le podemos dar una vuelta y arreglarlo usando formato condicional en vez de pintar toda la fila. Salvo para este caso donde querramos formatear toda la fila que tiene el encabezado, `.set_row()` funciona correctamente.

[ ]:
```
with pd.ExcelWriter('formato_numerico.xlsx', engine='xlsxwriter') as writer:

    df1.to_excel(writer, sheet_name='sheet1', index=None)
    df2.to_excel(writer, sheet_name='sheet2', index=None)
    workbook:xlsxwriter.Workbook = writer.book

    formato_dinero = workbook.add_format({'num_format': '#.##0,00 [$ARS];-#.
↪##0,00 [$ARS]'})
    formato_bordes = workbook.add_format({'border': 1})
    formato_encabezado = workbook.add_format({'bold': True, "bg_color":
↪"yellow", 'font_color': 'red'})
    formato_encabezado.set_border(3)
    formato_encabezado.set_align("left")



    for worksheet in writer.sheets.values():
        hoja:xlsxwriter.Workbook.worksheet_class = worksheet
        hoja.conditional_format('A1:D1', {'type': 'no_blanks', 'format':␣
↪formato_encabezado})
        hoja.set_column(0,0, None, formato_bordes)
        hoja.conditional_format('A2:D4', {'type': 'no_blanks', 'format':␣
↪formato_dinero})
```

Con ese cambio funciona perfectamente, con formato condicional aplica el estilo si las celdas no están en blanco. La otra alternativa era hacer un loop sobre la primer fila e ir aplicando el estilo celda por celda.

1.2 Ejercicio en clase

El objetivo es leer y limpiar un excel de complejos exportadores para luego exportarlo en otro excel en el que en cada hoja tenga un complejo exportador distinto. Además podemos usar formato condicional en las celdas para diferenciar las variaciones positivas de las negativas.

[ ]:
```
var_complejos = pd.read_excel(
    "https://www.indec.gob.ar/ftp/cuadros/economia/complexp_variacion_2019_2022.
↪xls",
```

```
    header=3).dropna()
var_complejos.head()
```

[ ]:
```
                       Unnamed: 0   2019    2020*    2021*    2022* 2022*/19  \
2          Total de exportaciones  65115  54884.0  77934.0  88446.0     35.8
3           Principales complejos  59486  50066.0  71106.0  82066.0       38
4               Sector oleaginoso  18867  16730.0  26389.0  27989.0     48.3
5                   Complejo soja  16943  14865.0  23841.0  24868.0     46.8
6       Harinas y pellets de soja   8806   7806.0  11796.0  12041.0     36.7

   2022*/21*
2       13.5
3       15.4
4        6.1
5        4.3
6        2.1
```

Las jerarquía de datos esta delimitada por tabulaciones en la columna de descripciones

[ ]:
```
var_complejos.columns = [str(columna).replace('*','') for columna in␣
 ↪var_complejos.columns]

var_complejos_etiquetas = var_complejos.iloc[:,0]

var_complejos = var_complejos.iloc[:,1:].astype(float)

var_complejos_etiquetas = pd.DataFrame({
    "item":var_complejos_etiquetas,
    "nivel":var_complejos_etiquetas.apply(lambda item: (len(item) - len(item.
 ↪lstrip())) // 2)})

maximo_nivel = var_complejos_etiquetas.nivel.max()
for nivel in range(maximo_nivel+1):
    var_complejos_etiquetas[f'nivel{nivel}'] = var_complejos_etiquetas.apply(
        lambda fila: fila['item'] if fila['nivel']==nivel else pd.NA, axis=1)
for nivel in range(maximo_nivel):
    var_complejos_etiquetas[f'nivel{nivel}'] =␣
 ↪var_complejos_etiquetas[f'nivel{nivel}'].ffill()
    var_complejos_etiquetas[var_complejos_etiquetas['nivel'] == nivel] = (
        var_complejos_etiquetas
        .query('nivel == @nivel')
        .ffill(axis=1))
var_complejos_etiquetas = var_complejos_etiquetas.drop('item',axis=1).
 ↪applymap(lambda v: v.lstrip() if (type(v) == str) else v)
var_complejos = pd.concat([var_complejos_etiquetas,var_complejos],axis=1)
```

[ ]:
```
var_complejos.head()
```

[ ]:
```
   nivel                 nivel0                 nivel1  \
2      0  Total de exportaciones  Total de exportaciones
3      0   Principales complejos   Principales complejos
4      0       Sector oleaginoso       Sector oleaginoso
5      1       Sector oleaginoso           Complejo soja
6      2       Sector oleaginoso           Complejo soja

                     nivel2     2019     2020     2021     2022  2022/19  \
2    Total de exportaciones  65115.0  54884.0  77934.0  88446.0     35.8
3     Principales complejos  59486.0  50066.0  71106.0  82066.0     38.0
4         Sector oleaginoso  18867.0  16730.0  26389.0  27989.0     48.3
5             Complejo soja  16943.0  14865.0  23841.0  24868.0     46.8
6  Harinas y pellets de soja   8806.0   7806.0  11796.0  12041.0     36.7

   2022/21
2     13.5
3     15.4
4      6.1
5      4.3
6      2.1
```

Mucho mejor

Realmente este cuadro es un ejemplo de como publicar datos para que sean difíciles de trabajar, todo el código de arriba es la limpieza que hice.

[ ]:
```
var_complejos_agrupada_por_n0 = var_complejos.groupby("nivel0")

with pd.ExcelWriter("complejos_exportadores.xlsx", engine="xlsxwriter") as␣
 ↪writer:
    book:xlsxwriter.Workbook = writer.book
    formato_encabezado = book.add_format({'bold': True, "bg_color":"#ADD8E6"})
    formato_encabezado.set_border(6)
    formato_encabezado.set_align("left")
    for nombre, df in var_complejos_agrupada_por_n0:
        (
            df
            .drop(["nivel","nivel0"],axis=1)
            .to_excel(writer,sheet_name=nombre[:31], index=False))
        hoja: xlsxwriter.Workbook.worksheet_class = writer.sheets[nombre[:31]]
        hoja.conditional_format('A1:H1', {'type': 'no_blanks', 'format':␣
 ↪formato_encabezado})
        hoja.set_column(0,1,20)
```