GUÍA INSTRUCCIONAL – TALLER 2

PYTHON I

Instructor: Neptalí Piña

Email: njpm28@gmail.com

Teléfono: 0414-5227568

Fecha: 01-10-2022

GracoSoft

Centro Empresarial Plaza Madrid, piso 9, oficinas 9-7 a la 9-10

Tabla de contenidos

TALLER 2

Documentos Excel

Una hoja de cálculo (spreadsheet) es llamado libro de trabajo (workbook). Un simple libro de trabajo es guardado en un archivo con extensión .xlsx, cada libro de trabajo puede contener múltiples hojas (sheets) también llamado hojas de trabajo. La hoja de trabajo en la que actualmente está el usuario (o la última en la que estaba cuando cerro el Excel) es llamado hoja de trabajo activa (active sheet).

Cada hoja de trabajo tiene columnas (son identificadas por letras empezando desde la A) y filas (identificadas por números empezando desde el 1). Una caja particular de columna y fila es llamada celda, cada celda contiene un número o un valor de texto. Toda la cuadricula de celdas con datos forma una hoja de trabajo.

Instalando openpyxl

Python no viene con openpyxl, por lo que tendrá que instalarlo, para instalar openpyxl se debe ejecutar el siguiente comando por consola:

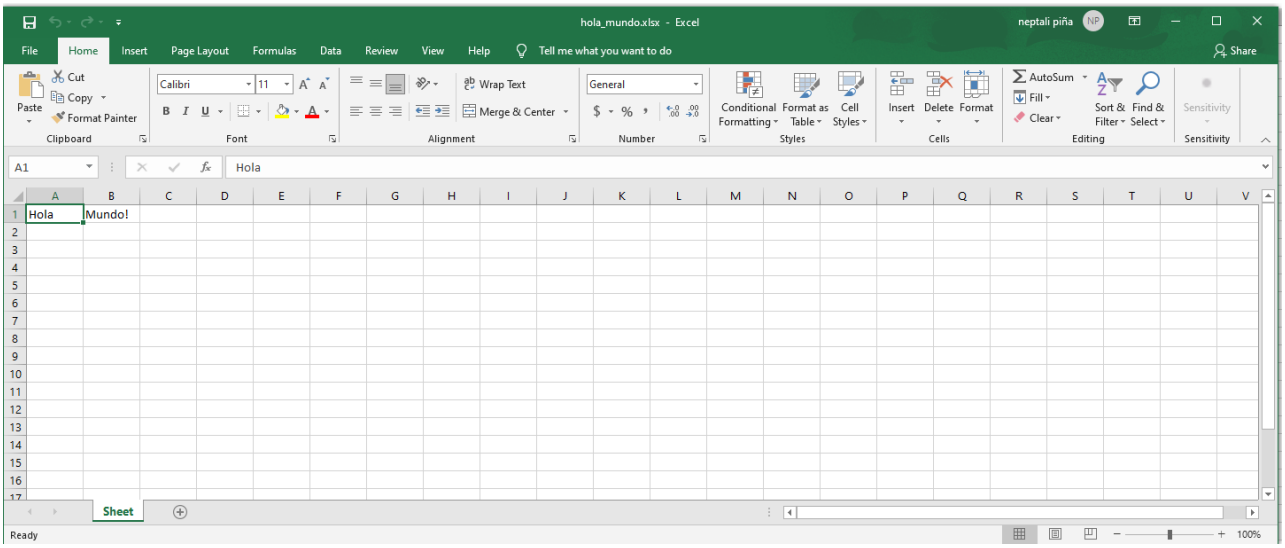
```
pip install openpyxl
```

Empezando con openpyxl

Después de instalar el paquete, debería poder crear una hoja de cálculo súper simple con el siguiente código:

```
1   from openpyxl import Workbook
2
3   workbook = Workbook()
4   sheet = workbook.active
5
6   sheet["A1"] = "Hola"
7   sheet["B1"] = "Mundo!"
8
9   workbook.save(filename="hola_mundo.xlsx")
10
```

El código anterior debería crear un archivo llamado `hola_mundo.xlsx` en la carpeta que está utilizando para ejecutar el código. Si abre ese archivo con Excel, debería ver algo como esto:



Agregar y actualizar valores de una celda

Para agregar el valor a una celda solo es necesario especificar la columna y la fila como se muestra a continuación:

```
6    sheet["A1"] = "Cambiando el valor"
```

Otra forma para acceder a la celda es especificando la columna por un número usando el .cell y pasando el valor con el atributo .value:

```
6    celda = sheet.cell(row=1, column=1)
7    celda.value = "Cambiando el valor"
```

Gestión de hojas

La gestión de hojas también es una de esas cosas que quizás necesite saber, aunque puede ser algo que no utilice con tanta frecuencia.

```
4    sheet = workbook.active
```

Esta es la forma de seleccionar la hoja predeterminada de una hoja de cálculo.

Si desea crear o eliminar hojas, también puede hacerlo con `.create_sheet()`y `.remove()`:

```
 9    ws_calculations = workbook.create_sheet('CALCULOS')
10    ws_budget = workbook.create_sheet('PRESUPUESTO')
```

```
12    workbook.remove(ws_calculations)
```

Combinar celdas

Una cosa que se requiere mucho es la posibilidad de mezclar celdas, esto nos permite agregar textos largos, para mezclar celdas se puede hacer de las siguientes formas:

```
6   sheet.merge_cells(start_row=1, start_column=1, end_row=1, end_column=9)
```

```
7   sheet.merge_cells('A1:I1')
```

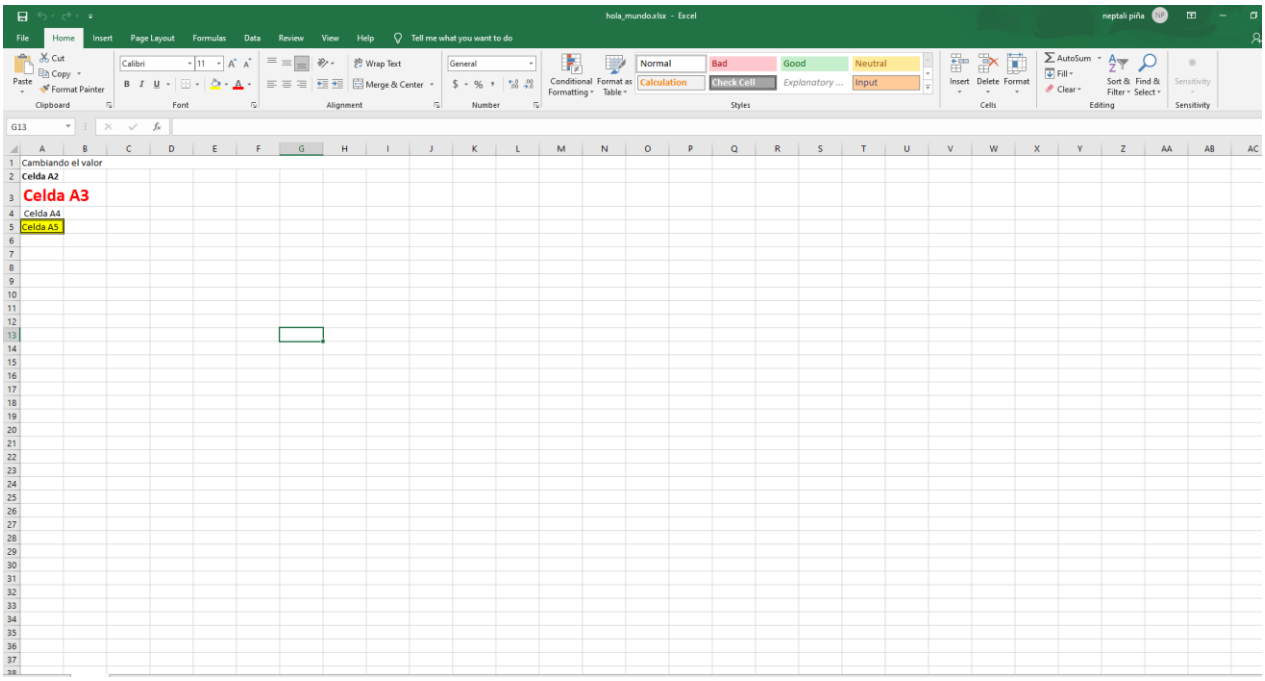
Agregar estilos

Aunque diseñar una hoja de cálculo puede no ser algo que haría todos los días, es bueno saber cómo hacerlo.

Con openpyxl, puede aplicar múltiples opciones de estilo a su hoja de cálculo, incluidas fuentes, bordes, colores, etc. Eche un vistazo a la openpyxl documentación para obtener más información.

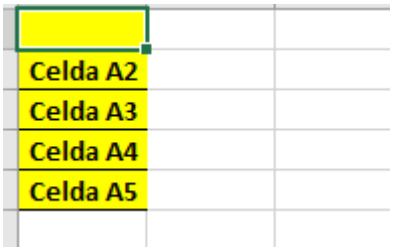
También puede optar por aplicar un estilo directamente a una celda o crear una plantilla y reutilizarla para aplicar estilos a varias celdas.

```
from openpyxl import Workbook
from openpyxl.styles import Font, PatternFill, Alignment, Border, Side

workbook = Workbook()
sheet = workbook.active
sheet.merge_cells('A1:I1')

celda = sheet.cell(row=1, column=1)
celda.value = "Cambiando el valor"


texto_bold = Font(bold=True)
texto_rojo_bold = Font(name="Calibri", size="20", bold=True, color="00FF0000", outline=True)
alinear_texto = Alignment(horizontal="center")
double_borde = Border(left=Side(border_style="double", color='000000'),
                      right=Side(border_style="double", color='000000'),
                      top=Side(border_style="double", color='000000'),
                      bottom=Side(border_style="double", color='000000'))
relleno = PatternFill(start_color='FFFF00', end_color='FFFF00', fill_type='solid')

sheet["A2"].font = texto_bold
sheet["A3"].font = texto_rojo_bold
sheet["A4"].alignment = alinear_texto
sheet["A5"].border = double_borde
sheet["A5"].fill = relleno

sheet["A2"].value = 'Celda A2'
sheet["A3"].value = 'Celda A3'
sheet["A4"].value = 'Celda A4'
sheet["A5"].value = 'Celda A5'
workbook.save(filename="hola_mundo.xlsx")
```



Cuando desee aplicar varios estilos a una o varias celdas, puede usar una `NamedStyle` clase en su lugar, que es como una plantilla de estilo que puede usar una y otra vez.

```
from openpyxl import Workbook
from openpyxl.styles import NamedStyle
from openpyxl.styles import Font, PatternFill, Alignment, Border, Side


workbook = Workbook()
sheet = workbook.active
header = NamedStyle(name="header")
header.font = Font(bold=True)
header.border = Border(bottom=Side(border_style="thin"))
header.alignment = Alignment(horizontal="center", vertical="center")
header.fill = PatternFill(start_color="FFFF00", end_color="FFFF00", fill_type="solid")

sheet["A2"].value = "Celda A2"
sheet["A3"].value = "Celda A3"
sheet["A4"].value = "Celda A4"
sheet["A5"].value = "Celda A5"

#for row in sheet.iter_rows(min_row=1, max_col=3, max_row=5):
for row in sheet["A1:A5"]:
    for cell in row:
        cell.style = header

workbook.save(filename="hola_mundo.xlsx")
```



Ejercicios propuestos:

1. Pasar los datos de camion.csv a un archivo .xlsx. El archivo debe tener la siguiente estructura:
    a. Título.
    b. Encabezados de cada columna.
    c. El listado de todas las frutas.
    d. El total de todos los precios de las frutas.
2. Crear un Excel con la siguiente estructura:

	A	B	C	D	E	F	G	H	I	J
1	CLIENTE:	GracoSoft					Código:	111		
2	Dirección:	Centro Empresarial Plaza Madrid								
3	Contacto:	Ana Mercedes Díaz		Teléfono:	4145227568		Correo:	gracosoft@gmail.com		
4	Producto:	Cursos								
5	Descripción:	Descripción								
6										
7										

Referencias bibliográficas

Textos Guía:

- Matthes, Eric. Python crash course: a hands-on, project-based introduction to programming / by Eric Matthes.
- Al Sweigart. Boring Stuff with Python: Practical Programming for total Beginners

Textos Complementarios:

- González Raúl D. Python para dodos https://www.utic.edu.py/citil/images/Manuales/Python_para_todos.pdf

Webs:

- https://realpython.com/openpyxl-excel-spreadsheets-python/
- https://openpyxl.readthedocs.io/en/stable/styles.html
- https://openpyxl.readthedocs.io/en/stable/index.html