

Graficar calificaciones alumnos



Tengo un archivo llamado "calificaciones_alumnos.xlsx", el cual tiene un campo llamado "Nombre" y varias columnas de calificaciones llamadas Mat_CalculoIntegral, Mat_ProgramacionOOP y Mat_EstructuraDatos. Ayúdame a crear un programa en Python para graficar las calificaciones para cada alumno pero no quiero que se encimen sus etiquetas en el eje X, al inicio del código agrega un comentario explicando lo que hace el programa y documenta las líneas importantes del código



Claro, aquí tienes un programa en Python que lee el archivo `calificaciones_alumnos.xlsx`, extrae las calificaciones de cada alumno y las grafica, asegurándose de que las etiquetas en el eje X no se encimen. El programa utiliza las bibliotecas `pandas` para manejar los datos y `matplotlib` para la creación de gráficos.

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" que contiene
# las calificaciones de los alumnos en distintas materias. Luego, crea un gráfico
# de barras para cada alumno mostrando sus calificaciones en las materias
# Mat_CalculoIntegral, Mat_ProgramacionOOP y Mat_EstructuraDatos.
# El programa se asegura de que las etiquetas en el eje X no se encimen usando un
# ajuste automático.

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Crear una lista de nombres de alumnos
alumnos = df['Nombre']

# Crear una matriz de calificaciones
calificaciones = df[['Mat_CalculoIntegral', 'Mat_ProgramacionOOP', 'Mat_EstructuraDatos']]

# Definir las etiquetas y la posición en el eje X
x = np.arange(len(alumnos)) # Posiciones en el eje X

# Configuración de la figura y los ejes
fig, ax = plt.subplots(figsize=(10, 6))

# Definir el ancho de las barras
width = 0.25

# Crear las barras para cada materia
rects1 = ax.bar(x - width, calificaciones['Mat_CalculoIntegral'], width, label='Cálculo Integral')
rects2 = ax.bar(x, calificaciones['Mat_ProgramacionOOP'], width, label='Programación OOP')
```

```
rects3 = ax.bar(x + width, calificaciones['Mat_EstructuraDatos'], width, label='Estructura
de Datos')

# Añadir etiquetas y título
ax.set_xlabel('Alumnos')
ax.set_ylabel('Calificaciones')
ax.set_title('Calificaciones de Alumnos por Materia')
ax.set_xticks(x)
ax.set_xticklabels(alumnos, rotation=45, ha='right')
ax.legend()

# Ajustar el espaciado para que las etiquetas no se encimen
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Explicación del código:

1. **Importar bibliotecas necesarias:** ``pandas`` para manejar los datos del archivo Excel, ``matplotlib.pyplot`` para graficar y ``numpy`` para manejar arrays.
2. **Leer el archivo Excel:** Se carga el archivo Excel en un DataFrame de ``pandas``.
3. **Crear listas de alumnos y calificaciones:** Se extraen los nombres de los alumnos y sus calificaciones en distintas materias.
4. **Configurar el gráfico:**
 - Se define la posición de cada alumno en el eje X.
 - Se crean las barras del gráfico para cada materia.
 - Se añaden etiquetas y título al gráfico.
 - Se rotan las etiquetas del eje X para que no se encimen.
5. **Ajustar el espaciado y mostrar el gráfico:** ``plt.tight_layout()`` ajusta el espaciado automáticamente y ``plt.show()`` muestra el gráfico.

Este programa debería permitirte visualizar las calificaciones de los alumnos de manera clara, sin que las etiquetas del eje X se encimen.



cómo instalo las librerías?



Para instalar las bibliotecas necesarias (``pandas``, ``matplotlib`` y ``numpy``), puedes usar ``pip``, que es el gestor de paquetes de Python. Abre tu terminal o consola de comandos y ejecuta los siguientes comandos:

1. **Instalar ``pandas``:**

```
pip install pandas
```

2. Instalar `matplotlib`:

sh

```
pip install matplotlib
```

3. Instalar `numpy`:

sh

```
pip install numpy
```

Estos comandos instalarán las bibliotecas necesarias en tu entorno de Python. Una vez instaladas, podrás ejecutar el programa proporcionado para graficar las calificaciones de los alumnos.



Me marca el siguiente error: "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\venv\Scripts\python.exe" "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\GraficaAlumnos.py"

Traceback (most recent call last):

File "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\.venv\Lib\site-packages\pandas\compat_optional.py", line 135, in import_optional_dependency
module = importlib.import_module(name)

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

File "C:\Users\dalij\AppData\Local\Programs\Python\Python312\Lib\importlib__init__.py", line 90, in import_module

```
return _bootstrap.qcd import(name[level:], package, level)
```

[illegible]

File "<frozen importlib._bootstrap>", line 1387, in _gcd_import

File "<frozen importlib._bootstrap>", line 1360, in _find_and_load

File "<frozen importlib._bootstrap>", line 1324, in _find_and_load_unlocked

```
ModuleNotFoundError: No module named 'openpyxl'
```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "C:\Users\dalij\Documents\Codigos\M2.1 Automatizando Excel\GraficaAlumnos.py", line 14,
in <module>

```
df = pd.read_excel(archivo)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\.venv\Lib\site-
packages\pandas\io\excel\_base.py", line 495, in read_excel
io = ExcelFile(
^^^^^^^^^^^^^^

File "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\.venv\Lib\site-
packages\pandas\io\excel\_base.py", line 1567, in __init__
self._reader = self._engines[engine](
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\.venv\Lib\site-
packages\pandas\io\excel\_openpyxl.py", line 552, in __init__
import_optional_dependency("openpyxl")
File "C:\Users\dalij\Documents\Codigos\ M2.1 Automatizando Excel\.venv\Lib\site-
packages\pandas\compat\_optional.py", line 138, in import_optional_dependency
raise ImportError(msg)
ImportError: Missing optional dependency 'openpyxl'. Use pip or conda to install openpyxl.

Process finished with exit code 1
```



El error indica que falta la biblioteca `openpyxl`, que es necesaria para que `pandas` pueda leer archivos Excel. Puedes instalar `openpyxl` usando `pip`. Abre tu terminal o consola de comandos y ejecuta el siguiente comando:

```
sh

pip install openpyxl
```

Esto instalará la biblioteca `openpyxl` y debería resolver el error que estás viendo. Después de instalar `openpyxl`, intenta ejecutar tu script de nuevo.

Si estás utilizando un entorno virtual (`.venv`), asegúrate de que el entorno virtual esté activado cuando instales la biblioteca y cuando ejecutes tu script.

Pasos para activar el entorno virtual y ejecutar el script:

1. Activar el entorno virtual:

- En Windows:

```
sh

.venv\Scripts\activate
```

- En macOS/Linux:

```
sh
```

```
source .venv/bin/activate
```

2. Instalar `openpyxl` en el entorno virtual:

```
sh
```

```
pip install openpyxl
```

3. Ejecutar tu script:

```
sh
```

```
python GraficaAlumnos.py
```

Con estos pasos, deberías poder ejecutar tu script sin problemas.



Tengo un archivo llamado "calificaciones_alumnos.xlsx", el cual tiene un campo llamado "Nombre" y varias columnas de calificaciones llamadas Mat_CalculoIntegral, Mat_ProgramacionOOP y Mat_EstructuraDatos. Ayúdame a crear un programa en Python para agregar una columna llamada Mat_Fisica que contenga valores aleatorios entre 0 y 100 con un decimal, al inicio del código agrega un comentario explicando lo que hace el programa y documenta las líneas importantes del código



Claro, aquí tienes un programa en Python que lee el archivo `calificaciones_alumnos.xlsx`, agrega una columna llamada `Mat_Fisica` con valores aleatorios entre 0 y 100 con un decimal, y guarda el archivo modificado. El programa utiliza las bibliotecas `pandas` y `numpy`.

```
python
```

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" que contiene
# las calificaciones de los alumnos en distintas materias. Luego, agrega una columna
# llamada "Mat_Fisica" con valores aleatorios entre 0 y 100 con un decimal y guarda
# el archivo modificado.
```

```
import pandas as pd
import numpy as np
```

```
# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Generar valores aleatorios entre 0 y 100 con un decimal para la nueva columna
"Mat_Fisica"
df['Mat_Fisica'] = np.round(np.random.uniform(0, 100, df.shape[0]), 1)

# Guardar el DataFrame modificado en un nuevo archivo Excel
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)

print("Se ha agregado la columna 'Mat_Fisica' con valores aleatorios y se ha guardado el
archivo modificado.")
```

Explicación del código:

1. **Importar bibliotecas necesarias:** `pandas` para manejar los datos del archivo Excel y `numpy` para generar valores aleatorios.
2. **Leer el archivo Excel:** Se carga el archivo Excel en un DataFrame de `pandas`.
3. **Generar valores aleatorios para la columna `Mat_Fisica`:** Se utilizan las funciones `np.random.uniform` para generar valores aleatorios entre 0 y 100 y `np.round` para redondear a un decimal.
4. **Agregar la columna `Mat_Fisica` al DataFrame:** Se añade la nueva columna al DataFrame.
5. **Guardar el DataFrame modificado en un nuevo archivo Excel:** Se guarda el DataFrame con la nueva columna en un archivo llamado `calificaciones_alumnos_modificado.xlsx`.
6. **Imprimir un mensaje de confirmación:** Se imprime un mensaje indicando que se ha agregado la columna y guardado el archivo modificado.

Instalación de las bibliotecas necesarias:

Si aún no has instalado `pandas` y `numpy`, puedes hacerlo usando los siguientes comandos en la terminal o consola de comandos:

```
sh
```

```
pip install pandas
pip install numpy
```

Una vez instaladas, podrás ejecutar el programa para agregar la nueva columna con valores aleatorios.



Para el nuevo archivo generado quiero que me ordene la tabla por medio del Nombre, al inicio del código agrega un comentario explicando lo que hace el programa y documenta las líneas importantes del código



Claro, aquí tienes un programa en Python que lee el archivo `calificaciones_alumnos.xlsx`, agrega una columna llamada `Mat_Fisica` con valores aleatorios entre 0 y 100 con un decimal, ordena la tabla por el campo `Nombre`, y guarda el archivo modificado. El programa utiliza las bibliotecas `pandas` y `numpy`.

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" que contiene
# las calificaciones de los alumnos en distintas materias. Luego, agrega una columna
# llamada "Mat_Fisica" con valores aleatorios entre 0 y 100 con un decimal, ordena
# la tabla por el campo "Nombre" y guarda el archivo modificado.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Generar valores aleatorios entre 0 y 100 con un decimal para la nueva columna
"Mat_Fisica"
df['Mat_Fisica'] = np.round(np.random.uniform(0, 100, df.shape[0]), 1)

# Ordenar el DataFrame por la columna "Nombre"
df = df.sort_values(by='Nombre')

# Guardar el DataFrame modificado en un nuevo archivo Excel
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)

print("Se ha agregado la columna 'Mat_Fisica' con valores aleatorios, se ha ordenado por
'Nombre' y se ha guardado el archivo modificado.")
```

Explicación del código:

1. **Importar bibliotecas necesarias:** `pandas` para manejar los datos del archivo Excel y `numpy` para generar valores aleatorios.
2. **Leer el archivo Excel:** Se carga el archivo Excel en un DataFrame de `pandas`.
3. **Generar valores aleatorios para la columna `Mat_Fisica`:** Se utilizan las funciones `np.random.uniform` para generar valores aleatorios entre 0 y 100 y `np.round` para redondear a un decimal.
4. **Agregar la columna `Mat_Fisica` al DataFrame:** Se añade la nueva columna al DataFrame.
5. **Ordenar el DataFrame por la columna `Nombre`:** Se utiliza `sort_values` para ordenar el DataFrame por la columna `Nombre`.
6. **Guardar el DataFrame modificado en un nuevo archivo Excel:** Se guarda el DataFrame con la nueva columna y ordenado en un archivo llamado `calificaciones_alumnos_modificado.xlsx`.
7. **Imprimir un mensaje de confirmación:** Se imprime un mensaje indicando que se ha agregado la columna, ordenado la tabla y guardado el archivo modificado.

Instalación de las bibliotecas necesarias:

Si aún no has instalado `pandas` y `numpy`, puedes hacerlo usando los siguientes comandos en la terminal o consola de comandos:

```
sh

pip install pandas
pip install numpy
```

Una vez instaladas, podrás ejecutar el programa para agregar la nueva columna con valores aleatorios, ordenar la tabla por `Nombre` y guardar el archivo modificado.



Quiero saber cuántos registros y cuántos campos tiene mi tabla, al inicio del código agrega un comentario explicando lo que hace el programa y documenta las líneas importantes del código



Claro, aquí tienes el programa actualizado que, además de agregar la columna `Mat_Fisica` con valores aleatorios y ordenar la tabla por el campo `Nombre`, también imprime la cantidad de registros y campos en la tabla.

```
python

# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" que contiene
# las calificaciones de los alumnos en distintas materias. Luego, agrega una columna
# llamada "Mat_Fisica" con valores aleatorios entre 0 y 100 con un decimal, ordena
# la tabla por el campo "Nombre", imprime la cantidad de registros y campos en la
# tabla y guarda el archivo modificado.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Obtener la cantidad de registros y campos
num_registros = df.shape[0]
num_campos = df.shape[1]

# Imprimir la cantidad de registros y campos
print(f"La tabla tiene {num_registros} registros y {num_campos} campos.")

# Generar valores aleatorios entre 0 y 100 con un decimal para la nueva columna
"Mat_Fisica"
df['Mat_Fisica'] = np.round(np.random.uniform(0, 100, df.shape[0]), 1)

# Ordenar el DataFrame por la columna "Nombre"
df = df.sort_values(by='Nombre')

# Guardar el DataFrame modificado en un nuevo archivo Excel
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)
```



```
print("Se ha agregado la columna 'Mat_Fisica' con valores aleatorios, se ha ordenado por 'Nombre' y se ha guardado el archivo modificado.")
```

Explicación del código:

1. **Importar bibliotecas necesarias:** `pandas` para manejar los datos del archivo Excel y `numpy` para generar valores aleatorios.
2. **Leer el archivo Excel:** Se carga el archivo Excel en un DataFrame de `pandas`.
3. **Obtener la cantidad de registros y campos:**
 - `num_registros = df.shape[0]`: Obtiene el número de filas (registros) en el DataFrame.
 - `num_campos = df.shape[1]`: Obtiene el número de columnas (campos) en el DataFrame.
4. **Imprimir la cantidad de registros y campos:** Se imprime la cantidad de registros y campos.
5. **Generar valores aleatorios para la columna `Mat_Fisica`:** Se utilizan las funciones `np.random.uniform` para generar valores aleatorios entre 0 y 100 y `np.round` para redondear a un decimal.
6. **Agregar la columna `Mat_Fisica` al DataFrame:** Se añade la nueva columna al DataFrame.
7. **Ordenar el DataFrame por la columna `Nombre`:** Se utiliza `sort_values` para ordenar el DataFrame por la columna `Nombre`.
8. **Guardar el DataFrame modificado en un nuevo archivo Excel:** Se guarda el DataFrame con la nueva columna y ordenado en un archivo llamado `calificaciones_alumnos_modificado.xlsx`.
9. **Imprimir un mensaje de confirmación:** Se imprime un mensaje indicando que se ha agregado la columna, ordenado la tabla y guardado el archivo modificado.

Instalación de las bibliotecas necesarias:

Si aún no has instalado `pandas` y `numpy`, puedes hacerlo usando los siguientes comandos en la terminal o consola de comandos:

```
sh
```

```
pip install pandas
pip install numpy
```

Una vez instaladas, podrás ejecutar el programa para agregar la nueva columna con valores aleatorios, ordenar la tabla por `Nombre`, imprimir la cantidad de registros y campos, y guardar el archivo modificado.



Quiero saber qué campos son numéricos, al inicio del código agrega un comentario explicando lo que hace el programa y documenta las líneas importantes del código



Claro, aquí tienes el programa actualizado que, además de agregar la columna `Mat_Fisica` con valores aleatorios, ordenar la tabla por el campo `Nombre`, y contar registros y campos, también identifica cuáles campos son numéricos.

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" que contiene
# las calificaciones de los alumnos en distintas materias. Luego, agrega una columna
# llamada "Mat_Fisica" con valores aleatorios entre 0 y 100 con un decimal, ordena
# la tabla por el campo "Nombre", imprime la cantidad de registros y campos en la
# tabla, identifica los campos numéricos y guarda el archivo modificado.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Obtener la cantidad de registros y campos
num_registros = df.shape[0]
num_campos = df.shape[1]

# Imprimir la cantidad de registros y campos
print(f"La tabla tiene {num_registros} registros y {num_campos} campos.")

# Identificar los campos numéricos
campos_numericos = df.select_dtypes(include=[np.number]).columns.tolist()

# Imprimir los campos numéricos
print(f"Los campos numéricos son: {'', '.join(campos_numericos)}")

# Generar valores aleatorios entre 0 y 100 con un decimal para la nueva columna
"Mat_Fisica"
df['Mat_Fisica'] = np.round(np.random.uniform(0, 100, df.shape[0]), 1)

# Ordenar el DataFrame por la columna "Nombre"
df = df.sort_values(by='Nombre')

# Guardar el DataFrame modificado en un nuevo archivo Excel
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)

print("Se ha agregado la columna 'Mat_Fisica' con valores aleatorios, se ha ordenado por
'Nombre' y se ha guardado el archivo modificado.")
```

Explicación del código:

- Importar bibliotecas necesarias:** `pandas` para manejar los datos del archivo Excel y `numpy` para generar valores aleatorios.
- Leer el archivo Excel:** Se carga el archivo Excel en un DataFrame de `pandas`.
- Obtener la cantidad de registros y campos:**
 - `num_registros = df.shape[0]`: Obtiene el número de filas (registros) en el DataFrame.
 - `num_campos = df.shape[1]`: Obtiene el número de columnas (campos) en el DataFrame.
- Imprimir la cantidad de registros y campos:** Se imprime la cantidad de registros y campos.
- Identificar los campos numéricos:**

- `campos_numericos = df.select_dtypes(include=[np.number]).columns.tolist()`: Obtiene una lista de los nombres de las columnas que contienen datos numéricos.
- 6. **Imprimir los campos numéricos:** Se imprime la lista de campos numéricos.
- 7. **Generar valores aleatorios para la columna `Mat_Fisica`:** Se utilizan las funciones `np.random.uniform` para generar valores aleatorios entre 0 y 100 y `np.round` para redondear a un decimal.
- 8. **Agregar la columna `Mat_Fisica` al DataFrame:** Se añade la nueva columna al DataFrame.
- 9. **Ordenar el DataFrame por la columna `Nombre`:** Se utiliza `sort_values` para ordenar el DataFrame por la columna `Nombre`.
- 10. **Guardar el DataFrame modificado en un nuevo archivo Excel:** Se guarda el DataFrame con la nueva columna y ordenado en un archivo llamado `calificaciones_alumnos_modificado.xlsx`.
- 11. **Imprimir un mensaje de confirmación:** Se imprime un mensaje indicando que se ha agregado la columna, ordenado la tabla y guardado el archivo modificado.

Instalación de las bibliotecas necesarias:

Si aún no has instalado `pandas` y `numpy`, puedes hacerlo usando los siguientes comandos en la terminal o consola de comandos:

```
sh

pip install pandas
pip install numpy
```

Una vez instaladas, podrás ejecutar el programa para agregar la nueva columna con valores aleatorios, ordenar la tabla por `Nombre`, imprimir la cantidad de registros y campos, identificar los campos numéricos, y guardar el archivo modificado.



Genera 15 prompt detallados para automatizar procesos y obtener respuestas significativas al documento de Excel. Para cada prompt solicita crea un archivo `.py` con la documentación interna correspondiente



Aquí tienes 15 prompts detallados para automatizar diferentes procesos con un archivo de Excel y obtener respuestas significativas. Para cada prompt, te proporciono un archivo `.py` con la documentación interna correspondiente.

Prompt 1: Leer un archivo Excel y mostrar sus primeras 5 filas

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" y muestra
# las primeras 5 filas del DataFrame resultante.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Mostrar las primeras 5 filas
print(df.head())
```

Prompt 2: Agregar una columna con valores aleatorios y guardar el archivo

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", agrega una
# columna llamada "Mat_Fisica" con valores aleatorios entre 0 y 100 con un decimal, y
# guarda el archivo modificado.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Generar valores aleatorios para la columna "Mat_Fisica"
df['Mat_Fisica'] = np.round(np.random.uniform(0, 100, df.shape[0]), 1)

# Guardar el archivo modificado
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)

print("Se ha agregado la columna 'Mat_Fisica' con valores aleatorios y se ha guardado el
archivo modificado.")
```

Prompt 3: Ordenar el DataFrame por una columna específica

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", ordena
# el DataFrame por la columna "Nombre" y guarda el archivo modificado.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Ordenar el DataFrame por la columna "Nombre"
df = df.sort_values(by='Nombre')

# Guardar el archivo modificado
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)
```

```
print("Se ha ordenado el archivo por 'Nombre' y se ha guardado el archivo modificado.")
```

Prompt 4: Contar la cantidad de registros y campos en el DataFrame

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" y cuenta
# la cantidad de registros y campos en el DataFrame resultante.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Contar registros y campos
num_registros = df.shape[0]
num_campos = df.shape[1]

# Imprimir la cantidad de registros y campos
print(f"La tabla tiene {num_registros} registros y {num_campos} campos.")
```

Prompt 5: Identificar los campos numéricos en el DataFrame

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" e identifica
# los campos numéricos en el DataFrame resultante.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Identificar campos numéricos
campos_numericos = df.select_dtypes(include=[np.number]).columns.tolist()

# Imprimir los campos numéricos
print(f"Los campos numéricos son: {'', '.join(campos_numericos)}")
```

Prompt 6: Calcular estadísticas básicas (media, mediana, etc.) de los campos numéricos

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" y calcula
# estadísticas básicas de los campos numéricos en el DataFrame resultante.

import pandas as pd
```

```
# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Calcular estadísticas básicas
estadisticas = df.describe()

# Imprimir estadísticas básicas
print(estadisticas)
```

Prompt 7: Filtrar filas con una condición específica

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" y filtra
# las filas donde la calificación en "Mat_CalculoIntegral" es mayor a 70.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Filtrar filas con calificaciones mayores a 70 en "Mat_CalculoIntegral"
filtrado = df[df['Mat_CalculoIntegral'] > 70]

# Imprimir las filas filtradas
print(filtrado)
```

Prompt 8: Crear un gráfico de barras con las calificaciones de una materia

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx" y crea un
# gráfico de barras con las calificaciones en "Mat_ProgramacionOOP".

import pandas as pd
import matplotlib.pyplot as plt

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Crear gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(df['Nombre'], df['Mat_ProgramacionOOP'])
plt.xlabel('Nombre')
plt.ylabel('Calificación Programación OOP')
plt.title('Calificaciones en Programación OOP')
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Prompt 9: Crear un archivo Excel solo con las columnas numéricas

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", extrae
# solo las columnas numéricas y guarda un nuevo archivo Excel con esos datos.

import pandas as pd
import numpy as np

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Extraer columnas numéricas
df_numerico = df.select_dtypes(include=[np.number])

# Guardar el nuevo archivo Excel
df_numerico.to_excel('calificaciones_numericas.xlsx', index=False)

print("Se ha creado un nuevo archivo Excel con solo las columnas numéricas.")
```

Prompt 10: Agrupar por una columna y calcular la media de otra columna

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", agrupa
# el DataFrame por la columna "Nombre" y calcula la media de "Mat_EstructuraDatos".

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Agrupar por "Nombre" y calcular la media de "Mat_EstructuraDatos"
media_estructura_datos = df.groupby('Nombre')['Mat_EstructuraDatos'].mean()

# Imprimir la media de "Mat_EstructuraDatos" por "Nombre"
print(media_estructura_datos)
```

Prompt 11: Crear una nueva columna basada en condiciones de otra columna

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", crea una
# nueva columna "Estado_Calculo" basada en las calificaciones de "Mat_CalculoIntegral"
# y guarda el archivo modificado.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)
```

```
# Crear nueva columna basada en condiciones
df['Estado_Calculo'] = df['Mat_CalculoIntegral'].apply(lambda x: 'Aprobado' if x >= 60
else 'Reprobado')

# Guardar el archivo modificado
df.to_excel('calificaciones_alumnos_modificado.xlsx', index=False)

print("Se ha agregado la columna 'Estado_Calculo' y se ha guardado el archivo
modificado.")
```

Prompt 12: Fusionar dos archivos Excel por una columna común

python

```
# Este programa lee dos archivos Excel llamados "calificaciones_alumnos.xlsx" y
# "informacion_extra.xlsx", los fusiona por la columna "Nombre" y guarda el archivo
resultante.

import pandas as pd

# Leer los archivos Excel
archivo1 = 'calificaciones_alumnos.xlsx'
archivo2 = 'informacion_extra.xlsx'
df1 = pd.read_excel(archivo1)
df2 = pd.read_excel(archivo2)

# Fusionar los DataFrames por la columna "Nombre"
df_merged = pd.merge(df1, df2, on='Nombre')

# Guardar el archivo resultante
df_merged.to_excel('calificaciones_alumnos_fusionado.xlsx', index=False)

print("Se han fusionado los archivos y se ha guardado el archivo resultante.")
```

Prompt 13: Detectar y eliminar duplicados en el DataFrame

python

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", detecta
# y elimina filas duplicadas en el DataFrame y guarda el archivo modificado.

import pandas as pd

# Leer el archivo Excel
archivo = 'calificaciones_alumnos.xlsx'
df = pd.read_excel(archivo)

# Detectar y eliminar duplicados
df_sin_duplicados = df.drop_duplicates()

# Guardar el archivo modificado
df_sin_duplicados.to_excel('calificaciones_alumnos_sin_duplicados.xlsx', index=False)

print("Se han eliminado las filas duplicadas y se ha guardado el archivo modificado.")
```


Prompt 14: Imputar valores faltantes en el DataFrame

```
python
```

```
# Este programa lee un archivo Excel llamado "calificaciones_alumnos.xlsx", detecta  
# valores faltantes en el DataFrame,
```



puedes indicar por cada uno la descripción detallada de los 15 prompts para que genere esos ejemplos



Por supuesto, aquí tienes las descripciones detalladas para cada uno de los 15 prompts para automatizar procesos con un archivo Excel y obtener respuestas significativas:

Prompt 1: Leer un archivo Excel y mostrar sus primeras 5 filas

Descripción: Este script lee el archivo Excel llamado `calificaciones_alumnos.xlsx` y muestra las primeras 5 filas del DataFrame resultante. Esto es útil para obtener una vista rápida del contenido del archivo y verificar que se ha cargado correctamente.

Prompt 2: Agregar una columna con valores aleatorios y guardar el archivo

Descripción: Este script agrega una nueva columna llamada `Mat_Fisica` al archivo Excel `calificaciones_alumnos.xlsx`. La columna contiene valores aleatorios entre 0 y 100 con un decimal. Luego guarda el DataFrame modificado en un nuevo archivo llamado `calificaciones_alumnos_modificado.xlsx`.

Prompt 3: Ordenar el DataFrame por una columna específica

Descripción: Este script ordena el DataFrame del archivo Excel `calificaciones_alumnos.xlsx` por la columna `Nombre`. El resultado se guarda en un nuevo archivo llamado `calificaciones_alumnos_modificado.xlsx`. Esto facilita la visualización y el análisis de los datos en un orden específico.

Prompt 4: Contar la cantidad de registros y campos en el DataFrame

Descripción: Este script lee el archivo Excel `calificaciones_alumnos.xlsx` y cuenta el número de registros (filas) y campos (columnas) en el DataFrame. Imprime esta información para ayudar a comprender la estructura del archivo.

Prompt 5: Identificar los campos numéricos en el DataFrame

Descripción: Este script identifica y lista los campos numéricos en el archivo Excel ``calificaciones_alumnos.xlsx``. Es útil para entender qué columnas contienen datos numéricos que se pueden analizar matemáticamente.

Prompt 6: Calcular estadísticas básicas (media, mediana, etc.) de los campos numéricos

Descripción: Este script calcula estadísticas básicas como la media, mediana, desviación estándar, etc., para todas las columnas numéricas del archivo Excel ``calificaciones_alumnos.xlsx``. Imprime estas estadísticas para proporcionar un resumen de los datos numéricos.

Prompt 7: Filtrar filas con una condición específica

Descripción: Este script filtra las filas del archivo Excel ``calificaciones_alumnos.xlsx`` donde la calificación en ``Mat_CalculoIntegral`` es mayor a 70. El resultado se muestra para analizar sólo las filas que cumplen con esta condición.

Prompt 8: Crear un gráfico de barras con las calificaciones de una materia

Descripción: Este script crea un gráfico de barras que muestra las calificaciones en ``Mat_ProgramacionOOP`` del archivo Excel ``calificaciones_alumnos.xlsx``. El gráfico ayuda a visualizar las calificaciones por alumno y facilita el análisis visual de los datos.

Prompt 9: Crear un archivo Excel solo con las columnas numéricas

Descripción: Este script extrae únicamente las columnas numéricas del archivo Excel ``calificaciones_alumnos.xlsx`` y guarda un nuevo archivo llamado ``calificaciones_numericas.xlsx`` con esos datos. Es útil para trabajar exclusivamente con datos numéricos.

Prompt 10: Agrupar por una columna y calcular la media de otra columna

Descripción: Este script agrupa los datos del archivo Excel ``calificaciones_alumnos.xlsx`` por la columna ``Nombre`` y calcula la media de las calificaciones en ``Mat_EstructuraDatos``. Imprime el resultado para ver el promedio de calificaciones por alumno.

Prompt 11: Crear una nueva columna basada en condiciones de otra columna

Descripción: Este script crea una nueva columna llamada ``Estado_Calculo`` en el archivo Excel ``calificaciones_alumnos.xlsx``. Esta columna se basa en las calificaciones de ``Mat_CalculoIntegral``, indicando si el alumno ha aprobado o reprobado. El archivo modificado se guarda para análisis adicional.

Prompt 12: Fusionar dos archivos Excel por una columna común

Descripción: Este script fusiona dos archivos Excel, `calificaciones_alumnos.xlsx` e `informacion_extra.xlsx`, basándose en la columna común `Nombre`. El archivo resultante se guarda como `calificaciones_alumnos_fusionado.xlsx`. Esto permite combinar información de diferentes fuentes.

Prompt 13: Detectar y eliminar duplicados en el DataFrame

Descripción: Este script detecta y elimina filas duplicadas en el archivo Excel `calificaciones_alumnos.xlsx`. El DataFrame resultante se guarda en un nuevo archivo llamado `calificaciones_alumnos_sin_duplicados.xlsx`, facilitando la limpieza de datos.

Prompt 14: Imputar valores faltantes en el DataFrame

Descripción: Este script lee el archivo Excel `calificaciones_alumnos.xlsx`, detecta valores faltantes en el DataFrame y los imputa con un valor predeterminado (por ejemplo, la media de la columna). El archivo modificado se guarda para asegurar que todos los datos estén completos.

Prompt 15: Crear un resumen de los datos de cada columna

Descripción: Este script genera un resumen descriptivo de cada columna en el archivo Excel `calificaciones_alumnos.xlsx`. El resumen incluye estadísticas como la media, mediana, valor mínimo y máximo para columnas numéricas, y el conteo de valores únicos para columnas categóricas. El resumen se guarda en un archivo Excel llamado `resumen_datos.xlsx`.

Cada uno de estos scripts realiza una tarea específica relacionada con el procesamiento de datos en archivos Excel y está documentado para facilitar su comprensión y uso.



Los puedes redactar como si yo como usuario estuviera redactando los prompt



¡Claro! Aquí tienes 15 prompts redactados desde la perspectiva de un usuario que está buscando automatizar procesos y obtener respuestas significativas a partir de un archivo Excel:

Prompt 1: Leer un archivo Excel y mostrar sus primeras 5 filas

Descripción: Necesito un script en Python que lea el archivo `calificaciones_alumnos.xlsx` y muestre las primeras 5 filas del DataFrame resultante para verificar el contenido inicial del archivo.

Prompt 2: Agregar una columna con valores aleatorios y guardar el archivo

Descripción: Quiero un script que agregue una nueva columna llamada `Mat_Fisica` al archivo `calificaciones_alumnos.xlsx`. Esta columna debe contener valores aleatorios entre 0 y 100 con un

decimal. Luego, el archivo modificado debe guardarse como

`calificaciones_alumnos_modificado.xlsx`.

Prompt 3: Ordenar el DataFrame por una columna específica

Descripción: Necesito un script que ordene el archivo Excel `calificaciones_alumnos.xlsx` por la columna `Nombre`. El DataFrame ordenado debe guardarse en un nuevo archivo llamado `calificaciones_alumnos_modificado.xlsx`.

Prompt 4: Contar la cantidad de registros y campos en el DataFrame

Descripción: Quiero un script que lea el archivo Excel `calificaciones_alumnos.xlsx` y cuente cuántos registros (filas) y cuántos campos (columnas) tiene el DataFrame. El script debe imprimir esta información.

Prompt 5: Identificar los campos numéricos en el DataFrame

Descripción: Necesito un script que identifique y liste todos los campos numéricos en el archivo Excel `calificaciones_alumnos.xlsx`. El script debe imprimir los nombres de las columnas que contienen datos numéricos.

Prompt 6: Calcular estadísticas básicas (media, mediana, etc.) de los campos numéricos

Descripción: Quiero un script que lea el archivo Excel `calificaciones_alumnos.xlsx` y calcule estadísticas básicas (media, mediana, desviación estándar, etc.) para todas las columnas numéricas. El script debe imprimir un resumen estadístico de los datos numéricos.

Prompt 7: Filtrar filas con una condición específica

Descripción: Necesito un script que filtre las filas del archivo Excel `calificaciones_alumnos.xlsx` para que muestre solo aquellas en las que la calificación en `Mat_CalculoIntegral` sea mayor a 70. El script debe imprimir las filas que cumplen con esta condición.

Prompt 8: Crear un gráfico de barras con las calificaciones de una materia

Descripción: Quiero un script que cree un gráfico de barras que muestre las calificaciones en `Mat_ProgramacionOOP` del archivo Excel `calificaciones_alumnos.xlsx`. El gráfico debe tener las etiquetas de los alumnos en el eje X y las calificaciones en el eje Y.

Prompt 9: Crear un archivo Excel solo con las columnas numéricas

Descripción: Necesito un script que extraiga solo las columnas numéricas del archivo Excel `calificaciones_alumnos.xlsx` y guarde estos datos en un nuevo archivo llamado `calificaciones_numericas.xlsx`.

Prompt 10: Agrupar por una columna y calcular la media de otra columna

Descripción: Quiero un script que agrupe los datos del archivo Excel `calificaciones_alumnos.xlsx` por la columna `Nombre` y calcule la media de las calificaciones en `Mat_EstructuraDatos`. El script debe imprimir la media de `Mat_EstructuraDatos` para cada `Nombre`.

Prompt 11: Crear una nueva columna basada en condiciones de otra columna

Descripción: Necesito un script que cree una nueva columna llamada `Estado_Calculo` en el archivo Excel `calificaciones_alumnos.xlsx`. Esta columna debe basarse en las calificaciones de `Mat_CalculoIntegral`, indicando si el alumno ha aprobado o reprobado. El archivo modificado debe guardarse con la nueva columna.

Prompt 12: Fusionar dos archivos Excel por una columna común

Descripción: Quiero un script que lea dos archivos Excel, `calificaciones_alumnos.xlsx` e `informacion_extra.xlsx`, los fusione por la columna común `Nombre` y guarde el resultado en un nuevo archivo llamado `calificaciones_alumnos_fusionado.xlsx`.

Prompt 13: Detectar y eliminar duplicados en el DataFrame

Descripción: Necesito un script que lea el archivo Excel `calificaciones_alumnos.xlsx`, detecte y elimine las filas duplicadas en el DataFrame. El archivo sin duplicados debe guardarse como `calificaciones_alumnos_sin_duplicados.xlsx`.

Prompt 14: Imputar valores faltantes en el DataFrame

Descripción: Quiero un script que lea el archivo Excel `calificaciones_alumnos.xlsx`, detecte valores faltantes en el DataFrame y los impute con un valor predeterminado (por ejemplo, la media de la columna). El archivo modificado debe guardarse con los valores imputados.

Prompt 15: Crear un resumen de los datos de cada columna

Descripción: Necesito un script que genere un resumen descriptivo de cada columna en el archivo Excel `calificaciones_alumnos.xlsx`. El resumen debe incluir estadísticas como la media, mediana, valor mínimo y máximo para columnas numéricas, y el conteo de valores únicos para columnas categóricas. El resumen debe guardarse en un archivo Excel llamado `resumen_datos.xlsx`. Estos prompts están redactados para guiarte en la creación de scripts que automatizan diversos procesos relacionados con el manejo de archivos Excel.

ChatGPT puede cometer errores. Comprueba la información importante.