

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 hace lo que pides. Utilizaremos `pandas` para leer el archivo Excel y `matplotlib` para graficar los datos.

python

```
# Este programa lee un archivo Excel que contiene las calificaciones de varios alumnos en
# diferentes materias.
# Luego, grafica las calificaciones para cada alumno, asegurando que las etiquetas de los
# alumnos en el eje X no se encimen.

import pandas as pd
import matplotlib.pyplot as plt

# Leer el archivo Excel y cargar los datos en un DataFrame de pandas
df = pd.read_excel('calificaciones_alumnos.xlsx')

# Crear una figura y un conjunto de subgráficos para manejar los gráficos
plt.figure(figsize=(10, 6))

# Para cada alumno, graficar sus calificaciones
for i, row in df.iterrows():
    plt.plot(['Cálculo Integral', 'Programación OOP', 'Estructura de Datos'],
             [row['Mat_CalculoIntegral'], row['Mat_ProgramacionOOP'],
              row['Mat_EstructuraDatos']],
             marker='o', label=row['Nombre'])

# Agregar un título al gráfico
plt.title('Calificaciones de los Alumnos por Materia')

# Rotar las etiquetas del eje X para que no se encimen
plt.xticks(rotation=45)

# Agregar etiquetas a los ejes
plt.xlabel('Materias')
plt.ylabel('Calificaciones')

# Agregar una leyenda con los nombres de los alumnos
plt.legend(title="Alumnos", bbox_to_anchor=(1.05, 1), loc='upper left')

# Ajustar el gráfico para que se vea mejor
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Explicación del código:

1. **Lectura del archivo Excel:** Se utiliza ``pandas`` para leer los datos desde un archivo Excel y cargarlos en un DataFrame.
2. **Creación del gráfico:** Utilizamos un bucle ``for`` para recorrer cada fila del DataFrame y graficar las calificaciones de cada alumno.
3. **Rotación de etiquetas en el eje X:** La función ``plt.xticks(rotation=45)`` asegura que las etiquetas de las materias en el eje X no se encimen, rotándolas 45 grados.
4. **Leyenda:** Se añade una leyenda fuera del gráfico para indicar a qué alumno pertenece cada línea.
5. **Ajuste del layout:** ``plt.tight_layout()`` se usa para ajustar los elementos del gráfico y evitar que se superpongan.

Este código debería ayudarte a visualizar las calificaciones de los alumnos de manera clara y ordenada.



como soluciono este error Could not find a version that satisfies the requirement pyplot (from versions: none)



El error que mencionas ocurre porque ``pyplot`` no es un paquete independiente que se pueda instalar directamente, sino que es un módulo dentro del paquete ``matplotlib``. Para solucionarlo, asegúrate de tener ``matplotlib`` instalado. Puedes hacerlo ejecutando el siguiente comando en tu entorno de Python:

```
bash
```

```
pip install matplotlib
```

Una vez que hayas instalado ``matplotlib``, puedes importar ``pyplot`` en tu código de esta manera:

```
python
```

```
import matplotlib.pyplot as plt
```

Esto debería resolver el problema y permitirte ejecutar el código sin errores.



del codigo creado me presenta el siguiente error como lo puedo corregir



Para corregir el error, necesitaría más información sobre el tipo de error que estás enfrentando. Sin embargo, te daré algunos pasos generales para depurar el código y resolver posibles problemas:

1. Errores Comunes y Soluciones:

a. Error de Lectura del Archivo Excel:

- **Error:** `FileNotFoundError: No such file or directory: 'calificaciones_alumnos.xlsx'`
- **Solución:** Asegúrate de que el archivo `calificaciones_alumnos.xlsx` esté en el mismo directorio donde estás ejecutando el script de Python, o proporciona la ruta completa al archivo.

```
python
```

```
df = pd.read_excel('ruta/completa/al/archivo/calificaciones_alumnos.xlsx')
```

b. Error de Importación de Módulos:

- **Error:** `ModuleNotFoundError: No module named 'pandas'`
- **Solución:** Instala los módulos necesarios ejecutando:

```
bash
```

```
pip install pandas matplotlib
```

c. Error de Tipo en las Columnas:

- **Error:** `KeyError: 'Mat_CalculoIntegral'` o `TypeError: cannot perform reduce with flexible type`
- **Solución:** Verifica que los nombres de las columnas en el archivo Excel coincidan exactamente con los que estás utilizando en el código. Asegúrate de que las columnas que contienen calificaciones no tengan valores no numéricos.

```
python
```

```
# Verifica los nombres de las columnas
print(df.columns)
```

d. Problemas con los Datos Faltantes:

- **Error:** `TypeError: unsupported operand type(s) for +: 'int' and 'str'`
- **Solución:** Asegúrate de que no haya valores faltantes o incorrectos en las columnas de calificaciones. Puedes manejar valores faltantes así:

```
python
```

```
df.fillna(0, inplace=True) # Reemplaza los valores faltantes con 0
```

2. Depuración del Código:



```
File "C:\codigos\M2_1Automatizando_Excel\.venv\Lib\site-
packages\pandas\compat\_optional.py", line 135, in import_optional_dependency
    module = importlib.import_module(name)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
return _bootstrap.gcd_import(name[level:], package, level)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
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:

```
File "C:\codigos\M2_1Automatizando_Excel\GraficaAlumnos.py", line 8, in <module>
    df = pd.read_excel('calificaciones_alumnos.xlsx')
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
io = ExcelFile(
    ^^^^^^^^^^^^
```

```
self._reader = self._engines[engine](
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
File "C:\codigos\M2_1Automatizando_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.



El error que estás viendo ocurre porque `pandas` intenta leer un archivo Excel utilizando la biblioteca `openpyxl`, que no está instalada en tu entorno. Para solucionar este problema, debes instalar `openpyxl`. Puedes hacerlo ejecutando el siguiente comando en tu entorno de Python:

```
bash
```

```
pip install openpyxl
```

Este comando instalará la biblioteca `openpyxl`, que es necesaria para que `pandas` pueda leer archivos Excel (.xlsx). Una vez que hayas instalado `openpyxl`, el error debería desaparecer y podrás ejecutar tu script sin problemas.



```
define libreria panda
```



`pandas` es una biblioteca de código abierto para el lenguaje de programación Python que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar. Es una de las bibliotecas más populares y esenciales en el ecosistema de Python para el análisis de datos.

Características Principales de `pandas`:

1. Estructuras de Datos Flexibles:

- **Series:** Estructura de datos unidimensional similar a un array unidimensional o a una lista de Python, pero con etiquetas (índices) para cada elemento.
- **DataFrame:** Estructura de datos bidimensional similar a una tabla de base de datos o una hoja de cálculo de Excel. Se compone de filas y columnas, donde cada columna puede tener un tipo de datos diferente (por ejemplo, números, texto, fechas).

2. Manejo de Datos Faltantes:

`pandas` permite identificar, filtrar y manipular datos faltantes (NaN) de manera eficiente.

3. Operaciones de Indexación y Selección:

Ofrece herramientas potentes para seleccionar y modificar subconjuntos de datos mediante etiquetas, posiciones, y condiciones.

4. Fusión y Combinar Conjuntos de Datos:

Permite combinar, fusionar y concatenar varios DataFrames de manera similar a SQL (join, merge).

5. Operaciones de Agrupación (GroupBy):

Facilita la agrupación de datos para aplicar funciones agregadas como sumas, medias, conteos, entre otros.

6. Manipulación de Datos Temporales:

Soporte integrado para datos temporales, con herramientas para realizar operaciones como resampling, rolling, y transformación de fechas.

7. Lectura y Escritura de Datos:

`pandas` es capaz de leer y escribir datos desde y hacia una amplia variedad de formatos, incluidos CSV, Excel, JSON, SQL, y más.

8. Funcionalidad de Graficado:

Aunque no es su propósito principal, `pandas` incluye métodos para crear gráficos simples directamente desde los DataFrames usando `matplotlib`.

Ejemplo Básico:

python

```
import pandas as pd

# Crear un DataFrame simple
data = {
    'Nombre': ['Ana', 'Luis', 'Marta'],
    'Edad': [23, 25, 22],
    'Ciudad': ['Madrid', 'Barcelona', 'Sevilla']
}

df = pd.DataFrame(data)

# Mostrar el DataFrame
print(df)

# Filtrar el DataFrame para mostrar solo personas mayores de 22 años
mayores_de_22 = df[df['Edad'] > 22]
print(mayores_de_22)
```

Este ejemplo básico crea un DataFrame a partir de un diccionario y luego muestra el DataFrame completo y una versión filtrada.

`pandas` es fundamental para cualquier trabajo que involucre manipulación y análisis de datos en Python, especialmente en campos como la ciencia de datos, el aprendizaje automático, la economía y la investigación.



define libreria openpyxl



`openpyxl` es una biblioteca de Python que permite leer y escribir archivos Excel en formato `.xlsx`. Este formato es el más reciente utilizado por Microsoft Excel y es el estándar de facto para trabajar con hojas de cálculo en muchas aplicaciones de datos y negocios.

Características Principales de `openpyxl`:

1. Leer y Escribir Archivos Excel:

- Puedes abrir archivos Excel existentes, modificar su contenido y guardarlos, así como crear nuevos archivos desde cero.

2. Manipulación de Hojas de Cálculo:

- Puedes acceder a diferentes hojas en un archivo Excel, crear nuevas hojas, eliminar hojas existentes y renombrarlas.

3. Lectura y Escritura de Celdas:

- `openpyxl` permite acceder a celdas individuales para leer o modificar su contenido. También puedes leer o escribir rangos de celdas o columnas completas.

4. Estilos y Formato:

- Ofrece la capacidad de aplicar estilos a las celdas, como fuentes, colores de fondo, bordes, alineación, y más.

5. Manejo de Fórmulas:

- Puedes trabajar con fórmulas dentro de las celdas, leer fórmulas existentes o escribir nuevas fórmulas que se calcularán en Excel.

6. Gráficos:

- Permite la creación de gráficos simples basados en datos de la hoja de cálculo.

7. Compatibilidad con Imágenes:

- Puedes insertar imágenes en las hojas de cálculo.

8. Manejo de Formato Condicional:

- Soporta la creación de reglas de formato condicional, que permiten cambiar el estilo de las celdas en función de sus valores.

9. Tablas Dinámicas (Pivot Tables):

- Aunque su soporte es más limitado, `openpyxl` puede manejar tablas dinámicas.

Ejemplo Básico de Uso:

python

```
from openpyxl import Workbook

# Crear un nuevo libro de trabajo (archivo Excel)
wb = Workbook()

# Seleccionar la hoja activa (por defecto, es la primera hoja)
ws = wb.active

# Escribir datos en celdas
ws['A1'] = 'Nombre'
ws['B1'] = 'Edad'

ws['A2'] = 'Ana'
ws['B2'] = 23

ws['A3'] = 'Luis'
ws['B3'] = 25

# Guardar el archivo
wb.save('ejemplo.xlsx')
```

En este ejemplo:

- **Se crea un nuevo archivo Excel** y se escribe en varias celdas.
- **Se guarda el archivo** con el nombre `ejemplo.xlsx`.

Cuándo Usar `openpyxl`:

- **Automatización de Procesos en Excel:** Cuando necesitas generar o modificar archivos Excel automáticamente desde un script de Python.
- **Análisis de Datos:** Si deseas analizar datos en archivos Excel sin necesidad de abrir Excel manualmente.
- **Interfaz con Otras Aplicaciones:** Ideal cuando estás desarrollando aplicaciones que necesitan leer o exportar datos en formato Excel para compartir con otros usuarios o sistemas.

`openpyxl` es especialmente útil para trabajar con Excel cuando el formato y la estructura de los datos son importantes, o cuando se necesita soporte avanzado para funcionalidades de Excel como gráficos, estilos y fórmulas.



define matplotlib



`matplotlib` es una biblioteca de Python ampliamente utilizada para crear visualizaciones de datos en 2D, como gráficos de líneas, gráficos de barras, histogramas, gráficos de dispersión y muchos otros tipos de gráficos. Es una herramienta esencial en el análisis de datos y la ciencia de datos, y es particularmente popular debido a su versatilidad y capacidad de personalización.

Características Principales de `matplotlib`:

1. Diversidad de Gráficos:

- Soporta una amplia gama de gráficos: líneas, barras, dispersión (scatter), histogramas, tortas (pie charts), gráficos de caja (box plots), y más.

2. Alta Personalización:

- Cada aspecto de un gráfico, como colores, tipos de línea, grosor, estilo de marcadores, leyendas, títulos, etiquetas de ejes, y más, se puede personalizar de manera detallada.

3. Interfaz Similar a MATLAB:

- `matplotlib` tiene una interfaz de usuario similar a la de MATLAB, lo que facilita su uso para aquellos que están familiarizados con MATLAB.

4. Integración con Otros Paquetes:

- Se integra bien con otras bibliotecas de Python, como `pandas` para el análisis de datos y `NumPy` para cálculos numéricos, lo que permite crear gráficos directamente desde DataFrames o arrays.

5. Capacidad para Crear Subgráficos:

- Permite la creación de gráficos con múltiples subgráficos (subplots) en una sola figura, lo que es útil para comparar diferentes conjuntos de datos en una sola visualización.

6. Gráficos Interactivos:

- Con `matplotlib`, es posible crear gráficos interactivos cuando se utiliza en conjunto con bibliotecas como `ipyml` y `Jupyter Notebooks`.

7. Soporte para Diferentes Backends:

- `matplotlib` soporta diferentes backends de gráficos, lo que permite generar gráficos para diferentes formatos de salida, como archivos PNG, PDF, SVG, y visualización en pantalla.

8. Mapas y Gráficos Avanzados:

- A través de bibliotecas complementarias como `Basemap` y `Cartopy`, `matplotlib` también permite crear mapas y gráficos geográficos.

Ejemplo Básico:

python

```
import matplotlib.pyplot as plt

# Datos de ejemplo
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Crear un gráfico de líneas
plt.plot(x, y, label='Datos')

# Agregar título y etiquetas
plt.title('Ejemplo de Gráfico de Líneas')
```



```
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar la leyenda
plt.legend()

# Mostrar el gráfico
plt.show()
```

En este ejemplo:

- `plt.plot(x, y)`: Crea un gráfico de líneas con los datos proporcionados.
- `plt.title`, `plt.xlabel`, `plt.ylabel`: Añaden título y etiquetas a los ejes.
- `plt.legend()`: Muestra la leyenda.
- `plt.show()`: Muestra el gráfico en pantalla.

Cuándo Usar `matplotlib`:

- **Visualización Básica y Avanzada**: Ideal para crear gráficos básicos y altamente personalizados.
- **Investigación y Ciencia de Datos**: Amplio uso en investigación y ciencia de datos para visualizar patrones, tendencias y relaciones en los datos.
- **Desarrollo de Informes**: Generación de gráficos para informes y presentaciones.

`matplotlib` es fundamental para cualquier trabajo de visualización de datos en Python, proporcionando herramientas potentes y flexibles para transformar datos en gráficos claros y comprensibles.



define pyplot



`pyplot` es un submódulo de la biblioteca `matplotlib` en Python. Es la interfaz de más alto nivel que ofrece `matplotlib` para crear y personalizar gráficos de manera sencilla y eficiente. La estructura de `pyplot` se asemeja mucho a la de MATLAB, lo que facilita la creación de gráficos con una sintaxis simple y directa, utilizando funciones que mantienen el estado (state-based).

Características Principales de `pyplot`:

1. **Creación de Gráficos Rápida y Sencilla**:
 - Permite crear gráficos rápidamente con solo unas pocas líneas de código. Ofrece funciones para crear gráficos de líneas, barras, dispersión (scatter), histogramas, tortas, entre otros.
2. **Mantiene el Estado de los Gráficos**:
 - `pyplot` funciona como una máquina de estado, donde se mantiene el estado del gráfico actual (como el eje, el título, las etiquetas, etc.). Esto permite agregar elementos a un gráfico en construcción de manera sencilla.
3. **Funciones para Configuración**:
 - Ofrece funciones como `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.legend()`, y `plt.grid()` para configurar aspectos específicos del gráfico, como el título, las etiquetas de los ejes, la leyenda, y la cuadrícula.
4. **Mostrar y Guardar Gráficos**:

- `plt.show()`: Muestra el gráfico en una ventana interactiva.
- `plt.savefig('nombre_archivo.png')`: Guarda el gráfico en un archivo con el formato especificado (PNG, PDF, SVG, etc.).

5. Creación de Subgráficos:

- `pyplot` facilita la creación de subgráficos dentro de una misma figura mediante la función `plt.subplot()`, lo que permite visualizar múltiples gráficos juntos.

6. Personalización de Gráficos:

- Permite personalizar detalles como colores, estilos de línea, marcadores, límites de los ejes, y mucho más, proporcionando un control fino sobre la apariencia del gráfico.

Ejemplo Básico:

python

```
import matplotlib.pyplot as plt

# Datos de ejemplo
x = [0, 1, 2, 3, 4]
y = [0, 1, 4, 9, 16]

# Crear un gráfico de líneas
plt.plot(x, y, marker='o', linestyle='-', color='b', label='Cuadrado')

# Configurar el título y las etiquetas de los ejes
plt.title('Gráfico de Ejemplo con pyplot')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar la leyenda
plt.legend()

# Mostrar el gráfico
plt.show()
```

Explicación del Código:

- `plt.plot(x, y, marker='o', linestyle='-', color='b', label='Cuadrado')`: Crea un gráfico de líneas con marcadores circulares, línea sólida, y color azul, etiquetado como "Cuadrado".
- `plt.title('Gráfico de Ejemplo con pyplot')`: Añade un título al gráfico.
- `plt.xlabel('Eje X')` y `plt.ylabel('Eje Y')`: Etiquetan los ejes X e Y respectivamente.
- `plt.legend()`: Muestra la leyenda en el gráfico.
- `plt.show()`: Muestra el gráfico en una ventana.

Cuándo Usar `pyplot`:

- **Visualización Rápida:** Cuando necesitas generar gráficos simples y rápidos para exploración de datos o presentación de resultados.
- **Aprendizaje y Prototipado:** Ideal para aprender a graficar en Python y para la creación de prototipos de gráficos antes de implementaciones más complejas.
- **Integración con Notebooks:** Muy utilizado en Jupyter Notebooks para mostrar gráficos de manera interactiva y en línea.

`pyplot` es la herramienta principal dentro de `matplotlib` para crear gráficos de manera intuitiva, haciendo que la visualización de datos en Python sea accesible y poderosa al mismo tiempo.



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



Memoria actualizada

Aquí tienes un programa en Python que agrega una columna llamada `Mat_Fisica` con valores aleatorios entre 0 y 100 con un decimal a tu archivo `calificaciones_alumnos.xlsx`:

python

```
# Este programa lee un archivo Excel que contiene las calificaciones de varios alumnos.
# Luego, agrega una columna llamada "Mat_Fisica" con valores aleatorios entre 0 y 100, y
# guarda el archivo actualizado.

import pandas as pd
import numpy as np

# Leer el archivo Excel y cargar los datos en un DataFrame de pandas
df = pd.read_excel('calificaciones_alumnos.xlsx')

# 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, len(df)), 1)

# Guardar el DataFrame actualizado de vuelta en el archivo Excel
df.to_excel('calificaciones_alumnos_actualizado.xlsx', index=False)
```

Explicación del código:

- Lectura del archivo Excel:** Se utiliza `pandas` para leer los datos desde el archivo `calificaciones_alumnos.xlsx` y cargarlos en un DataFrame.
- Generación de valores aleatorios:**
 - Se utiliza `numpy` para generar valores aleatorios uniformes entre 0 y 100.
 - `np.round()` redondea los valores a un decimal, que es lo que se almacena en la nueva columna `Mat_Fisica`.
- Guardar el archivo actualizado:** El DataFrame con la nueva columna se guarda en un nuevo archivo llamado `calificaciones_alumnos_actualizado.xlsx`.

Este código agregará la columna `Mat_Fisica` a tu archivo Excel y guardará una copia del archivo con el nuevo nombre especificado.



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



Aquí tienes el programa modificado que agrega la columna `Mat_Fisica` y luego ordena la tabla por la columna `Nombre` antes de guardar el nuevo archivo Excel:

python

```
# Este programa lee un archivo Excel que contiene las calificaciones de varios alumnos.
# Luego, agrega una columna llamada "Mat_Fisica" con valores aleatorios entre 0 y 100,
# ordena la tabla por la columna "Nombre", y guarda el archivo actualizado.

import pandas as pd
import numpy as np

# Leer el archivo Excel y cargar los datos en un DataFrame de pandas
df = pd.read_excel('calificaciones_alumnos.xlsx')

# 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, len(df)), 1)

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

# Guardar el DataFrame ordenado y actualizado de vuelta en un archivo Excel
df.to_excel('calificaciones_alumnos_actualizado.xlsx', index=False)
```

Explicación del código:

1. **Lectura del archivo Excel:** Igual que antes, el archivo `calificaciones_alumnos.xlsx` se carga en un DataFrame.
2. **Generación de la columna `Mat_Fisica`:** Se añaden valores aleatorios entre 0 y 100 con un decimal a la nueva columna `Mat_Fisica`.
3. **Ordenar por `Nombre`:**
 - `df.sort_values(by='Nombre', inplace=True)`: Ordena el DataFrame por la columna `Nombre` en orden alfabético. El parámetro `inplace=True` asegura que el DataFrame se modifique en su lugar.
4. **Guardar el archivo ordenado:** El DataFrame actualizado y ordenado se guarda en `calificaciones_alumnos_actualizado.xlsx`.

Este código generará un nuevo archivo Excel donde la tabla estará ordenada alfabéticamente por el nombre de los alumnos.



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



Aquí tienes un programa en Python que carga tu archivo Excel y muestra el número de registros (filas) y el número de campos (columnas) en tu tabla:

python

```
# Este programa lee un archivo Excel que contiene calificaciones de varios alumnos
# y muestra el número de registros (filas) y el número de campos (columnas) en la tabla.

import pandas as pd

# Leer el archivo Excel y cargar los datos en un DataFrame de pandas
df = pd.read_excel('calificaciones_alumnos.xlsx')

# Obtener el número de registros (filas) en el DataFrame
num_registros = len(df)

# Obtener el número de campos (columnas) en el DataFrame
num_campos = len(df.columns)

# Imprimir el número de registros y campos
print(f'Número de registros (filas): {num_registros}')
print(f'Número de campos (columnas): {num_campos}')
```

Explicación del código:

- Lectura del archivo Excel:** Utiliza `pandas` para leer los datos del archivo `calificaciones_alumnos.xlsx` y cargarlos en un DataFrame.
- Número de registros (filas):**
 - `len(df)`: Calcula el número de filas en el DataFrame, que representa el número de registros.
- Número de campos (columnas):**
 - `len(df.columns)`: Calcula el número de columnas en el DataFrame, que representa el número de campos.
- Impresión de resultados:** Utiliza `print()` para mostrar el número de registros y campos en la consola.

Este programa te dará la cantidad total de registros y campos en tu tabla.



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



Aquí tienes un programa en Python que carga tu archivo Excel y determina qué campos (columnas) son numéricos:

python

```
# Este programa lee un archivo Excel que contiene calificaciones de varios alumnos
# y muestra qué campos (columnas) son numéricos.

import pandas as pd

# Leer el archivo Excel y cargar los datos en un DataFrame de pandas
df = pd.read_excel('calificaciones_alumnos.xlsx')

# Identificar columnas numéricas
# Utiliza el método select_dtypes para seleccionar solo las columnas con tipos de datos
```

```
numéricos
columnas_numericas = df.select_dtypes(include='number').columns

# Imprimir los nombres de las columnas numéricas
print('Campos numéricos:')
for columna in columnas_numericas:
    print(columna)
```

Explicación del código:

- 1. Lectura del archivo Excel:** Usa `pandas` para leer el archivo `calificaciones_alumnos.xlsx` y cargar los datos en un DataFrame.
- 2. Identificación de columnas numéricas:**
 - `df.select_dtypes(include='number')`: Selecciona las columnas cuyo tipo de datos es numérico. `include='number'` asegura que solo se incluyan columnas con tipos numéricos (como enteros y flotantes).
 - `.columns`: Extrae los nombres de estas columnas.
- 3. Impresión de resultados:**
 - `print('Campos numéricos:')`: Imprime un encabezado para los campos numéricos.
 - `for columna in columnas_numericas:`: Itera sobre los nombres de las columnas numéricas y las imprime.

Este código te mostrará una lista de todas las columnas en el archivo Excel que contienen datos numéricos.

ChatGPT puede cometer errores. Comprueba la información importante.