

# Conexión Python MySQL

You said:



Ayúdame a crear un programa de Python para conectarse a una MySQL

ChatGPT said:



 Memoria actualizada

Para crear un programa en Python que se conecte a una base de datos MySQL, puedes usar el módulo `mysql-connector-python` o `PyMySQL`. A continuación te muestro un ejemplo básico utilizando `mysql-connector-python`:

## 1. Instala el módulo necesario

Primero, necesitas instalar el módulo `mysql-connector-python` si no lo tienes ya instalado:

```
bash
```

```
pip install mysql-connector-python
```

## 2. Escribe el código para conectarse a la base de datos

Aquí te dejo un ejemplo básico de cómo conectarte a una base de datos MySQL:

```
python
```

```
import mysql.connector
from mysql.connector import Error

def connect_to_mysql():
    try:
        # Establece la conexión
        connection = mysql.connector.connect(
            host='tu_host', # Ejemplo: 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            db_info = connection.get_server_info()
            print(f"Conectado a MySQL Server versión {db_info}")
            cursor = connection.cursor()
            cursor.execute("SELECT DATABASE();")
            record = cursor.fetchone()
            print(f"Conectado a la base de datos: {record}")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")
```

```
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    connect_to_mysql()
```

### 3. Configura tus credenciales

Asegúrate de reemplazar ``tu\_host``, ``tu\_base\_de\_datos``, ``tu\_usuario``, y ``tu\_contraseña`` con la información correspondiente de tu servidor MySQL.

### 4. Ejecuta el programa

Guarda el script y ejecútalo en tu entorno de desarrollo. Si la conexión es exitosa, verás información sobre la versión de MySQL y la base de datos a la que te has conectado.

Este ejemplo básico te permitirá establecer una conexión, realizar consultas simples y cerrar la conexión de manera segura.

You said:



Ahora quiero consultar los registros de la tabla curso y que los convierta en formato Dataframe y los exporte a un Excel

ChatGPT said:



 Memoria actualizada

Puedes realizar una consulta de la tabla "curso", convertir los resultados en un DataFrame de `pandas` y luego exportarlos a un archivo Excel. Aquí te dejo un ejemplo completo:

#### 1. Instala los módulos necesarios

Si no tienes `pandas` y `openpyxl` instalados, puedes instalarlos con:

bash

```
pip install pandas openpyxl mysql-connector-python
```

#### 2. Escribe el código para consultar, convertir y exportar

python

```

import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='tu_host',
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            cursor = connection.cursor(dictionary=True)
            cursor.execute("SELECT * FROM curso")
            records = cursor.fetchall()

            # Convertir los registros en un DataFrame de pandas
            df = pd.DataFrame(records)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("curso.xlsx", index=False, engine='openpyxl')
            print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()

```

### 3. Explicación del código

- **Conexión a MySQL:** El código establece una conexión a la base de datos MySQL utilizando las credenciales proporcionadas.
- **Consulta SQL:** Se realiza una consulta `SELECT * FROM curso` para obtener todos los registros de la tabla "curso".
- **DataFrame:** Los registros obtenidos se convierten en un DataFrame de `pandas`.
- **Exportación a Excel:** El DataFrame se exporta a un archivo Excel llamado `curso.xlsx` usando `openpyxl` como motor para manejar el formato de Excel.

### 4. Ejecuta el código

Guarda el código y ejecútalo. Si todo está configurado correctamente, deberías obtener un archivo `curso.xlsx` en tu directorio de trabajo con los registros de la tabla "curso".

Si necesitas más personalización en el archivo Excel o la consulta, puedes ajustar el código según tus necesidades.

You said:



el ultimo codigo me genera el siguiente error: `TypeError: Cannot convert numpy.ndarray to numpy.ndarray`

ChatGPT said:



El error `TypeError: Cannot convert numpy.ndarray to numpy.ndarray` puede ocurrir en algunos casos al trabajar con `pandas` y `openpyxl`. Una solución es asegurarse de que los datos se manejen correctamente durante la exportación. Aquí te muestro cómo ajustar el código para evitar ese error:

## Código Ajustado

python

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='tu_host',
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            cursor = connection.cursor(dictionary=True)
            cursor.execute("SELECT * FROM curso")
            records = cursor.fetchall()

            # Convertir los registros en un DataFrame de pandas
            df = pd.DataFrame(records)

            # Manejar posibles problemas de conversión de tipos
            for column in df.columns:
                if df[column].dtype == 'object':
                    df[column] = df[column].astype(str)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("curso.xlsx", index=False, engine='openpyxl')
            print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()
```

## Explicación de los Cambios

- **Conversión de Tipos de Datos:** Se agregó un bucle para convertir todas las columnas que tienen el tipo de datos `object` a cadenas de texto (`str`). Esto puede prevenir problemas de conversión que surgen al exportar a Excel.

Este ajuste debería resolver el problema que mencionaste. Intenta ejecutar el código nuevamente y verifica si se genera correctamente el archivo Excel.

You said:



El error persiste y se debe a que la estructura del código no permite guardar la información en el formato requerido, específicamente en las siguientes líneas de código: # Convertir los registros en un DataFrame de pandas

```
df = pd.DataFrame(records)

# Manejar posibles problemas de conversión de tipos
for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = df[column].astype(str)

# Exportar el DataFrame a un archivo Excel
df.to_excel("curso.xlsx", index=False, engine='openpyxl')
print("Los datos se han exportado a 'curso.xlsx' con éxito.")
```

ChatGPT said:



Gracias por la aclaración. Si el problema persiste en las líneas que mencionaste, podríamos explorar una forma alternativa para convertir los registros en un DataFrame y exportarlos a Excel. A continuación, te muestro un enfoque revisado que debería evitar el error:

## Código Revisado

python

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='tu_host',
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM curso")
            # Obtén los nombres de las columnas
            column_names = [desc[0] for desc in cursor.description]
```

```

        # Obtén los datos
        records = cursor.fetchall()

        # Convertir los registros en un DataFrame de pandas usando columnas
        específicas
        df = pd.DataFrame(records, columns=column_names)

        # Exportar el DataFrame a un archivo Excel
        df.to_excel("curso.xlsx", index=False, engine='openpyxl')
        print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()

```

## Ajustes Realizados

1. **Obtención de Nombres de Columnas:** Antes de convertir los registros en un DataFrame, se obtienen los nombres de las columnas a partir de ``cursor.description``.
2. **Creación del DataFrame:** Se utiliza ``pd.DataFrame(records, columns=column_names)`` para asegurarse de que los datos se estructuren correctamente, respetando los nombres de las columnas.

Este enfoque debería resolver el problema que mencionaste y permitir la exportación correcta de los datos a un archivo Excel. Prueba este código y verifica si el error persiste.

You said:



El siguiente código incluye los datos a la base de datos con la que se realiza la conexión, dicho código sigue arrojando el mismo error, que cambios sugieres? import mysql.connector  
import pandas as pd  
from mysql.connector import Error

```

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58', # Ejemplo: 'localhost'
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()

```

```

cursor.execute("SELECT * FROM curso")
# Obtén los nombres de las columnas
column_names = [desc[0] for desc in cursor.description]
# Obtén los datos
records = cursor.fetchall()

# Convertir los registros en un DataFrame de pandas usando columnas específicas
df = pd.DataFrame(records, columns=column_names)

# Exportar el DataFrame a un archivo Excel
df.to_excel("curso.xlsx", index=False, engine='openpyxl')
print("Los datos se han exportado a 'curso.xlsx' con éxito.")

except Error as e:
    print(f"Error al conectar a MySQL: {e}")

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()

```

ChatGPT said:



Para resolver el problema que estás enfrentando, te sugiero que pruebes los siguientes ajustes en tu código:

## 1. Conversión explícita de tipos de datos

Dado que el problema parece estar relacionado con la conversión de ciertos tipos de datos, intenta convertir todos los objetos (como `datetime`, `decimal`, etc.) a cadenas (`str`). Esto puede evitar errores durante la exportación a Excel.

## 2. Código ajustado

Aquí tienes el código revisado:

python

```

import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL

```

```

connection = mysql.connector.connect(
    host='195.179.238.58',
    database='u927419088_testing_sql',
    user='u927419088_admin',
    password='#Admin12345#'
)

if connection.is_connected():
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM curso")

    # Obtén los nombres de las columnas
    column_names = [desc[0] for desc in cursor.description]

    # Obtén los datos y convierte cada registro a cadena
    records = cursor.fetchall()
    records = [[str(item) for item in record] for record in records]

    # Convertir los registros en un DataFrame de pandas usando columnas
    específicas
    df = pd.DataFrame(records, columns=column_names)

    # Exportar el DataFrame a un archivo Excel
    df.to_excel("curso.xlsx", index=False, engine='openpyxl')
    print("Los datos se han exportado a 'curso.xlsx' con éxito.")

except Error as e:
    print(f"Error al conectar a MySQL: {e}")

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()

```

## Explicación de los Cambios

1. **Conversión de Datos a Cadenas:** Se añadió una conversión explícita de todos los elementos en los registros a cadenas de texto (`str`). Esto ayuda a manejar mejor los diferentes tipos de datos que podrían causar problemas al exportar a Excel.
2. **Exportación a Excel:** La exportación se realiza después de asegurar que todos los datos están en un formato compatible.

## 3. Prueba el Código

Ejecuta el código ajustado. Este enfoque debería eliminar el error relacionado con la conversión de `numpy.ndarray` y permitir la exportación correcta de los datos a un archivo Excel.

You said:



Sigue apareciendo el error: TypeError: Cannot convert numpy.ndarray to numpy.ndarray

ChatGPT said:





El error puede estar relacionado con una incompatibilidad entre los tipos de datos o estructuras internas al manejar el DataFrame o al intentar exportar el DataFrame a Excel. Como última alternativa, podríamos intentar evitar el uso de NumPy directamente, asegurándonos de que todos los datos se mantengan en tipos nativos de Python antes de convertirlos en un DataFrame.

Aquí te dejo un código revisado que debería evitar este error:

## Código Revisado

python

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58',
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM curso")

            # Obtén los nombres de las columnas
            column_names = [desc[0] for desc in cursor.description]

            # Obtén los datos y convierte cada registro a tipos nativos de Python
            records = cursor.fetchall()
            processed_records = []

            for record in records:
                processed_record = []
                for item in record:
                    if isinstance(item, (bytes, bytearray)):
                        processed_record.append(item.decode('utf-8'))
                    else:
                        processed_record.append(item)
                processed_records.append(tuple(processed_record))

            # Convertir los registros en un DataFrame de pandas usando columnas
            específicas
            df = pd.DataFrame(processed_records, columns=column_names)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("curso.xlsx", index=False, engine='openpyxl')
            print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()
```

## Explicación de los Cambios

1. **Decodificación de bytes:** Si alguno de los datos es de tipo ``bytes``, se decodifica a una cadena de texto.
2. **Procesamiento de los Registros:** Todos los registros son procesados y almacenados como tuplas nativas de Python. Esto debería evitar cualquier conversión involuntaria a tipos de datos que puedan causar problemas.

## Siguiente Paso

Intenta ejecutar este código revisado. Este enfoque debería prevenir el error de conversión a ``numpy.ndarray`` y permitir la exportación correcta de los datos a un archivo Excel.

You said:



sigue persistiendo el mismo error

ChatGPT said:



Lamento que el error persista. En este caso, podemos probar un enfoque alternativo para la conversión y exportación de los datos. Vamos a intentar usar una técnica diferente para asegurar que no haya problemas con la conversión a ``numpy.ndarray``.

## Enfoque Alternativo

En lugar de usar ``fetchall()``, podemos iterar a través de los resultados de la consulta y construir el DataFrame directamente. Aquí tienes un código modificado para intentar evitar el problema:

## Código Alternativo

python

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58',
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor(dictionary=True)
            cursor.execute("SELECT * FROM curso")

            # Obtener los nombres de las columnas
            column_names = cursor.column_names
```

```

        # Crear una lista de diccionarios
        records = []
        for row in cursor:
            records.append(row)

        # Convertir los registros en un DataFrame de pandas usando columnas
        específicas
        df = pd.DataFrame(records, columns=column_names)

        # Exportar el DataFrame a un archivo Excel
        df.to_excel("curso.xlsx", index=False, engine='openpyxl')
        print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()

```

## Explicación de los Cambios

1. **Uso de `dictionary=True` en el Cursor:** Esto permite que cada fila de resultados se devuelva como un diccionario, lo que facilita la conversión a un DataFrame sin problemas de tipo.
2. **Iteración sobre el Cursor:** En lugar de usar `fetchall()`, iteramos sobre el cursor para construir una lista de diccionarios. Esto evita problemas que podrían surgir con la conversión de tipos de datos.

## Verificación

Prueba este código y verifica si el problema persiste. Este enfoque debería manejar mejor los datos y evitar errores relacionados con `numpy.ndarray`.

You said:



aun continua el mismo error

ChatGPT said:



Dado que el problema persiste, intentemos otra solución para solucionar el problema de la conversión a `numpy.ndarray`. En lugar de trabajar directamente con `pandas` para la conversión y exportación, podríamos intentar simplificar el proceso para asegurar que los datos estén en un formato que `pandas` pueda manejar sin problemas.

Aquí te presento un enfoque utilizando una conversión intermedia a una lista de listas y evitando el uso de `numpy` directamente:

## Código Alternativo Simplificado

python

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_curso_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58',
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM curso")

            # Obtener los nombres de las columnas
            column_names = [desc[0] for desc in cursor.description]

            # Crear una lista de listas para los datos
            data = []
            for row in cursor:
                data.append(list(row)) # Convertir cada fila a una lista

            # Convertir los datos a un DataFrame
            df = pd.DataFrame(data, columns=column_names)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("curso.xlsx", index=False, engine='openpyxl')
            print("Los datos se han exportado a 'curso.xlsx' con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_curso_to_excel()
```

## Explicación del Enfoque

1. **Conversión a Lista de Listas:** En lugar de utilizar `fetchall()`, iteramos sobre el cursor y convertimos cada fila a una lista. Esto asegura que los datos estén en un formato simple y compatible.
2. **Creación del DataFrame:** Se crea el DataFrame directamente a partir de la lista de listas, lo que debería evitar problemas de conversión.

## Verificación

Prueba este código para ver si soluciona el problema. La clave aquí es simplificar el formato de los datos antes de la conversión, lo que debería evitar el error relacionado con `numpy.ndarray`. Si el

problema persiste, podría ser útil verificar los tipos de datos específicos en la tabla `curso` para asegurarnos de que todos los datos son compatibles con `pandas`.

You said:



continua el mismo error

ChatGPT said:



Si el error persiste a pesar de los cambios, es posible que haya un problema más profundo en cómo se manejan los datos en la conexión o en la conversión de los datos. Aquí te ofrezco un enfoque diferente que utiliza `csv` como intermediario antes de convertir a Excel. Esto puede ayudar a identificar y solucionar problemas con la conversión de datos.

## Enfoque Alternativo con CSV Intermedio

1. Guarda los datos en un archivo CSV.
2. Luego convierte el archivo CSV a Excel.

## Código

python

```
import mysql.connector
import pandas as pd
import csv
from mysql.connector import Error

def export_curso_to_csv():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58',
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM curso")

            # Obtener los nombres de las columnas
            column_names = [desc[0] for desc in cursor.description]

            # Guardar los datos en un archivo CSV
            with open('curso_temp.csv', 'w', newline='', encoding='utf-8') as csvfile:
                csvwriter = csv.writer(csvfile)
                csvwriter.writerow(column_names)
                for row in cursor:
                    csvwriter.writerow(row)

            print("Los datos se han exportado a 'curso_temp.csv' con éxito.")
            convert_csv_to_excel()

    except Error as e:
```

```
print(f"Error al conectar a MySQL: {e}")

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

def convert_csv_to_excel():
    try:
        # Leer el archivo CSV y convertirlo a un DataFrame
        df = pd.read_csv('curso_temp.csv')

        # Exportar el DataFrame a un archivo Excel
        df.to_excel('curso.xlsx', index=False, engine='openpyxl')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    export_curso_to_csv()
```

## Explicación del Código

1. **Exportar a CSV:** Los datos se guardan primero en un archivo CSV (`curso\_temp.csv`). Este paso utiliza la biblioteca `csv`, que es muy confiable para manejar datos simples y puede evitar problemas de conversión.
2. **Convertir CSV a Excel:** Luego, leemos el archivo CSV y lo convertimos a Excel usando `pandas`.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script.
2. **Revisa los Archivos Generados:** Verifica si `curso\_temp.csv` se genera correctamente y si luego se convierte a `curso.xlsx` sin errores.

Este enfoque divide el proceso en dos pasos, lo que puede ayudar a aislar el problema y garantizar que los datos se manejen correctamente en cada etapa.

You said:



funciono la exportacion a csv, pero ahora al guardarlos en excel arroja el siguiente error: Error al convertir CSV a Excel: Cannot convert numpy.ndarray to numpy.ndarray

ChatGPT said:



Es un buen avance que la exportación a CSV haya funcionado. El problema al convertir CSV a Excel podría estar relacionado con cómo se está leyendo el archivo CSV o con datos específicos en el archivo.

Aquí hay un enfoque ajustado para manejar la conversión de CSV a Excel, asegurando que los datos sean manejados adecuadamente:

## Código Revisado

python

```
import pandas as pd

def convert_csv_to_excel():
    try:
        # Leer el archivo CSV y convertirlo a un DataFrame
        df = pd.read_csv('curso_temp.csv', dtype=str) # Leer todos los datos como cadenas
        para evitar problemas de tipo

        # Exportar el DataFrame a un archivo Excel
        df.to_excel('curso.xlsx', index=False, engine='openpyxl')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    convert_csv_to_excel()
```

## Explicación del Código

1. **Leer el CSV como Cadenas:** Utiliza `dtype=str` al leer el archivo CSV para asegurar que todos los datos se interpreten como cadenas. Esto puede prevenir problemas con tipos de datos no esperados.
2. **Exportar a Excel:** Luego convierte el DataFrame a un archivo Excel.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script para convertir el archivo CSV a Excel.
2. **Verifica el Archivo Generado:** Comprueba que `curso.xlsx` se cree correctamente sin errores.

Este enfoque asegura que los datos se traten de manera uniforme como cadenas durante la lectura del CSV, lo que debería evitar problemas de conversión.

You said:



surge el error: Error al convertir CSV a Excel: Cannot convert numpy.ndarray to numpy.ndarray

ChatGPT said:



Lamentablemente, el error `Cannot convert numpy.ndarray to numpy.ndarray` persiste, lo que sugiere que podría haber un problema subyacente en la interacción entre pandas y openpyxl, o un problema con los datos en sí.

Vamos a probar un enfoque diferente, eliminando posibles problemas con la conversión al intentar exportar el DataFrame a Excel de una manera más básica. A continuación, te muestro un código que

usa un enfoque alternativo para manejar la conversión, evitando problemas potenciales con la conversión de tipos de datos.

## Código Alternativo para Conversión CSV a Excel

python

```
import pandas as pd

def convert_csv_to_excel():
    try:
        # Leer el archivo CSV
        df = pd.read_csv('curso_temp.csv')

        # Convertir todos los valores del DataFrame a cadena de texto
        for column in df.columns:
            df[column] = df[column].astype(str)

        # Exportar el DataFrame a un archivo Excel
        df.to_excel('curso.xlsx', index=False, engine='openpyxl')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    convert_csv_to_excel()
```

## Explicación del Código

1. **Leer el CSV:** Lee el archivo CSV normalmente.
2. **Convertir a Cadenas:** Convierte todos los valores del DataFrame a cadenas de texto para evitar problemas con tipos de datos.
3. **Exportar a Excel:** Exporta el DataFrame a Excel utilizando `openpyxl` como motor.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script para convertir el archivo CSV a Excel.
2. **Verifica el Archivo:** Comprueba que `curso.xlsx` se cree correctamente y sin errores.

Si el problema persiste, verifica los siguientes puntos:

1. **Datos en CSV:** Asegúrate de que el archivo CSV no contenga datos corruptos o inusuales.
2. **Versión de Bibliotecas:** Asegúrate de que `pandas` y `openpyxl` están actualizados a versiones recientes.
3. **Entorno:** Considera probar el código en un entorno limpio o diferente para descartar problemas específicos del entorno.

Si sigues enfrentando problemas, podría ser útil analizar más a fondo los datos específicos del CSV y considerar la posibilidad de que los datos mismos puedan estar causando problemas.

You said:





persiste el mismo error

ChatGPT said:



Entiendo que el error persiste, lo que indica que podría haber un problema más profundo con el entorno o las bibliotecas que estás utilizando. A continuación, te propongo una solución alternativa utilizando `xlsxwriter` en lugar de `openpyxl` para la conversión de CSV a Excel, y asegurar que los datos se manejen de la manera más básica posible.

## Código Alternativo con `xlsxwriter`

Primero, asegúrate de instalar `xlsxwriter` si aún no lo has hecho:

bash

```
pip install xlsxwriter
```

Luego, usa el siguiente código:

python

```
import pandas as pd

def convert_csv_to_excel():
    try:
        # Leer el archivo CSV
        df = pd.read_csv('curso_temp.csv', dtype=str) # Asegúrate de que todos los datos
        se manejen como cadenas

        # Exportar el DataFrame a un archivo Excel usando xlsxwriter
        df.to_excel('curso.xlsx', index=False, engine='xlsxwriter')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    convert_csv_to_excel()
```

## Explicación del Código

1. **Leer el CSV como Cadenas:** Se lee el archivo CSV asegurando que todos los datos se manejen como cadenas (`dtype=str`).
2. **Exportar a Excel con `xlsxwriter`:** Se utiliza el motor `xlsxwriter` para la exportación, que es una alternativa a `openpyxl`.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script para convertir el archivo CSV a Excel.
2. **Verifica el Archivo Generado:** Comprueba que `curso.xlsx` se cree correctamente y sin errores.

## Posibles Soluciones Adicionales

Si el problema persiste, aquí hay algunos pasos adicionales para la resolución de problemas:

1. **Verifica Datos en CSV:** Asegúrate de que el archivo CSV no contenga caracteres especiales o datos que puedan estar causando problemas. Puedes intentar abrir el archivo CSV en un editor de texto para revisar su contenido.
2. **Verifica Versión de Bibliotecas:** Asegúrate de que `pandas`, `xlsxwriter`, y `openpyxl` estén actualizados a sus versiones más recientes. A veces, errores en bibliotecas pueden ser resueltos en versiones más recientes.
3. **Entorno Aislado:** Considera ejecutar el código en un entorno virtual limpio para asegurarte de que no haya conflictos con otras bibliotecas instaladas.
4. **Depuración Detallada:** Intenta agregar impresiones de depuración para ver si hay datos específicos en el DataFrame que podrían estar causando el problema:

python

```
print(df.head()) # Imprime las primeras filas del DataFrame
print(df.dtypes) # Imprime los tipos de datos en el DataFrame
```

Estos pasos deberían ayudarte a identificar y resolver el problema subyacente.

You said:



el error se puede deber al separador ?

ChatGPT said:



Sí, el error podría deberse a un problema con el separador utilizado en el archivo CSV. Si el separador en el archivo CSV no coincide con el que pandas está utilizando por defecto, puede provocar errores en la lectura y la conversión de datos.

Por defecto, `pandas.read\_csv` utiliza la coma (`,` ) como separador. Si tu archivo CSV utiliza un separador diferente (por ejemplo, punto y coma `;` ), tendrás que especificarlo al leer el archivo. Aquí te muestro cómo ajustar el código para manejar diferentes separadores:

## Ajuste del Separador en el CSV

1. **Determina el Separador Correcto:** Si no estás seguro del separador en el archivo CSV, puedes abrir el archivo en un editor de texto para verificar qué carácter se usa para separar los valores.
2. **Modifica el Código para Leer el CSV con el Separador Correcto:**

python

```
import pandas as pd

def convert_csv_to_excel():
    try:
        # Especifica el separador correcto, por ejemplo, ';' si es un punto y coma
        df = pd.read_csv('curso_temp.csv', sep=';', dtype=str) # Ajusta el 'sep'
        según sea necesario

        # Exportar el DataFrame a un archivo Excel usandoxlsxwriter
        df.to_excel('curso.xlsx', index=False, engine='xlsxwriter')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    convert_csv_to_excel()
```

## Explicación del Código

- `sep=';'`: Este parámetro especifica el carácter que se utiliza para separar los valores en el archivo CSV. Cambia el valor de `sep` según el separador utilizado en tu archivo CSV.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script con el separador adecuado.
2. **Verifica el Archivo Generado:** Asegúrate de que el archivo `curso.xlsx` se cree correctamente sin errores.

## Alternativas para Diagnosticar el Problema

Si el problema persiste incluso después de ajustar el separador, prueba estos pasos adicionales:

1. **Abrir CSV en un Editor de Texto:** Asegúrate de que el archivo CSV se está guardando correctamente y verifica que el separador utilizado sea el que estás especificando en el código.
2. **Guardar CSV de Nuevo:** Si es posible, intenta guardar el archivo CSV de nuevo desde la aplicación que lo generó, asegurándote de utilizar el formato y separador adecuados.
3. **Ejemplo de CSV:** Si puedes proporcionar un pequeño ejemplo del contenido de tu archivo CSV, puedo ofrecerte una ayuda más específica para ajustar el código.

Estos ajustes deberían ayudarte a manejar correctamente el archivo CSV y evitar problemas relacionados con el separador.

You said:



no se solucionó el error

ChatGPT said:



Dado que el error persiste y hemos probado varias soluciones sin éxito, vamos a intentar un enfoque más básico para investigar el problema. Vamos a dividir el proceso en partes más pequeñas para identificar el origen del problema.

## Pasos para Diagnosticar el Problema

1. **Verifica el Archivo CSV:** Asegúrate de que el archivo CSV se genere correctamente y esté en el formato esperado.

2. **Prueba una Lectura Básica del CSV:**

Verifica si el archivo CSV se puede leer y visualizar correctamente con pandas sin intentar exportar a Excel. Esto nos ayudará a confirmar que el archivo CSV está bien formado.

python

```
import pandas as pd

def test_read_csv():
    try:
        # Lee el archivo CSV con el separador predeterminado (coma)
        df = pd.read_csv('curso_temp.csv')
        print("CSV leído correctamente:")
        print(df.head()) # Imprime las primeras filas para verificar
    except Exception as e:
        print(f"Error al leer el CSV: {e}")

if __name__ == "__main__":
    test_read_csv()
```

Ejecuta este código para ver si pandas puede leer el archivo CSV sin errores.

3. **Guarda el DataFrame como CSV:**

Antes de convertir a Excel, asegúrate de que el DataFrame se guarda correctamente en un archivo CSV temporal.

python

```
def save_df_to_csv():
    try:
        # Lee el archivo CSV
        df = pd.read_csv('curso_temp.csv')

        # Guarda el DataFrame en un nuevo archivo CSV
        df.to_csv('test_output.csv', index=False)
        print("El DataFrame se ha guardado correctamente en 'test_output.csv'.")
    except Exception as e:
        print(f"Error al guardar el DataFrame como CSV: {e}")

if __name__ == "__main__":
    save_df_to_csv()
```

4. **Convierte el CSV a Excel con un Proceso Manual:**

Intenta abrir el archivo CSV (`curso\_temp.csv`) en una aplicación como Microsoft Excel o Google Sheets, y luego guarda el archivo manualmente como Excel (`\*.xlsx`). Esto puede ayudar a verificar si hay problemas con los datos.

## 5. Revisa las Versiones de las Bibliotecas:

Asegúrate de que las bibliotecas `pandas`, `xlsxwriter`, y `openpyxl` estén actualizadas a las versiones más recientes.

```
bash
```

```
pip install --upgrade pandas xlsxwriter openpyxl
```

## 6. Revisar Datos Específicos:

Si puedes identificar filas o columnas específicas que podrían estar causando el problema, revisa los datos de esas filas para ver si contienen caracteres especiales o datos no válidos.

## Código Final para Diagnóstico

Prueba el siguiente código que combina lectura, guardado y conversión para asegurarnos de que todos los pasos se realizan correctamente:

```
python
```

```
import pandas as pd

def convert_csv_to_excel():
    try:
        # Lee el archivo CSV con el separador adecuado
        df = pd.read_csv('curso_temp.csv', dtype=str) # Asegúrate de que todos los datos
        se manejen como cadenas

        # Imprime información sobre el DataFrame
        print("DataFrame leído correctamente:")
        print(df.head())
        print(df.dtypes)

        # Exportar el DataFrame a un archivo Excel usando xlsxwriter
        df.to_excel('curso.xlsx', index=False, engine='xlsxwriter')
        print("El archivo CSV se ha convertido a 'curso.xlsx' con éxito.")

    except Exception as e:
        print(f"Error al convertir CSV a Excel: {e}")

if __name__ == "__main__":
    convert_csv_to_excel()
```

Estos pasos deberían ayudarte a identificar la fuente del problema. Si encuentras algún problema específico en cualquiera de los pasos, por favor proporciónamelo y podré ofrecerte una solución más enfocada.

You said:



no puede leer el archivo csv

ChatGPT said:



Si pandas no puede leer el archivo CSV, es importante verificar la integridad del archivo CSV y asegurarnos de que esté en el formato correcto. Aquí hay algunos pasos detallados para diagnosticar y solucionar problemas con la lectura del archivo CSV:

## Pasos para Diagnosticar Problemas con el Archivo CSV

### 1. Verifica el Archivo CSV Manualmente

- **Abre el Archivo CSV:** Usa un editor de texto (como Notepad++ o VSCode) para abrir el archivo CSV y verificar su contenido.
- **Verifica el Formato:** Asegúrate de que el archivo CSV esté separado correctamente por comas (o el separador que estás utilizando). Asegúrate de que los datos no contengan caracteres no esperados o líneas en blanco.

### 2. Verifica el Separador

- **Usa el Separador Correcto:** Asegúrate de que el separador utilizado en el archivo CSV coincida con el especificado en el código. Si no estás seguro, intenta leer el archivo CSV con diferentes separadores.

### 3. Prueba Leer un CSV Simple

- **Crea un CSV Simple de Prueba:** Crea un archivo CSV simple para verificar si pandas puede leerlo correctamente. Guarda el siguiente contenido en un archivo llamado `test.csv`:

```
CSV

columna1,columna2,columna3
valor1,valor2,valor3
valor4,valor5,valor6
```

- **Lee el Archivo CSV Simple:** Usa el siguiente código para leer el archivo CSV simple:

```
python

import pandas as pd

def test_read_simple_csv():
    try:
        df = pd.read_csv('test.csv')
        print("CSV simple leído correctamente:")
        print(df.head())
    except Exception as e:
        print(f"Error al leer el CSV simple: {e}")

if __name__ == "__main__":
    test_read_simple_csv()
```

### 4. Revisa el Código para Leer el CSV Original

- **Código para Leer el CSV Original:** Usa este código para leer el archivo CSV original y proporciona detalles adicionales si se produce un error.

```
python

import pandas as pd
```

```
def read_original_csv():
    try:
        # Ajusta el separador según el archivo CSV
        df = pd.read_csv('curso_temp.csv', sep=',', dtype=str) # Cambia 'sep'
        si es necesario
        print("CSV leído correctamente:")
        print(df.head())
        print(df.dtypes)
    except Exception as e:
        print(f"Error al leer el CSV original: {e}")

if __name__ == "__main__":
    read_original_csv()
```

## 5. Revisa el Archivo CSV por Errores Comunes

- **Encoding:** Asegúrate de que el archivo CSV esté guardado en una codificación compatible, como UTF-8. Si el archivo tiene una codificación diferente, especifica el encoding al leer el archivo:

python

```
df = pd.read_csv('curso_temp.csv', sep=',', encoding='utf-8', dtype=str) #
Ajusta 'encoding' si es necesario
```

- **Líneas en Blanco:** Asegúrate de que no haya líneas en blanco al final del archivo CSV, ya que esto puede causar problemas con la lectura.

## Pasos para Resolver Problemas

- **Si Puedes Leer el CSV Simplemente:** Si puedes leer el archivo CSV simple sin problemas, pero no el archivo original, el problema podría estar relacionado con el contenido o el formato del archivo CSV original.
- **Si No Puedes Leer el CSV Simplemente:** Si el archivo CSV simple también presenta problemas, el problema puede estar relacionado con la instalación de pandas o el entorno.

Estos pasos deberían ayudarte a identificar y resolver problemas con la lectura del archivo CSV. Si el problema persiste, por favor proporciona más detalles sobre cualquier error específico que encuentres, para que pueda ofrecerte una solución más precisa.

You said:



Sigue arrojando el error: Error al leer el CSV simple: Cannot convert numpy.ndarray to numpy.ndarray

ChatGPT said:



Dado que el error `Cannot convert numpy.ndarray to numpy.ndarray` persiste incluso con un archivo CSV simple, esto sugiere que el problema podría estar relacionado con la instalación de pandas,

numpy o algún conflicto en tu entorno. Aquí hay algunas acciones adicionales que puedes tomar para resolver el problema:

## 1. Reinstalar Pandas y Numpy

A veces, los errores pueden ser causados por instalaciones corruptas de bibliotecas. Reinstalar `pandas` y `numpy` podría resolver el problema.

```
bash
```

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

## 2. Probar en un Entorno Virtual

Crear un entorno virtual limpio para aislar el problema puede ser útil. Aquí están los pasos:

```
bash
```

```
# Crear un nuevo entorno virtual
python -m venv myenv

# Activar el entorno virtual
# En Windows
myenv\Scripts\activate
# En macOS/Linux
source myenv/bin/activate

# Instalar pandas y numpy en el entorno virtual
pip install pandas numpy
```

Luego, ejecuta el código dentro de este entorno virtual para ver si el problema persiste.

## 3. Código Alternativo con Escritura Básica

Si el problema persiste, puedes intentar una solución básica para verificar si el problema está en la conversión o en la lectura de los datos. Usa este código para leer y luego imprimir datos de un archivo CSV básico.

```
python
```

```
import pandas as pd

def read_simple_csv():
    try:
        # Lee un archivo CSV simple
        df = pd.read_csv('test.csv', dtype=str)
        print("CSV leído correctamente:")
        print(df.head())
        print(df.dtypes)
    except Exception as e:
        print(f"Error al leer el CSV simple: {e}")

if __name__ == "__main__":
    read_simple_csv()
```



## 4. Verificar la Configuración del Entorno

Asegúrate de que no haya conflictos entre las bibliotecas. Revisa los siguientes puntos:

- **Versiones de Pandas y Numpy:** Verifica las versiones instaladas:

```
python
```

```
import pandas as pd
import numpy as np
print(pd.__version__)
print(np.__version__)
```

- **Archivos en el Entorno:** Asegúrate de que no haya archivos con nombres que puedan estar causando conflictos, como ``numpy.py`` o ``pandas.py``, en el directorio donde estás ejecutando el script.

## 5. Ejecutar en un Script Limpio

A veces, ejecutar el código en un archivo independiente puede ayudar a evitar conflictos. Guarda y ejecuta el siguiente código en un archivo nuevo llamado ``read_csv_test.py``:

```
python
```

```
import pandas as pd

def test_csv_reading():
    try:
        # Prueba con un CSV simple
        df = pd.read_csv('test.csv', dtype=str)
        print("CSV leído correctamente:")
        print(df.head())
        print(df.dtypes)
    except Exception as e:
        print(f"Error al leer el CSV: {e}")

if __name__ == "__main__":
    test_csv_reading()
```

## 6. Revisar Logs de Errores

Asegúrate de revisar los logs de errores y la traza de errores completos. A veces, la información adicional en el error puede ayudar a identificar el problema específico.

Estos pasos deberían ayudarte a identificar y resolver el problema. Si después de seguir estos pasos el error sigue ocurriendo, por favor, comparte cualquier información adicional sobre el entorno o errores específicos que encuentres para poder ofrecer una ayuda más precisa.

You said:



el error se debía a la version de numpy que se tenia instalado

ChatGPT said:



#### Memoria actualizada

Me alegra saber que identificaste y resolviste el problema actualizando numpy. Si necesitas más ayuda con la manipulación de datos o cualquier otro problema relacionado con Python, no dudes en preguntar. ¡Estoy aquí para ayudarte!

You said:



Ahora quiero consultar los registros de la tabla `asignatura` y que los convierta en formato DataFrame y los exporte a un Excel

ChatGPT said:



Para consultar los registros de una tabla llamada ``asignatura`` en una base de datos MySQL, convertirlos en un DataFrame de pandas y exportarlos a un archivo Excel, puedes seguir un proceso similar al que utilizaste anteriormente. Aquí te muestro el código ajustado para esta tarea:

## Código para Consultar Registros y Exportar a Excel

1. **Asegúrate de tener ``pandas`` y ``openpyxl`` instalados.** Si aún no los tienes, instálalos:

```
bash
```

```
pip install pandas openpyxl mysql-connector-python
```

2. **Código Python:**

```
python
```

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def export_asignatura_to_excel():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58', # Cambia esto según tu configuración
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM asignatura")

            # Obtén los nombres de las columnas
```

```

column_names = [desc[0] for desc in cursor.description]

# Obtén los datos
records = cursor.fetchall()

# Convertir los registros en un DataFrame de pandas
df = pd.DataFrame(records, columns=column_names)

# Exportar el DataFrame a un archivo Excel
df.to_excel("asignatura.xlsx", index=False, engine='openpyxl')
print("Los datos se han exportado a 'asignatura.xlsx' con éxito.")

except Error as e:
    print(f"Error al conectar a MySQL: {e}")

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

if __name__ == "__main__":
    export_asignatura_to_excel()

```

## Explicación del Código

1. **Conexión a la Base de Datos:** Se establece una conexión a la base de datos MySQL usando ``mysql.connector``.
2. **Consulta de Datos:** Se ejecuta una consulta SQL para seleccionar todos los registros de la tabla ``asignatura``.
3. **Obtención de Columnas:** Se obtienen los nombres de las columnas a partir de la descripción del cursor.
4. **Creación del DataFrame:** Los registros obtenidos se convierten en un DataFrame de pandas.
5. **Exportación a Excel:** El DataFrame se exporta a un archivo Excel llamado ``asignatura.xlsx`` usando el motor ``openpyxl``.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script para verificar que los datos se exporten correctamente a ``asignatura.xlsx``.
2. **Revisa el Archivo Excel:** Asegúrate de que ``asignatura.xlsx`` contenga los datos correctos.

Este código debería ayudarte a consultar y exportar los datos de la tabla ``asignatura`` a un archivo Excel. Si encuentras algún problema, no dudes en decírmelo para que pueda asistirte.

You said:



Ahora quiero que insertes un registro en la tabla asignatura, genera los datos de forma aleatoria

ChatGPT said:



Para insertar un registro aleatorio en la tabla `asignatura`, puedes utilizar la biblioteca `random` para generar datos aleatorios. A continuación, te muestro cómo hacerlo. Asumiré que la tabla `asignatura` tiene algunas columnas típicas como `id`, `nombre`, `descripcion`, etc. Si tu tabla tiene diferentes columnas, ajusta el código en consecuencia.

## Código para Insertar un Registro Aleatorio

### 1. Instala las bibliotecas necesarias si aún no lo has hecho:

```
bash
```

```
pip install mysql-connector-python
```

### 2. Código Python:

```
python
```

```
import mysql.connector
from mysql.connector import Error
import random
import faker

# Inicializa el generador de datos ficticios
fake = faker.Faker()

def insert_random_record():
    try:
        # Conectarse a la base de datos MySQL
        connection = mysql.connector.connect(
            host='195.179.238.58', # Cambia esto según tu configuración
            database='u927419088_testing_sql',
            user='u927419088_admin',
            password='#Admin12345#'
        )

        if connection.is_connected():
            cursor = connection.cursor()

            # Generar datos aleatorios
            id_asignatura = random.randint(1, 1000) # Ajusta el rango según tus
necesidades
            nombre_asignatura = fake.word()
            descripcion_asignatura = fake.sentence()

            # Inserción de datos en la tabla
            insert_query = """
            INSERT INTO asignatura (id, nombre, descripcion)
            VALUES (%s, %s, %s)
            """
            record = (id_asignatura, nombre_asignatura, descripcion_asignatura)

            cursor.execute(insert_query, record)
            connection.commit()

            print("Registro insertado con éxito.")

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")
```

```
if __name__ == "__main__":
    insert_random_record()
```

## Explicación del Código

1. **Generador de Datos Ficticios:** Utiliza la biblioteca ``faker`` para generar datos aleatorios. Puedes instalarla usando ``pip install faker``.
2. **Generar Datos Aleatorios:** Crea valores aleatorios para ``id``, ``nombre``, y ``descripcion`` de la asignatura.
3. **Conectar a la Base de Datos:** Establece una conexión a la base de datos MySQL.
4. **Insertar Datos:** Ejecuta una consulta SQL para insertar el registro en la tabla ``asignatura``.
5. **Cerrar Conexión:** Asegúrate de cerrar el cursor y la conexión después de la inserción.

## Verificación

1. **Ejecuta el Código:** Guarda y ejecuta el script para insertar un registro aleatorio en la tabla ``asignatura``.
2. **Revisa la Base de Datos:** Verifica en la base de datos que el nuevo registro se haya insertado correctamente.

Este código debería permitirte insertar un registro aleatorio en la tabla ``asignatura``. Si la tabla tiene columnas diferentes o si necesitas ajustar los datos generados, adapta el código según tus necesidades.

You said:



Imagen cargada

considera la siguiente estructura de datos para rediseñar el código que propones para insertar un registro: curso idAsignatura nombre cuatrimestre creditos caracter coordinador

|   |       |   |   |   |             |       |
|---|-------|---|---|---|-------------|-------|
| 1 | AS001 | ÁLGEBRA LINEAL                                    | 2 | 6 | obligatoria | PR001 |
| 1 | AS002 | Análisis exploratorio de datos                    | 1 | 6 | obligatoria | PR002 |
| 1 | AS003 | Análisis matemático                               | 1 | 6 | obligatoria | PR003 |
| 1 | AS004 | Fundamentos de computadores y sistemas operativos | 2 | 6 | obligatoria | PR004 |
| 1 | AS005 | Fundamentos de organización de empresas           | 1 | 6 | obligatoria | PR005 |
| 1 | AS006 | Fundamentos de programación                       | 1 | 6 | obligatoria | PR004 |
| 1 | AS007 | Matemáticas discretas                             | 1 | 6 | obligatoria | PR003 |
| 1 | AS008 | Modelos estadísticos para la toma de decisiones I | 2 | 6 | obligatoria | PR008 |

ChatGPT said:



You've reached our limits of messages. Please try again later.

Ocurrió un error al generar una respuesta.

 Regenerar

ChatGPT puede cometer errores. Comprueba la información importante.