

Guía Práctica

TEMA:
MySQL + Python

Martes, 04 Mayo 2021



Práctica #2 - MySQL + Pyhton 2º - CLASE

Venció ayer a las 11:20 • Se cerró ayer a las 11:20

Instrucciones

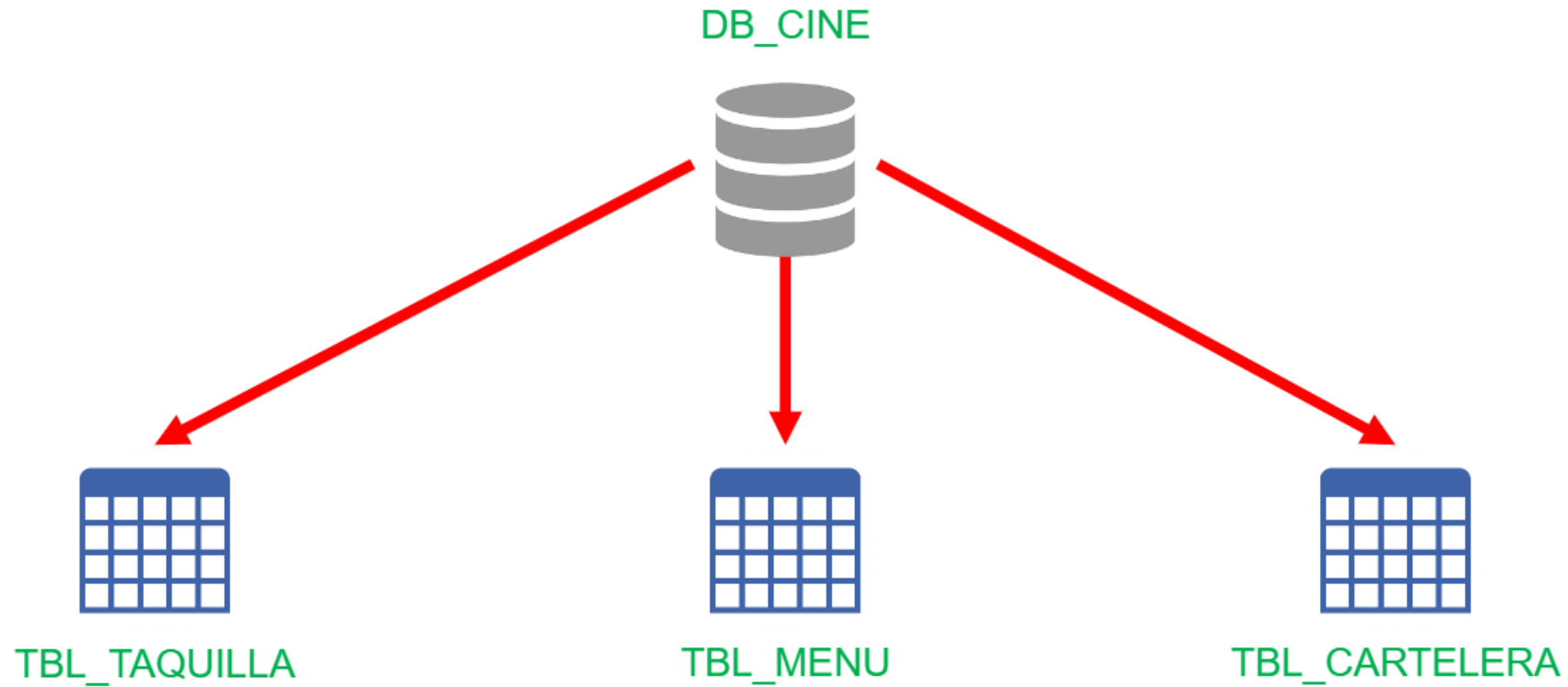
ACTIVIDAD INDIVIDUAL

Seguir las siguientes instrucciones:

1. Desarrollar en **PhpMyAdmin** una base de datos con la siguiente estructura (ver **DB_CINE_DISEÑO.pdf** adjunto)
2. Exportar la base de datos en formato **.sql** y renombrar el archivo bajo la siguiente nomenclatura:
 - Cuenta_Nombre_DB_CINE, Ejemplo: **13_Deyanira_DB_CINE.sql**
3. Crear un archivo Python con el nombre **cine_database.py** donde debe configurar la conexión a la BASE DE DATOS previamente creada y sus respectivas funciones en base al (ver **mydatabase.png** adjunto)
4. Descargar la **INTERFAZ de Tkinter** correspondiente al archivo adjunto nombrado **ui_cine.py** para desarrollar el ejercicio de clase donde debe editarlo de tal manera que le permita insertar datos en la base de datos creada y configurar la función en base al (ver **mysql_ui_form.png** adjunto)
5. Crear un repositorio en GITHUB y subir archivos Python:
 1. **ui_cine.py**
 2. **cine_database.py**
 3. **13_Deyanira_DB_CINE.sql**
6. Adjuntar a la plataforma el enlace del REPOSITORIO DE GITHUB con todo su trabajo realizado en clase.

Diseño Base de Datos

PASO 1



Diseño Base de Datos

PASO 1



RESULTADO ESPERADO

Los límites y tipos de CAMPOS son definidos a criterio propio



The screenshot displays the phpMyAdmin interface. On the left, a sidebar shows a database tree with 'db_cine' selected, containing tables 'tbl_taquilla', 'db_grupo8', 'db_pruebas', and 'db_red_social'. The main area shows the 'Estructura de tabla' (Table Structure) for 'tbl_taquilla' in the 'db_cine' database. The table has five columns: ID_TAQUILLA (int(11), primary key, auto-increment), SALA (varchar(100), utf8mb4_general_ci), BUTAKAS (int(100)), BOLETOS (int(100)), and PRECIO (int(255)). Each column has 'Cambiar' (Change) and 'Eliminar' (Delete) actions. At the bottom, there are options to 'Seleccionar todo' (Select all) and 'Agregar a columnas centrales' (Add to central columns).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 ID_TAQUILLA	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
<input type="checkbox"/>	2 SALA	varchar(100)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 BUTAKAS	int(100)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	4 BOLETOS	int(100)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	5 PRECIO	int(255)			No	Ninguna			Cambiar Eliminar Más

☐ Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Primaria Único Índice Espacial Texto completo

Agregar a columnas centrales Eliminar de las columnas centrales

Práctica #2 - MySQL + Python 2º - CLASE

Venció ayer a las 11:20 • Se cerró ayer a las 11:20

Instrucciones

ACTIVIDAD INDIVIDUAL

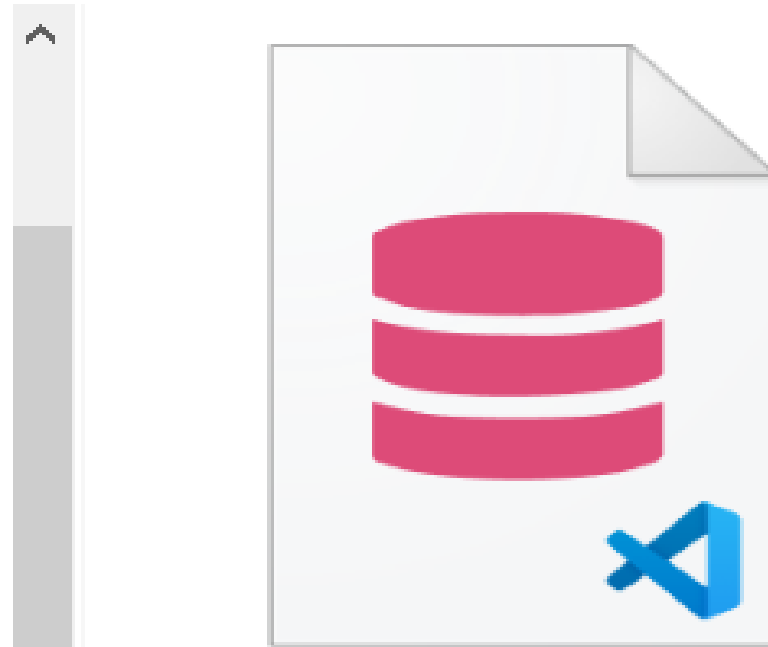
Seguir las siguientes instrucciones:

1. Desarrollar en **PhpMyAdmin** una base de datos con la siguiente estructura (ver **DB_CINE_DISEÑO.pdf** adjunto)
2. Exportar la base de datos en formato **.sql** y renombrar el archivo bajo la siguiente nomenclatura:
 - Cuenta_Nombre_DB_CINE, Ejemplo: **13_Deyanira_DB_CINE.sql**
3. Crear un archivo Python con el nombre **cine_database.py** donde debe configurar la conexión a la BASE DE DATOS previamente creada y sus respectivas funciones en base al (ver **mydatabase.png** adjunto)
4. Descargar la **INTERFAZ de Tkinter** correspondiente al archivo adjunto nombrado **ui_cine.py** para desarrollar el ejercicio de clase donde debe editarlo de tal manera que le permita insertar datos en la base de datos creada y configurar la función en base al (ver **mysql_ui_form.png** adjunto)
5. Crear un repositorio en GITHUB y subir archivos Python:
 1. **ui_cine.py**
 2. **cine_database.py**
 3. **13_Deyanira_DB_CINE.sql**
6. Adjuntar a la plataforma el enlace del REPOSITORIO DE GITHUB con todo su trabajo realizado en clase.

RESULTADO ESPERADO

PASO 2

Prácticas de Clase > Práctica #2 - MySQL + Python



13_Deyanira_DB_CINE

Práctica #2 - MySQL + Python 2º - CLASE

Venció ayer a las 11:20 • Se cerró ayer a las 11:20

Instrucciones

ACTIVIDAD INDIVIDUAL

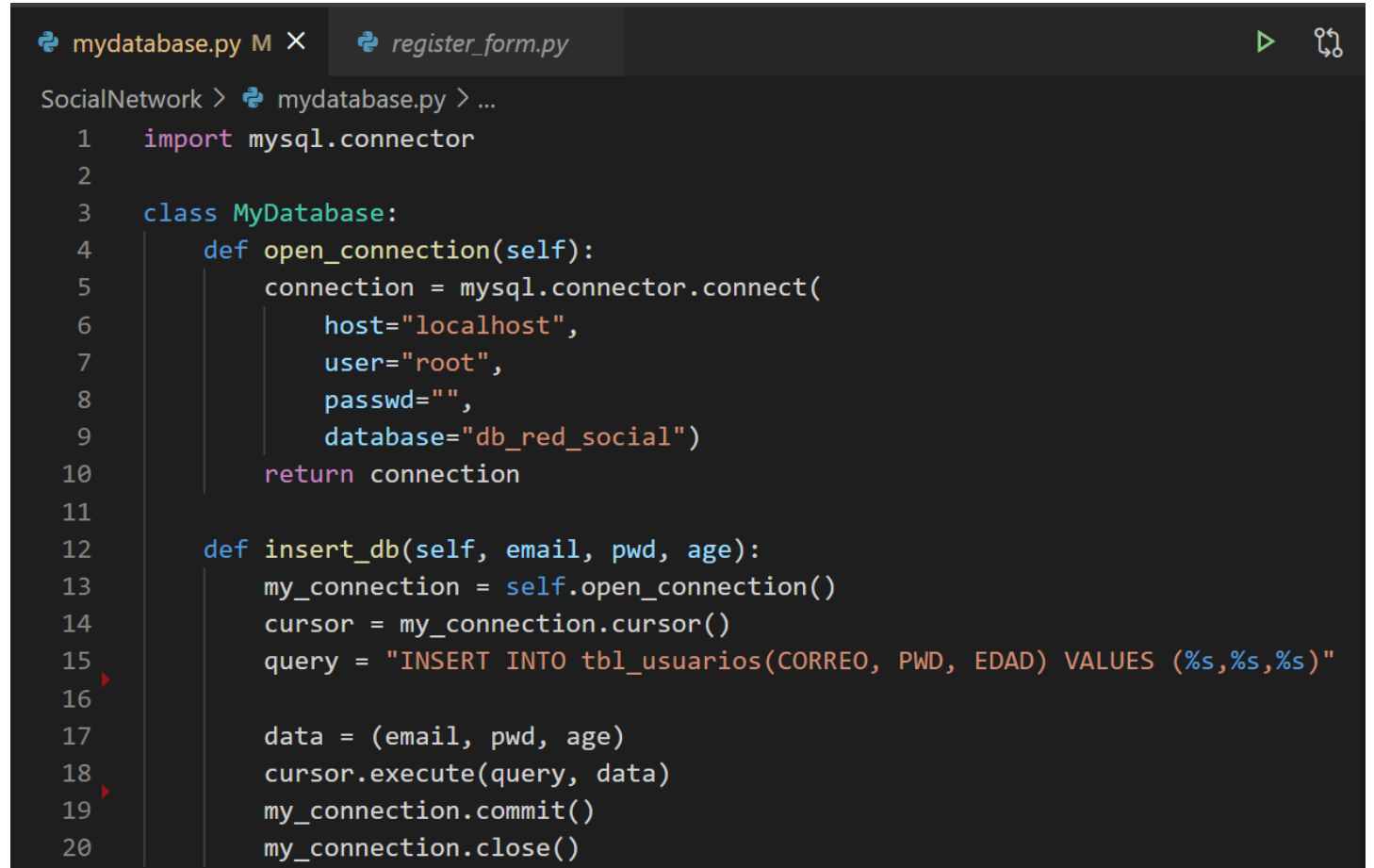
Seguir las siguientes instrucciones:

1. Desarrollar en **PhpMyAdmin** una base de datos con la siguiente estructura (ver **DB_CINE_DISEÑO.pdf** adjunto)
2. Exportar la base de datos en formato **.sql** y renombrar el archivo bajo la siguiente nomenclatura:
 - Cuenta_Nombre_DB_CINE, Ejemplo: **13_Deyanira_DB_CINE.sql**
3. Crear un archivo Python con el nombre **cine_database.py** donde debe configurar la conexión a la BASE DE DATOS previamente creada y sus respectivas funciones en base al archivo (ver **mydatabase.png** adjunto)
4. Descargar la **INTERFAZ de Tkinter** correspondiente al archivo adjunto nombrado **ui_cine.py** para desarrollar el ejercicio de clase donde debe editarlo de tal manera que le permita insertar datos en la base de datos creada y configurar la función en base al (ver **mysql_ui_form.png** adjunto)
5. Crear un repositorio en GITHUB y subir archivos Python:
 1. **ui_cine.py**
 2. **cine_database.py**
 3. **13_Deyanira_DB_CINE.sql**
6. Adjuntar a la plataforma el enlace del REPOSITORIO DE GITHUB con todo su trabajo realizado en clase.



PAUTA
mydatabase.png

PASO 3



```
mydatabase.py M X  register_form.py
SocialNetwork > mydatabase.py > ...
1  import mysql.connector
2
3  class MyDatabase:
4      def open_connection(self):
5          connection = mysql.connector.connect(
6              host="localhost",
7              user="root",
8              passwd="",
9              database="db_red_social")
10         return connection
11
12     def insert_db(self, email, pwd, age):
13         my_connection = self.open_connection()
14         cursor = my_connection.cursor()
15         query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
16
17         data = (email, pwd, age)
18         cursor.execute(query, data)
19         my_connection.commit()
20         my_connection.close()
```

Creando cine_database.py

PASO 3 – Parte 1

ui_cine.py

cine_database.py X

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!  
2  import mysql.connector  
3  
4
```

Creando cine_database.py

PASO 3 – Parte 2

ui_cine.py ×

cine_database.py ×

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #          para poder crear la conexión en el archivo Python
6  #          de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8
```

Creando cine_database.py

PASO 3 – Parte 3

ui_cine.py

cine_database.py ●

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos correspondiente
10     def open_connection(self):
11
```

Creando cine_database.py

PASO 3 – Parte 4

```
ui_cine.py X cine_database.py X
Práctica #2 - MySQL + Python > cine_database.py
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos correspondiente
10     def open_connection(self):
11         # PASO 4: Crear una variable "connection",
12         #         para almacenar la configuración de la base de datos correspondiente
13         connection = mysql.connector.connect(
14             host="localhost",
15             user="root",
16             passwd="",
17             database="db_red_social"])
18
```

Creando cine_database.py

PASO 3 – Parte 5

```
ui_cine.py  cine_database.py ●
Práctica #2 - MySQL + Python > cine_database.py
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos correspondiente
10     def open_connection(self):
11         # PASO 4: Crear una variable "connection",
12         #         para almacenar la configuración de la base de datos correspondiente
13         connection = mysql.connector.connect(
14             host="localhost",
15             user="root",
16             passwd="",
17             database="db_red_social")
18         # PASO 5: Retornar la variable "connection",
19         #         para reutilizarla en el archivo Python de la Interfaz (UI_Tkinter) correspondiente
20         return connection
```

Creando
cine_database.py

PASO 3 – Parte 6

ui_cine.py

cine_database.py X

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos
10     def open_connection(self):
11         # PASO 4: Crear una variable "connection",
12         #         para almacenar la configuración de la base de datos
13         connection = mysql.connector.connect(
14             host="localhost",
15             user="root",
16             passwd="",
17             database="db_red_social")
18         # PASO 5: Retornar la variable "connection",
19         #         para reutilizarla en el archivo Python de la Interfa
20         #         correspondiente
21         return connection
22
23     # PASO 6: Crear la función "insert_db",
24     #         para crear registros dentro de una tabla específica
25     def insert_db(self, email, pwd, age):
26
```

Creando cine_database.py

PASO 3 – Parte 7

ui_cine.py

cine_database.py X

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos correspondiente
10     def open_connection(self):
11         # PASO 4: Crear una variable "connection",
12         #         para almacenar la configuración de la base de datos correspondiente
13         connection = mysql.connector.connect(
14             host="localhost",
15             user="root",
16             passwd="",
17             database="db_red_social")
18         # PASO 5: Retornar la variable "connection",
19         #         para reutilizarla en el archivo Python de la Interfaz (UI_Tkinter)
20         #         correspondiente
21         return connection
22
23     # PASO 6: Crear la función "insert_db",
24     #         para crear registros dentro de una tabla específica
25     def insert_db(self, email, pwd, age):
26         # PASO 7: Crear la variable "my_connection",
27         #         para crear y almacenar la conexión a la base de datos mediante
28         #         la función "open_connection", creada en el PASO 4.
29         my_connection = self.open_connection()
30
```


Creando
cine_database.py

PASO 3 – Parte 8

ui_cine.py

cine_database.py X

Práctica #2 - MySQL + Python > cine_database.py

```
1  # PASO 1: Importar el módulo "mysql.connector" previamente ¡INSTALADO!
2  import mysql.connector
3
4  # PASO 2: Crear una clase llamada "MyDatabase",
5  #         para poder crear la conexión en el archivo Python
6  #         de la Interfaz (UI_Tkinter) correspondiente
7  class MyDatabase:
8      # PASO 3: Crear la función "open_connection",
9      #         para abrir la conexión y tener acceso a la base de datos correspondiente
10     def open_connection(self):
11         # PASO 4: Crear una variable "connection",
12         #         para almacenar la configuración de la base de datos correspondiente
13         connection = mysql.connector.connect(
14             host="localhost",
15             user="root",
16             passwd="",
17             database="db_red_social")
18         # PASO 5: Retornar la variable "connection",
19         #         para reutilizarla en el archivo Python de la Interfaz (UI_Tkinter)
20         #         correspondiente
21         return connection
22
23     # PASO 6: Crear la función "insert_db",
24     #         para crear registros dentro de una tabla específica
25     def insert_db(self, email, pwd, age):
26         # PASO 7: Crear la variable "my_connection",
27         #         para crear y almacenar la conexión a la base de datos mediante
28         #         la función "open_connection", creada en el PASO 4.
29         my_connection = self.open_connection()
30         # PASO 8: Crear la variable "cursor",
31         #         para crear un puntero que nos permita acceder a un lugar específico
32         #         de nuestra base de datos
33         cursor = my_connection.cursor()
```

Creando cine_database.py

PASO 3 – Parte 9

```
23 # PASO 6: Crear la función "insert_db",
24 #     para crear registros dentro de una tabla específica
25 def insert_db(self, email, pwd, age):
26     # PASO 7: Crear la variable "my_connection",
27     #     para crear y almacenar la conexión a la base de datos mediante
28     #     la función "open_connection", creada en el PASO 4.
29     my_connection = self.open_connection()
30     # PASO 8: Crear la variable "cursor",
31     #     para crear un puntero que nos permita acceder a un lugar específico
32     #     de nuestra base de datos
33     cursor = my_connection.cursor()
34     # PASO 9: Crear la variable "query",
35     #     para crear la instrucción SQL que nos permita INSERTAR o CREAR un registro
36     #     en la base de datos
37     #
38     # IMPORTANTE: La estructura de la instrucción SQL está definida
39     #     y lo único que cambia son los CAMPOS y el NOMBRE DE LA TABLA
40     query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
41
```

```

23 # PASO 6: Crear la función "insert_db",
24 #     para crear registros dentro de una tabla específica
25 def insert_db(self, email, pwd, age):
26     # PASO 7: Crear la variable "my_connection",
27     #     para crear y almacenar la conexión a la base de datos mediante
28     #     la función "open_connection", creada en el PASO 4.
29     my_connection = self.open_connection()
30     # PASO 8: Crear la variable "cursor",
31     #     para crear un puntero que nos permita acceder a un lugar específico
32     #     de nuestra base de datos
33     cursor = my_connection.cursor()
34     # PASO 9: Crear la variable "query",
35     #     para crear la instrucción SQL que nos permita INSERTAR o CREAR un registro
36     #     en la base de datos
37     #
38     # IMPORTANTE: La estructura de la instrucción SQL está definida
39     #             y lo único que cambia son los CAMPOS y el NOMBRE DE LA TABLA
40     query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
41     # PASO 10: Crear la variable "data",
42     #     para almacenar los datos o información correspondiente al registro
43     #     que deseamos insertar en el paso anterior
44     #
45     # IMPORTANTE: los valores de las variables (EMAIL, PWD, AGE) debe OBTENERLOS
46     #             mediante la función GET() en la Interfaz de Tkinter y enviarlos
47     #             como PARÁMETROS a la función insert_db()
48     data = (email, pwd, age)
49

```

Creando
 cine_database.py

PASO 3 – Parte 10

Creando cine_database.py

PASO 3 – Parte 11

```
# PASO 6: Crear la función "insert_db",
#           para crear registros dentro de una tabla específica
def insert_db(self, email, pwd, age):
    # PASO 7: Crear la variable "my_connection",
    #           para crear y almacenar la conexión a la base de datos mediante
    #           la función "open_connection", creada en el PASO 4.
    my_connection = self.open_connection()
    # PASO 8: Crear la variable "cursor",
    #           para crear un puntero que nos permita acceder a un lugar específico
    #           de nuestra base de datos
    cursor = my_connection.cursor()
    # PASO 9: Crear la variable "query",
    #           para crear la instrucción SQL que nos permita INSERTAR o CREAR un registro
    #           en la base de datos
    #
    # IMPORTANTE: La estructura de la instrucción SQL está definida
    #               y lo único que cambia son los CAMPOS y el NOMBRE DE LA TABLA
    query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
    # PASO 10: Crear la variable "data",
    #           para almacenar los datos o información correspondiente al registro
    #           que deseamos insertar en el paso anterior
    #
    # IMPORTANTE: los valores de las variables (EMAIL, PWD, AGE) debe OBTENERLOS
    #               mediante la función GET() en la Interfaz de Tkinter y enviarlos
    #               como PARÁMETROS a la función insert_db()
    data = (email, pwd, age)
    # PASO 11: Ejecutar la función "EXECUTE()",
    #           y enviarle como PÁRAMETROS las variables "query" y "data" previamente creadas
    #           en los pasos 11 y 10,
    #           para realizar la CONSULTA SQL a la base de datos
    #
    # IMPORTANTE: Esta función permite ejecutar lenguaje SQL dentro de un archivo Python
    #               y así modificar la base de datos desde el BackEnd y no desde la interfaz de PhpMyAdmin
    cursor.execute(query, data)
```

Creando cine_database.py

PASO 3 – Parte 12

```
my_connection = self.open_connection()
# PASO 8: Crear la variable "cursor",
#         para crear un puntero que nos permita acceder a un lugar específico
#         de nuestra base de datos
cursor = my_connection.cursor()
# PASO 9: Crear la variable "query",
#         para crear la instrucción SQL que nos permita INSERTAR o CREAR un registro
#         en la base de datos
#
# IMPORTANTE: La estructura de la instrucción SQL está definida
#             y lo único que cambia son los CAMPOS y el NOMBRE DE LA TABLA
query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
# PASO 10: Crear la variable "data",
#          para almacenar los datos o información correspondiente al registro
#          que deseamos insertar en el paso anterior
#
# IMPORTANTE: los valores de las variables (EMAIL, PWD, AGE) debe OBTENERLOS
#             mediante la función GET() en la Interfaz de Tkinter y enviarlos
#             como PARÁMETROS a la función insert_db()
data = (email, pwd, age)
# PASO 11: Ejecutar la función "EXECUTE()",
#          y enviarle como PÁRAMETROS las variables "query" y "data" previamente creadas
#          en los pasos 11 y 10,
#          para realizar la CONSULTA SQL a la base de datos
#
# IMPORTANTE: Esta función permite ejecutar lenguaje SQL dentro de un archivo Python
#             y así modificar la base de datos desde el BackEnd y no desde la interfaz de PhpMyAdmin
cursor.execute(query, data)
# PASO 12: Ejecutar la función "commit()",
#          para confirmar la escritura en la base de datos
my_connection.commit()
```

Creando cine_database.py

PASO 3 – Parte 13

```
cursor = my_connection.cursor()
# PASO 9: Crear la variable "query",
#         para crear la instrucción SQL que nos permita INSERTAR o CREAR un registro
#         en la base de datos
#
# IMPORTANTE: La estructura de la instrucción SQL está definida
#             y lo único que cambia son los CAMPOS y el NOMBRE DE LA TABLA
query = "INSERT INTO tbl_usuarios(CORREO, PWD, EDAD) VALUES (%s,%s,%s)"
# PASO 10: Crear la variable "data",
#          para almacenar los datos o información correspondiente al registro
#          que deseamos insertar en el paso anterior
#
# IMPORTANTE: los valores de las variables (EMAIL, PWD, AGE) debe OBTENERLOS
#             mediante la función GET() en la Interfaz de Tkinter y enviarlos
#             como PARÁMETROS a la función insert_db()
data = (email, pwd, age)
# PASO 11: Ejecutar la función "EXECUTE()",
#          y enviarle como PÁRAMETROS las variables "query" y "data" previamente creadas
#          en los pasos 11 y 10,
#          para realizar la CONSULTA SQL a la base de datos
#
# IMPORTANTE: Esta función permite ejecutar lenguaje SQL dentro de un archivo Python
#             y así modificar la base de datos desde el BackEnd y no desde la interfaz de PhpMyAdmin
cursor.execute(query, data)
# PASO 12: Ejecutar la función "commit()",
#          para confirmar la escritura en la base de datos
my_connection.commit()
# PASO 13: Ejecutar la función "close()",
#          para cerrar la conexión a la base de datos
my_connection.close()
```

Práctica #2 - MySQL + Python 2º - CLASE

Venció ayer a las 11:20 • Se cerró ayer a las 11:20

Instrucciones

ACTIVIDAD INDIVIDUAL

Seguir las siguientes instrucciones:

1. Desarrollar en **PhpMyAdmin** una base de datos con la siguiente estructura ([ver DB_CINE_DISEÑO.pdf adjunto](#))
2. Exportar la base de datos en formato **.sql** y renombrar el archivo bajo la siguiente nomenclatura:
 - Cuenta_Nombre_DB_CINE, Ejemplo: **13_Deyanira_DB_CINE.sql**
3. Crear un archivo Python con el nombre **cine_database.py** donde debe configurar la conexión a la BASE DE DATOS previamente creada y sus respectivas funciones en base al archivo ([ver mydatabase.png adjunto](#))
4. Descargar la **INTERFAZ de Tkinter** correspondiente al archivo adjunto nombrado **ui_cine.py** para desarrollar el ejercicio de clase donde debe editarlo de tal manera que le permita insertar datos en la base de datos creada y configurar la función en base al ([ver mysql_ui_form.png adjunto](#))
5. Crear un repositorio en GITHUB y subir archivos Python:
 1. **ui_cine.py**
 2. **cine_database.py**
 3. **13_Deyanira_DB_CINE.sql**
6. Adjuntar a la plataforma el enlace del REPOSITORIO DE GITHUB con todo su trabajo realizado en clase.

PAUTA

mysql_ui_form.png

PASO 4

```
mydatabase.py M X  register_form.py X
SocialNetwork > register_form.py > frame_navbar
1  from tkinter import *
2  from tkinter import ttk
3  import mydatabase
4
5  window = Tk()
6  frame_app = Frame(window, width=400, height=600, bg="red")
7  frame_app.pack()
8
9  email = StringVar()
10 pwd = StringVar()
11 age = StringVar()
12
13 def register():
14     email = entry_email.get()
15     pwd = entry_pwd.get()
16     age = entry_age.get()
17
18     redsocial_db = mydatabase.MyDatabase()
19     redsocial_db.insert_db(email, pwd, age)
20
```


Práctica #2 - MySQL + Python 2º - CLASE

Venció ayer a las 11:20 • Se cerró ayer a las 11:20

Instrucciones

ACTIVIDAD INDIVIDUAL

Seguir las siguientes instrucciones:

1. Desarrollar en **PhpMyAdmin** una base de datos con la siguiente estructura (ver **DB_CINE_DISEÑO.pdf** adjunto)
2. Exportar la base de datos en formato **.sql** y renombrar el archivo bajo la siguiente nomenclatura:
 - Cuenta_Nombre_DB_CINE, Ejemplo: **13_Deyanira_DB_CINE.sql**
3. Crear un archivo Python con el nombre **cine_database.py** donde debe configurar la conexión a la BASE DE DATOS previamente creada y sus respectivas funciones en base al archivo (ver **mydatabase.png** adjunto)
4. Descargar la **INTERFAZ de Tkinter** correspondiente al archivo adjunto nombrado **ui_cine.py** para desarrollar el ejercicio de clase donde debe editarlo de tal manera que le permita insertar datos en la base de datos creada y configurar la función en base al (ver **mysql ui form.png** adjunto)
5. Crear un repositorio en GITHUB y subir archivos Python:
 1. **ui_cine.py**
 2. **cine_database.py**
 3. **13_Deyanira_DB_CINE.sql**
6. Adjuntar a la plataforma el enlace del REPOSITORIO DE GITHUB con todo su trabajo realizado en clase.

Tutorial GitHub

PASOS PARA SUBIR MI PROYECTO A LA NUBE

1. **git init** Convierte una carpeta en un repositorio
2. **git add .** Prepara todos los archivos dentro de la carpeta
3. **git commit -m "Mensaje"** Carga y confirma los archivos en la carpeta listos para subirlos a la nube
4. **git push -u origin master** Sube los archivos dentro del commit al repositorio en la nube previamente creado

RESULTADO ESPERADO

PASO 5

Argude01 / Connection-MySQL

Unwatch 1 Star 1 Fork 1

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

master had recent pushes 34 minutes ago

Compare & pull request

master 2 branches 0 tags

Go to file Add file Code

This branch is 8 commits ahead, 1 commit behind main.

Pull request Compare

Argude01 Tutorial agregado a los archivos mydatabase.py y register_form.py 4da79b9 35 minutes ago 8 commits

__pycache__	Tutorial agregado a los archivos mydatabase.py y register_form.py	35 minutes ago
connection.py	Formulario modificado	6 days ago
database_connection.py	Formulario modificado	6 days ago
insert_database.py	Formulario modificado	6 days ago
mydatabase.py	Tutorial agregado a los archivos mydatabase.py y register_form.py	35 minutes ago
register_form.py	Tutorial agregado a los archivos mydatabase.py y register_form.py	35 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

About

13 WALESKA DEYANRIA GUTIERREZ ARTEAGA - Programación - 12 BTP A

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)