No os perdáis mi futuro contenido, seguidme en Twitter y Youtube 😁

Ejercicios « Bases de datos SQLite

Ejercicio 1

A lo largo de estos ejercicios vamos a crear un pequeño sistema para gestionar los platos del menú de un restaurante.

Parte 1

Por ahora debes empezar creando un script llamado **restaurante.py** y dentro una función **crear_bd()** que creará una pequeña base de datos **restaurante.db** con las siguientes dos tablas:

```
CREATE TABLE categoria(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre VARCHAR(100) UNIQUE NOT NULL)
```

```
CREATE TABLE plato(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre VARCHAR(100) UNIQUE NOT NULL,
    categoria_id INTEGER NOT NULL,
    FOREIGN KEY(categoria_id) REFERENCES categoria(id))
```

Si ya existen deberá tratar la excepción y mostrar que las tablas ya existen, lo notarás porque en este caso no usamos el *IF NOT EXISTS* y eso lanzará un error. En caso contrario mostrará que se han creado correctamente.

> **Nota**
>
> La línea **FOREIGN KEY(categoria_id) REFERENCES categoria(id)** indica un tipo de clave especial (foránea), por la cual se crea una relación entre la categoría de un plato con el registro de categorías.

Llama a la función y comprueba que la base de datos se crea correctamente.

Parte 2

Crea una función llamada **agregar_categoria()** que pida al usuario un nombre de categoría y se encargue de crear la categoría en la base de datos (ten en cuenta que si ya existe dará un error, por que el nombre es UNIQUE).

Luego crea un pequeño menú de opciones dentro del script, que te de la bienvenida al sistema y te permita *Crear una categoría* o *Salir*. Puedes hacerlo en una función **mostrar_menu()**. Añade las siguientes tres categorías utilizando este menú de opciones:

- Primeros
- Segundos
- Postres

Parte 3

Crea una función llamada **agregar_plato()** que muestre al usuario las categorías disponibles y le permita escoger una (escribiendo un número).

Luego le pedirá introducir el nombre del plato y lo añadirá a la base de datos, teniendo en cuenta que la categoria del plato concuerde con el id de la categoría y que el nombre del plato no puede repetirse (no es necesario comprobar si la categoría realmente existe, en ese caso simplemente no se insertará el plato).

Agrega la nueva opción al menú de opciones y añade los siguientes platos:

- **Primeros**: Ensalada del tiempo / Zumo de tomate
- **Segundos**: Estofado de pescado / Pollo con patatas
- **Postres**: Flan con nata / Fruta del tiempo

Parte 4

Crea una función llamada **mostrar_menu()** que muestre el menú con todos los platos de forma ordenada: los primeros, los segundos y los postres. Optativamente puedes adornar la forma en que muestras el menú por pantalla.

Solución

restaurante.py

```
import sqlite3

def crear_bd():
    conexion = sqlite3.connect("restaurante.db")
    cursor = conexion.cursor()

    try:
        cursor.execute('''CREATE TABLE categoria(
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nombre VARCHAR(100) UNIQUE NOT NULL)''')
    except sqlite3.OperationalError:
        print("La tabla de Categorías ya existe.")
    else:
        print("La tabla de Categorías se ha creado
correctamente.")

    try:
        cursor.execute('''CREATE TABLE plato(
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nombre VARCHAR(100) UNIQUE NOT NULL,
                categoria_id INTEGER NOT NULL,
                FOREIGN KEY(categoria_id) REFERENCES
categoria(id))''')
    except sqlite3.OperationalError:
        print("La tabla de Platos ya existe.")
    else:
        print("La tabla de Platos se ha creado correctamente.")


    conexion.close()


def agregar_categoria():
    categoria = input("¿Nombre de la nueva categoría?\n> ")

    conexion = sqlite3.connect("restaurante.db")
    cursor = conexion.cursor()

    try:
        cursor.execute("INSERT INTO categoria VALUES (null,
'{}')".format(
            categoria) )
```

```
    except sqlite3.IntegrityError:
        print("La categoría '{}' ya existe.".format(categoria))
    else:
        print("Categoría '{}' creada
correctamente.".format(categoria))

    conexion.commit()
    conexion.close()


def agregar_plato():

    conexion = sqlite3.connect("restaurante.db")
    cursor = conexion.cursor()

    categorias = cursor.execute("SELECT * FROM
categoria").fetchall()
    print("Selecciona una categoría para añadir el plato:")
    for categoria in categorias:
        print("[{}] {}".format(categoria[0], categoria[1]))

    categoria_usuario = int(input("> "))

    plato = input("¿Nombre del nuevo plato?\n> ")

    try:
        cursor.execute("INSERT INTO plato VALUES (null, '{}',
{})".format(
            plato, categoria_usuario) )
    except sqlite3.IntegrityError:
        print("El plato '{}' ya existe.".format(plato))
    else:
        print("Plato '{}' creado correctamente.".format(plato))


    conexion.commit()
    conexion.close()


def mostrar_menu():

    conexion = sqlite3.connect("restaurante.db")
    cursor = conexion.cursor()

    categorias = cursor.execute("SELECT * FROM
categoria").fetchall()
    for categoria in categorias:
        print(categoria[1])
        platos = cursor.execute(
```

```
            "SELECT * FROM plato WHERE categoria_id={}".format(
                categoria[0])).fetchall()
        for plato in platos:
            print("\t{}".format(plato[1]))

    conexion.close()


# Crear la base de datos
crear_bd()

# Menú de opciones del programa
while True:
    print("\nBienvenido al gestor del restaurante!")
    opcion = input(
        "\nIntroduce una opción:" \
        "\n[1] Agregar una categoría" \
        "\n[2] Agregar un plato" \
        "\n[3] Mostrar el menú" \
        "\n[4] Salir del programa\n\n> ")

    if opcion == "1":
        agregar_categoria()

    elif opcion == "2":
        agregar_plato()

    elif opcion == "3":
        mostrar_menu()

    elif opcion == "4":
        print("Nos vemos!")
        break

    else:
        print("Opción incorrecta")
```

Ejercicio 2

En este ejercicios debes crear una interfaz gráfica con tkinter (menu.py) que muestre de forma elegante el menú del restaurante.

- Tú eliges el nombre del restaurante y el precio del menú, así como las tipografías, colores, adornos y tamaño de la ventana.

- El único requisito es que el programa se conectará a la base de datos para buscar la lista categorías y platos.

Solución

menu.py

```
import sqlite3
from tkinter import *

# Configuración de la raíz
root = Tk()
root.title("Bar Don Costa - Menú")
root.resizable(0,0)
root.config(bd=25, relief="sunken")

Label(root, text="   Bar Don Costa   ", fg="darkgreen", font=(
    "Times New Roman",28,"bold italic")).pack()
Label(root, text="Menú del día", fg="green", font=(
    "Times New Roman",24,"bold italic")).pack()

# Separación de títulos y categorias
Label(root, text="").pack()

conexion = sqlite3.connect("restaurante.db")
cursor = conexion.cursor()

# Buscar las categorías y platos de la bd
categorias = cursor.execute("SELECT * FROM
categoria").fetchall()
for categoria in categorias:
    Label(root, text=categoria[1], fg="black", font=(
        "Times New Roman",20,"bold italic")).pack()

    platos = cursor.execute(
        "SELECT * FROM plato WHERE categoria_id={}".format(
            categoria[0])).fetchall()
    for plato in platos:
        Label(root, text=plato[1], fg="gray", font=(
            "Verdana",15,"italic")).pack()

    # Separación entre categorias
    Label(root, text="").pack()

conexion.close()
```

```
# Precio del menú
Label(root, text="12€ (IVA incl.)", fg="darkgreen", font=(
    "Times New Roman",20,"bold italic")).pack(side="right")

# Finalmente ejecutamos el bucle
root.mainloop()
```

Última edición: 5 de Octubre de 2018