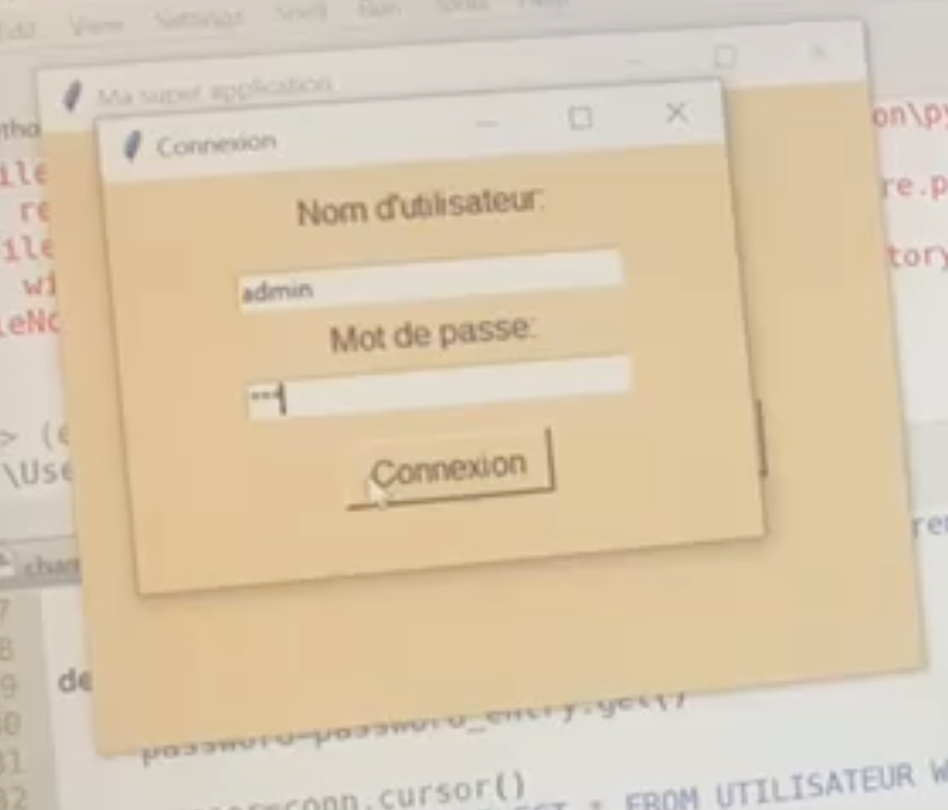
**FAIT PAR :MAROUA CHEGRI**

**KENZA OUTTASS**

**ABDOU RAHAMAN BENSALAK**

***Le code fourni est une application simple utilisant la bibliothèque Tkinter pour créer une interface utilisateur. L'application permet à l'utilisateur de se connecter ou de s'inscrire en utilisant un nom d'utilisateur et un mot de passe.***

******

POUR LE PREMIER FILE TEST1.PY

1. Importation des bibliothèques nécessaires :
   * **tkinter** : Pour créer l'interface graphique.
   * **sqlite3** : Pour gérer la base de données SQLite.
   * **messagebox** : Pour afficher des messages d'information.
   * **sys** : Pour ajouter un chemin spécifique au module de la chambre (dans un répertoire particulier).
2. import tkinter as tk  
   import sqlite3  
   from tkinter import messagebox  
   import sys  
   sys.path.append('C:\\Users\\Imane\\Downloads\\test1\\test1')  
     
   import chambre

Établissement de la connexion à la base de données :

* + Le fichier de base de données est défini comme "database.db".
  + La connexion à la base de données est établie à l'aide de **sqlite3.connect()** et stockée dans la variable **conn**.

1. database="database.db"  
   conn=sqlite3.connect(database)
2. Création de la table "Utilisateur" :
   * La table est créée avec trois colonnes : "IdUtilisateur" (clé primaire auto-incrémentée), "NomUtilisateur" (chaîne de caractères) et "password" (chaîne de caractères).
   * La requête de création de table est exécutée à l'aide de **cursor.execute()**.
3. cursor = conn.cursor()  
   cursor.execute("""CREATE TABLE IF NOT EXISTS Utilisateur (  
    IdUtilisateur INTEGER PRIMARY KEY AUTOINCREMENT,  
    NomUtilisateur NVARCHAR(50) NOT NULL,  
    password NVARCHAR(50) NOT NULL)  
     
    """)
4. Définition des fonctions pour les fenêtres de connexion et d'inscription :
   * **open\_login\_window()** : Ouvre une nouvelle fenêtre pour la connexion. La fenêtre contient des étiquettes pour le nom d'utilisateur et le mot de passe, ainsi que des champs d'entrée et un bouton de connexion.
5. def open\_login\_window():  
    login\_window = tk.Toplevel(root)  
    login\_window.title("Connexion")  
    login\_window.geometry("300x200")  
    login\_window.configure(bg="#F5DEB3")  
    username\_label = tk.Label(login\_window, text="Nom d'utilisateur:", bg="#F5DEB3", font=("Arial", 12))  
    username\_label.pack(pady=10)  
    username\_entry = tk.Entry(login\_window, width=30)  
    username\_entry.pack(pady=5)  
    password\_label = tk.Label(login\_window, text="Mot de passe:", bg="#F5DEB3", font=("Arial", 12))  
    password\_label.pack()  
    password\_entry = tk.Entry(login\_window, width=30, show="\*")  
    password\_entry.pack(pady=5)  
    login\_button = tk.Button(login\_window, text="Connexion", bg="#F5DEB3", font=("Arial", 12), width=10, command=lambda : login(username\_entry, password\_entry))  
    login\_button.pack(pady=10)
   * **open\_register\_window()** : Ouvre une nouvelle fenêtre pour l'inscription. La fenêtre contient des étiquettes pour le nom d'utilisateur et le mot de passe, ainsi que des champs d'entrée et un bouton d'inscription.
6. def open\_register\_window():  
    register\_window = tk.Toplevel(root)  
    register\_window.title("Inscription")  
    register\_window.geometry("300x200")  
    register\_window.configure(bg="#F5DEB3")  
    username\_label = tk.Label(register\_window, text="Nom d'utilisateur:", bg="#F5DEB3", font=("Arial", 12))  
    username\_label.pack(pady=10)  
    username\_entry = tk.Entry(register\_window, width=30)  
    username\_entry.pack(pady=5)  
    password\_label = tk.Label(register\_window, text="Mot de passe:", bg="#F5DEB3", font=("Arial", 12))  
    password\_label.pack()  
    password\_entry = tk.Entry(register\_window, width=30, show="\*")  
    password\_entry.pack(pady=5)  
    register\_button = tk.Button(register\_window, text="Inscription", bg="#F5DEB3", font=("Arial", 12), width=10 , command=lambda : register(username\_entry, password\_entry))  
    register\_button.pack(pady=10)
7. Création de la fenêtre principale de l'application :
   * Une instance de **Tk()** est créée et stockée dans la variable **root**.
   * La fenêtre est configurée avec un titre, une taille et une couleur de fond.
   * Une étiquette de titre est ajoutée à la fenêtre à l'aide de **tk.Label()** et affichée à l'aide de **label.pack()**.
   * Un cadre de boutons est ajouté à la fenêtre à l'aide de **tk.Frame()** et affiché à l'aide de **button\_frame.pack()**.
   * Deux boutons, "Connexion" et "Inscription", sont ajoutés au cadre de boutons à l'aide de **tk.Button()** et affichés à l'aide de **login\_button.pack()** et **register\_button.pack()** respectivement.
   * Les boutons sont associés à leurs fonctions respectives (**open\_login\_window()** et **open\_register\_window()**) à l'aide du paramètre **command**.
8. root = tk.Tk()  
   root.title("Ma super application")  
   root.geometry("400x300")  
   root.configure(bg="#F5DEB3")  
   title\_label = tk.Label(root, text="Bienvenue !", bg="#F5DEB3", font=("Arial", 20, "italic"))  
   title\_label.pack(pady=50)  
   button\_frame = tk.Frame(root, bg="#F5DEB3")  
   button\_frame.pack(pady=20)  
   login\_button = tk.Button(button\_frame, text="Connexion", bg="#FFFACD", font=("Arial", 14), width=10, command=open\_login\_window)  
   login\_button.pack(side=tk.RIGHT, padx=10)  
   register\_button = tk.Button(button\_frame, text="Inscription", bg="#FFFACD", font=("Arial", 14), width=10, command=open\_register\_window)  
   register\_button.pack(side=tk.LEFT, padx=10)
9. Définition des fonctions pour l'inscription et la connexion :
   * **register(username\_entry, password\_entry)** : Récupère le nom d'utilisateur et le mot de passe à partir des champs d'entrée, puis exécute une requête SQL pour insérer ces valeurs dans la table "Utilisateur". Si l'opération réussit, une boîte de dialogue d'information est affichée avec un message de réussite. Sinon, une boîte de dialogue d'information est affichée.
10. def register(username\_entry, password\_entry):  
     cursor=conn.cursor()  
     username=username\_entry.get()  
     password=password\_entry.get()  
     try:  
     cursor.execute("INSERT INTO Utilisateur(nomutilisateur, password) VALUES (? , ?)", (username, password))  
     conn.commit()  
     messagebox.showinfo("Info", "utilisateur enregistré avec succes")  
     except Exception as e:  
     messagebox.showinfo("info", f"enregistrement utilisateur echoue {str(e)}")

14.La fonction login(username\_entry, password\_entry) :

Cette fonction est appelée lorsque l'utilisateur appuie sur le bouton "Connexion" dans la fenêtre principale. Elle récupère les valeurs entrées par l'utilisateur pour le nom d'utilisateur et le mot de passe. Elle exécute une requête SQL pour récupérer les informations de l'utilisateur correspondant aux entrées de l'utilisateur. Si les informations sont trouvées, la fonction ferme la fenêtre de connexion et ouvre la fenêtre de la chambre. Sinon, une boîte de dialogue d'information est affichée pour informer l'utilisateur que le nom d'utilisateur ou le mot de passe est incorrect. La fonction utilise le module messagebox pour afficher les messages d'information.

def login(username\_entry, password\_entry):  
 username=username\_entry.get()  
 password=password\_entry.get()  
  
 cursor=conn.cursor()  
 cursor.execute("SELECT \* FROM UTILISATEUR WHERE NOMUTILISATEUR=? and password = ?", (username, password))  
 existe=cursor.fetchall()  
 if not existe:  
 messagebox.showinfo("info", "username et password errones")  
 else:  
 root.destroy()

15. La fonction chambre\_window() :

Cette fonction ouvre une nouvelle fenêtre pour la chambre. La fonction utilise le module sys pour ajouter un chemin spécifique au module de la chambre (dans un répertoire particulier). La fenêtre contient une étiquette de titre, un bouton "Déconnexion" et un message de bienvenue personnalisé contenant le nom d'utilisateur.

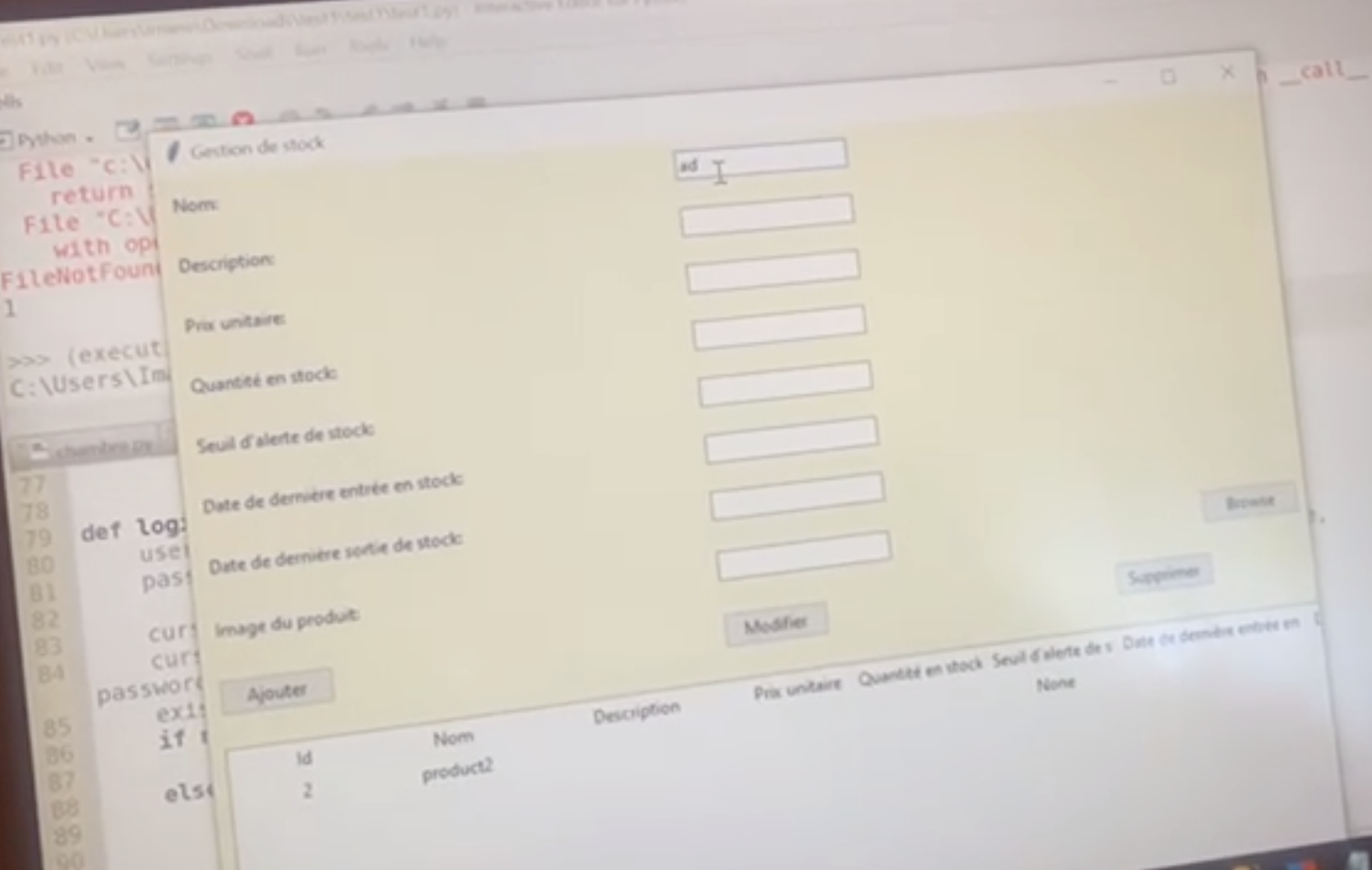
chambre.chambre\_window()

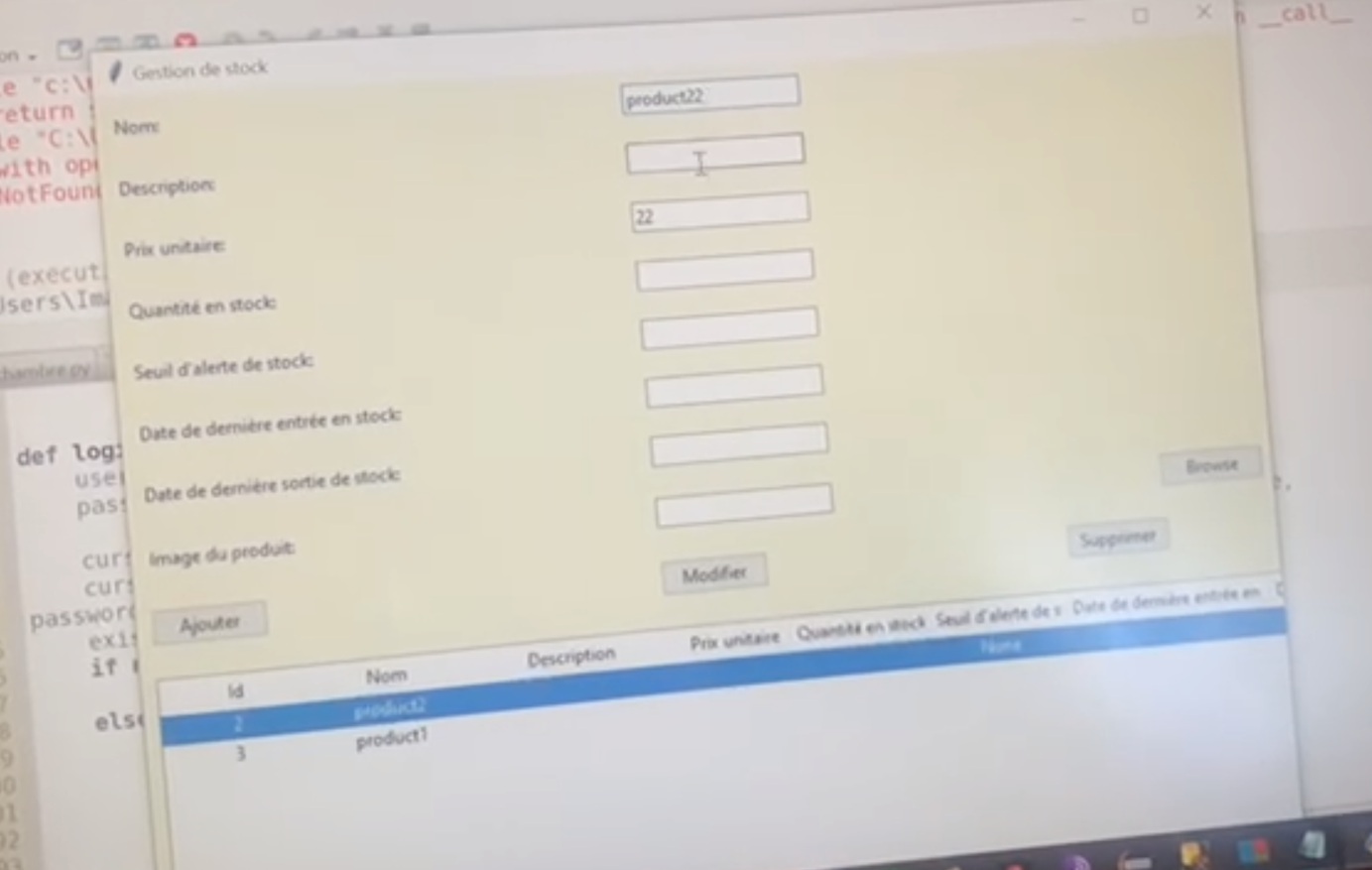
Haut du formulaire

POUR LE DESCIEME FILE CHAMBRE.PY

Bas du formulaire

**L'application permet d'ajouter, de modifier, de supprimer et de rechercher des produits dans un stock. Voici un aperçu du code :**

****

Importation des bibliothèques nécessaires :

* tkinter : Pour créer l'interface graphique.
* ttk : Pour les widgets améliorés de Tkinter.
* sqlite3 : Pour gérer la base de données SQLite.
* messagebox : Pour afficher des messages d'information.
* filedialog : Pour ouvrir une boîte de dialogue de sélection de fichier.
* PIL : Pour travailler avec des images.
* import tkinter as tk  
  from tkinter import ttk  
  import sqlite3  
  from tkinter import messagebox  
  from tkinter import filedialog  
  from PIL import Image, ImageTk

Connexion à la base de données :

* Le fichier de base de données est défini comme "stock.db".
* La connexion à la base de données est établie à l'aide de sqlite3.connect() et stockée dans la variable conn.
* Une table "PRODUIT" est créée dans la base de données pour stocker les informations sur les produits.
* database="database.db"  
  conn=sqlite3.connect("stock.db")

Fenêtre principale de l'application :

* Une instance de Tk() est créée et stockée dans la variable root.
* La fenêtre est configurée avec un titre, une taille et une couleur de fond.
* Des étiquettes et des champs d'entrée sont créés pour les attributs du produit tels que le nom, la description, le prix, la quantité, etc.
* Un bouton de parcourir est ajouté pour sélectionner une image du produit.
* Des boutons pour ajouter, modifier, supprimer et rechercher des produits sont ajoutés.
* Un Treeview est utilisé pour afficher les produits existants.

def chambre\_window():  
 # création de la fenêtre principale  
 root = tk.Tk()  
 root.title("Gestion de stock")  
 root.geometry("800x600")  
 root.config(bg="#f5f5dc")  
  
 # création des labels pour les attributs du produit  
 name\_label = ttk.Label(root, text="Nom:")  
 name\_label.grid(column=0, row=0, padx=10, pady=10, sticky="w")  
  
 description\_label = ttk.Label(root, text="Description:")  
 description\_label.grid(column=0, row=1, padx=10, pady=10, sticky="w")  
  
 price\_label = ttk.Label(root, text="Prix unitaire:")  
 price\_label.grid(column=0, row=2, padx=10, pady=10, sticky="w")  
  
 quantity\_label = ttk.Label(root, text="Quantité en stock:")  
 quantity\_label.grid(column=0, row=3, padx=10, pady=10, sticky="w")  
  
 alert\_label = ttk.Label(root, text="Seuil d'alerte de stock:")  
 alert\_label.grid(column=0, row=4, padx=10, pady=10, sticky="w")  
  
 last\_entry\_label = ttk.Label(root, text="Date de dernière entrée en stock:")  
 last\_entry\_label.grid(column=0, row=5, padx=10, pady=10, sticky="w")  
  
 last\_exit\_label = ttk.Label(root, text="Date de dernière sortie de stock:")  
 last\_exit\_label.grid(column=0, row=6, padx=10, pady=10, sticky="w")  
  
 image\_label = ttk.Label(root, text="Image du produit:")  
 image\_label.grid(column=0, row=7, padx=10, pady=10, sticky="w")  
  
 # création des entrées pour les attributs du produit  
 name\_entry = ttk.Entry(root)  
 name\_entry.grid(column=1, row=0, padx=10, pady=10, sticky="w")  
 description\_entry = ttk.Entry(root)  
 description\_entry.grid(column=1, row=1, padx=10, pady=10, sticky="w")  
 price\_entry = ttk.Entry(root)  
 price\_entry.grid(column=1, row=2, padx=10, pady=10, sticky="w")  
 quantity\_entry = ttk.Entry(root)  
 quantity\_entry.grid(column=1, row=3, padx=10, pady=10, sticky="w")  
 alert\_entry = ttk.Entry(root)  
 alert\_entry.grid(column=1, row=4, padx=10, pady=10, sticky="w")  
 last\_entry\_entry = ttk.Entry(root)  
 last\_entry\_entry.grid(column=1, row=5, padx=10, pady=10, sticky="w")  
 last\_exit\_entry = ttk.Entry(root)  
 last\_exit\_entry.grid(column=1, row=6, padx=10, pady=10, sticky="w")  
 image\_entry = ttk.Entry(root)  
 image\_entry.grid(column=1, row=7, padx=10, pady=10, sticky="w")  
 def browse\_image():  
 filename = filedialog.askopenfilename()  
 image\_entry.delete(0, "end")  
 image\_entry.insert(0, filename)  
  
 browse\_button = ttk.Button(root, text="Browse", command=browse\_image)  
 browse\_button.grid(column=2, row=7, padx=10, pady=10)  
  
 search\_entry = ttk.Entry(root,width=20, font=("bold", 14))

Fonctions pour ajouter, modifier, supprimer et rechercher des produits :

* add\_product() : Récupère les valeurs des champs d'entrée, insère les données dans la base de données et ajoute le produit au Treeview.
* modify\_product() : Récupère le produit sélectionné dans le Treeview, met à jour les valeurs dans la base de données et dans le Treeview.
* delete\_product() : Supprime le produit sélectionné de la base de données et du Treeview.
* search\_product() : Recherche un produit dans le Treeview en fonction d'un mot-clé saisi.

def add\_product():  
 name = name\_entry.get()  
 description = description\_entry.get()  
 price = price\_entry.get()  
 quantity = quantity\_entry.get()  
 alert = alert\_entry.get()  
 last\_entry = last\_entry\_entry.get()  
 last\_exit = last\_exit\_entry.get()  
 image = image\_entry.get()  
 with open(image, "rb") as f:  
 image\_data = f.read()  
 try:  
 cursor=conn.cursor()  
 cursor.execute("""INSERT INTO PRODUIT(NomProduit, DescriptionProduit, PrixUnitaire, StockActuel,Sdalert, DDE, DDS, imgp) VALUES(?,?,?,?,?,?,?,?)""",(name,description,price, quantity,alert,last\_entry,last\_exit,image\_data))  
 conn.commit()  
 id = cursor.lastrowid  
 item = (id,name, description, price, quantity, alert, last\_entry, last\_exit, image)  
 product\_list.insert("", "end", values=item)  
 name\_entry.delete(0, "end")  
 description\_entry.delete(0, "end")  
 price\_entry.delete(0, "end")  
 quantity\_entry.delete(0, "end")  
 alert\_entry.delete(0, "end")  
 last\_entry\_entry.delete(0, "end")  
 last\_exit\_entry.delete(0, "end")  
 image\_entry.delete(0, "end")  
 messagebox.showinfo("Success", "Record added successfully.")  
  
 except:  
 messagebox.showinfo("Info", "Données saisies erronées ou non completes!")  
  
  
  
  
  
# fonction pour modifier un produit sélectionné dans la liste  
def modify\_product():  
 selected\_item = product\_list.selection()[0]  
 id = int(product\_list.item(selected\_item, "values")[0])  
 selected = product\_list.focus()  
 product\_list.item(selected, values=(id, name\_entry.get(), description\_entry.get(), price\_entry.get(), quantity\_entry.get(), alert\_entry.get(), last\_entry\_entry.get(), last\_exit\_entry.get(), image\_entry.get()))  
 with open(image\_entry.get(), "rb") as f:  
 image\_data = f.read()  
 cursor=conn.cursor()  
 try:  
 conn.execute("UPDATE produit SET NomProduit= ?, DescriptionProduit = ? , PrixUnitaire = ?, StockActuel = ? , Sdalert = ?, dde = ?, dds= ?, imgp = ? WHERE IdProduit=? ",(name\_entry.get(), description\_entry.get(), price\_entry.get(), quantity\_entry.get(), alert\_entry.get(), last\_entry\_entry.get(), last\_exit\_entry.get(), image\_data,id))  
 conn.commit()  
 messagebox.showinfo("Success", "Record modified successfully.")  
  
 except:  
 messagebox.showinfo("Info", "Données saisies erronées ou non completes!")  
  
# fonction pour supprimer un produit sélectionné dans la liste  
def delete\_product():  
 selected\_item = product\_list.selection()[0]  
 id = int(product\_list.item(selected\_item, "values")[0])  
 print(id)  
 delete\_query = "DELETE FROM produit WHERE IdProduit = ?"  
 try:  
 conn.execute(delete\_query, ((id),))  
 conn.commit()  
 product\_list.delete(selected\_item)  
 messagebox.showinfo("Success", "Record deleted successfully.")  
 except Exception as e:  
 messagebox.showerror("Error", f"Error deleting record: {str(e)}")  
  
  
# fonction pour rechercher un produit dans la liste  
def search\_product():  
 keyword = search\_entry.get()  
 results = product\_list.get\_children()  
 for result in results:  
 values = product\_list.item(result)["values"]  
 if keyword.lower() in str(values).lower():  
 product\_list.selection\_set(result)  
 product\_list.focus(result)  
 else:  
 product\_list.selection\_remove(result)

Ajout des fonctions aux boutons :

* Les fonctions définies ci-dessus sont associées aux boutons correspondants pour effectuer les actions respectives.

# ajout des fonctions aux boutons  
add\_button.config(command=add\_product)  
modify\_button.config(command=modify\_product)  
delete\_button.config(command=delete\_product)  
search\_button.config(command=search\_product)

Exécution de la fenêtre principale de l'application :

* La fonction chambre\_window() est appelée pour afficher la fenêtre principale de l'application.
* La boucle principale tkinter (root.mainloop()) est exécutée pour afficher l'interface utilisateur et gérer les événements.
* root.mainloop()

***MERCI***