Benlamrabet Abdelwadoud

Belahrech Abderrahmane

Boulafrouh Ayoub

```python
import sqlite3
from tkinter import *
from tkinter import ttk

```

Les deux premières lignes importent les modules sqlite3 et ttk de tkinter pour utiliser la base de données SQLite et créer des widgets améliorés tels que TreeView.

```python
root = Tk()
root.title("Gestion de stock")
root.geometry("1080x720")
my_tree = ttk.Treeview(root)
root.config(background="#1b87d2")
storeName = "Gestion de stock"
```

Cette partie crée une fenêtre de l'interface graphique avec un titre, une taille et une couleur de fond. Un TreeView est également créé pour afficher les données stockées.

```python
def reverse(tuples):
    new_tup = tuples[::-1]
    return new_tup
```

Cette fonction renvoie le tuple inversé.

```
def insert( id, name, price, quantity):
    conn = sqlite3.connect("data.db")
    cursor = conn.cursor()

    cursor.execute("""CREATE TABLE IF NOT EXISTS
    inventory(itemId TEXT, itemName TEXT, itemPrice TEXT, itemQuantity TEXT)""")

    cursor.execute("INSERT INTO inventory VALUES ('" + str(id) + "','" + str(name) + "','" + str(price) + "','" + str(quantity) + "')")
    conn.commit()
```

Cette fonction permet d'insérer des données dans la base de données SQLite. Elle crée une connexion à la base de données, exécute une requête CREATE TABLE pour créer une table inventory si elle n'existe pas déjà, puis insère les valeurs dans la table.

```
def delete(data):
    conn = sqlite3.connect("data.db")
    cursor = conn.cursor()

    cursor.execute("""CREATE TABLE IF NOT EXISTS
        inventory(itemId TEXT, itemName TEXT, itemPrice TEXT, itemQuantity TEXT)""")

    cursor.execute("DELETE FROM inventory WHERE itemId = '" + str(data) + "'")
    conn.commit()
```

Cette fonction permet de supprimer les données de la table. Elle crée une connexion à la base de données, exécute une requête CREATE TABLE pour créer une table inventory si elle n'existe pas déjà, puis supprime les données de la table.

```
def update(id, name, price, quantity,  idName):
    conn = sqlite3.connect("data.db")
    cursor = conn.cursor()

    cursor.execute("""CREATE TABLE IF NOT EXISTS
        inventory(itemId TEXT, itemName TEXT, itemPrice TEXT, itemQuantity TEXT)""")

    cursor.execute("UPDATE inventory SET itemId = '" + str(id) + "', itemName = '" + str(name) + "', itemPrice = '" + str(price) + "', itemQuantity = '" + str(quantity) +
    conn.commit()
```

Cette fonction permet de mettre à jour les données dans la table. Elle crée une connexion à la base de données, exécute une requête CREATE TABLE pour créer une table inventory si elle n'existe pas déjà, puis met à jour les données dans la table.

```python
def read():
    conn = sqlite3.connect("data.db")
    cursor = conn.cursor()

    cursor.execute("""CREATE TABLE IF NOT EXISTS
        inventory(itemId TEXT, itemName TEXT, itemPrice TEXT, itemQuantity TEXT)""")

    cursor.execute("SELECT * FROM inventory")
    results = cursor.fetchall()
    conn.commit()
    return results
```

Cette fonction lit les données de la table. Elle crée une connexion à la base de données, exécute une requête CREATE TABLE pour créer une table inventory si elle n'existe pas déjà, puis sélectionne toutes les données de la table.

```python
def insert_data():
    itemId = str(entryId.get())
    itemName = str(entryName.get())
    itemPrice = str(entryPrice.get())
    itemQuantity = str(entryQuantity.get())
    if itemId == "" or itemName == " ":
        print("Error Inserting Id")
    if itemName == "" or itemName == " ":
        print("Error Inserting Name")
    if itemPrice == "" or itemPrice == " ":
        print("Error Inserting Price")
    if itemQuantity == "" or itemQuantity == " ":
        print("Error Inserting Quantity")
    else:
        insert(str(itemId), str(itemName), str(itemPrice), str(itemQuantity))

    for data in my_tree.get_children():
        my_tree.delete(data)

    for result in reverse(read()):
        my_tree.insert(parent='', index='end', iid=result, text="", values=(result), tag="orow")

    my_tree.tag_configure('orow', background='#EEEEEE')
    my_tree.grid(row=1, column=5, columnspan=4, rowspan=5, padx=10, pady=10)
```

Cette fonction est appelée lorsque l'utilisateur appuie sur le bouton d'insertion de données. Elle obtient les valeurs saisies par l'utilisateur, vérifie si elles sont valides, puis appelle la fonction insert pour insérer les données dans la table. Elle affiche également les données stockées dans le TreeView.

```
def delete_data():
    selected_item = my_tree.selection()[0]
    deleteData = str(my_tree.item(selected_item)['values'][0])
    delete(deleteData)

    for data in my_tree.get_children():
        my_tree.delete(data)

    for result in reverse(read()):
        my_tree.insert(parent='', index='end', iid=result, text="", values=(result), tag="orow")

    my_tree.tag_configure('orow', background='#EEEEEE')
    my_tree.grid(row=1, column=5, columnspan=4, rowspan=5, padx=10, pady=10)
```

Cette fonction est appelée lorsque l'utilisateur appuie sur le bouton de suppression de données. Elle obtient l'élément sélectionné dans le TreeView, obtient l'ID correspondant et appelle la fonction delete pour supprimer les données de la table. Elle affiche également les données stockées dans le TreeView.

```
def update_data():
    selected_item = my_tree.selection()[0]
    update_name = my_tree.item(selected_item)['values'][0]
    update(entryId.get(), entryName.get(), entryPrice.get(), entryQuantity.get(), update_name)

    for data in my_tree.get_children():
        my_tree.delete(data)

    for result in reverse(read()):
        my_tree.insert(parent='', index='end', iid=result, text="", values=(result), tag="orow")

    my_tree.tag_configure('orow', background='#EEEEEE')
    my_tree.grid(row=1, column=5, columnspan=4, rowspan=5, padx=10, pady=10)
```

Cette fonction est appelée lorsque l'utilisateur appuie sur le bouton de mise à jour de données. Elle obtient l'élément sélectionné dans le TreeView, obtient l'ID correspondant et appelle la fonction update pour mettre à jour les données dans la table. Elle affiche également les données stockées dans le TreeView.

```
    titleLabel = Label(root, text=storeName, font=('Arial bold', 30),background='#1b87d2', bd=2)
    titleLabel.grid(row=0, column=0, columnspan=8, padx=20, pady=20)

    idLabel = Label(root, text="ID", font=('Arial bold', 15),background='#1b87d2')
    nameLabel = Label(root, text="Name", font=('Arial bold', 15),background='#1b87d2')
    priceLabel = Label(root, text="Price", font=('Arial bold', 15),background='#1b87d2')
    quantityLabel = Label(root, text="Quantity", font=('Arial bold', 15),background='#1b87d2')
    idLabel.grid(row=1, column=0, padx=10, pady=10)
    nameLabel.grid(row=2, column=0, padx=10, pady=10)
    priceLabel.grid(row=3, column=0, padx=10, pady=10)
    quantityLabel.grid(row=4, column=0, padx=10, pady=10)

    entryId = Entry(root, width=25, bd=5, font=('Arial bold', 15))
    entryName = Entry(root, width=25, bd=5, font=('Arial bold', 15))
    entryPrice = Entry(root, width=25, bd=5, font=('Arial bold', 15))
    entryQuantity = Entry(root, width=25, bd=5, font=('Arial bold', 15))
    entryId.grid(row=1, column=1, columnspan=3, padx=5, pady=5)
    entryName.grid(row=2, column=1, columnspan=3, padx=5, pady=5)
    entryPrice.grid(row=3, column=1, columnspan=3, padx=5, pady=5)
    entryQuantity.grid(row=4, column=1, columnspan=3, padx=5, pady=5)
```

Cette partie crée des libellés pour les différents champs de données et les affiche dans la fenêtre de l'interface graphique.

```
style = ttk.Style()

style.map("Custom.TButton",
    foreground=[('pressed', 'black'), ('active', 'red')],
    background=[('pressed', 'blue'), ('active', '#3c3c3c')])

style.configure("Custom.TButton", font=('Arial', 15), padding=5, width=8)

buttonEnter = ttk.Button(
    root, text="Enter",
    style='Custom.TButton', command=insert_data)
buttonEnter.grid(row=5, column=1, padx=5, pady=5)

buttonUpdate = ttk.Button(
    root, text="Update",
    style='Custom.TButton', command=update_data)
buttonUpdate.grid(row=5, column=2, padx=5, pady=5)

buttonDelete = ttk.Button(
    root, text="Delete",
    style='Custom.TButton', command=delete_data)
buttonDelete.grid(row=5, column=3, padx=5, pady=5)
```

Cette partie configure un style personnalisé pour les boutons en utilisant ttk (themed tk). Les options de style sont définies à l'aide de la méthode `map`, qui définit les couleurs de premier plan et d'arrière-plan pour différents états de bouton, tels que pressé ou actif. Ensuite, le style est configuré avec une police, un rembourrage et une largeur donnée, et trois boutons sont créés avec ce style personnalisé et placés dans la grille en utilisant la méthode `grid`. Les boutons ont également une commande associée à chaque bouton.

```
# style = ttk.Style()
style.configure("Treeview.Heading", font=('Arial bold', 15))

my_tree['columns'] = ("ID", "Name", "Price", "Quantity")
my_tree.column("#0", width=0, stretch=NO)
my_tree.column("ID", anchor=W, width=100)
my_tree.column("Name", anchor=W, width=200)
my_tree.column("Price", anchor=W, width=150)
my_tree.column("Quantity", anchor=W, width=150)
my_tree.heading("ID", text="ID", anchor=W)
my_tree.heading("Name", text="Name", anchor=W)
my_tree.heading("Price", text="Price", anchor=W)
my_tree.heading("Quantity", text="Quantity", anchor=W)

for data in my_tree.get_children():
    my_tree.delete(data)

for result in reverse(read()):
    my_tree.insert(parent='', index='end', iid=0, text="", values=(result), tag="orow")

my_tree.tag_configure('orow', background='#EEEEEE', font=('Arial bold', 15))
my_tree.grid(row=1, column=5, columnspan=4, rowspan=5, padx=10, pady=10)

root.mainloop()
```

Cette partie est responsable de la configuration et de la création de la table (treeview) dans laquelle nous stockerons les données. Elle configure également l'apparence de la table en modifiant la police et la taille du texte, la largeur et l'alignement des colonnes, ainsi que la couleur de fond et la police des lignes de données. Les données sont insérées dans la table en appelant la fonction read() et en les insérant dans la table à l'aide de la méthode insert(). Finalement, la méthode mainloop() est utilisée pour afficher la fenêtre à l'écran.

En résumé, le code crée une interface graphique pour permettre à l'utilisateur d'insérer, de supprimer et de mettre à jour des données stockées dans une base de données SQLite. Les données sont affichées dans un TreeView.