FORMATION PYTHON

DEMO SUR L'UTILISATION DE TKINTER

DEMO TKINTER : CRUD

1) Objectif et interface à obtenir

L'objectif de cette DEMO est de réaliser l'interaction d'un programme Python avec une base de données SQLite, en proposant les fonctionnalités de CRUD.

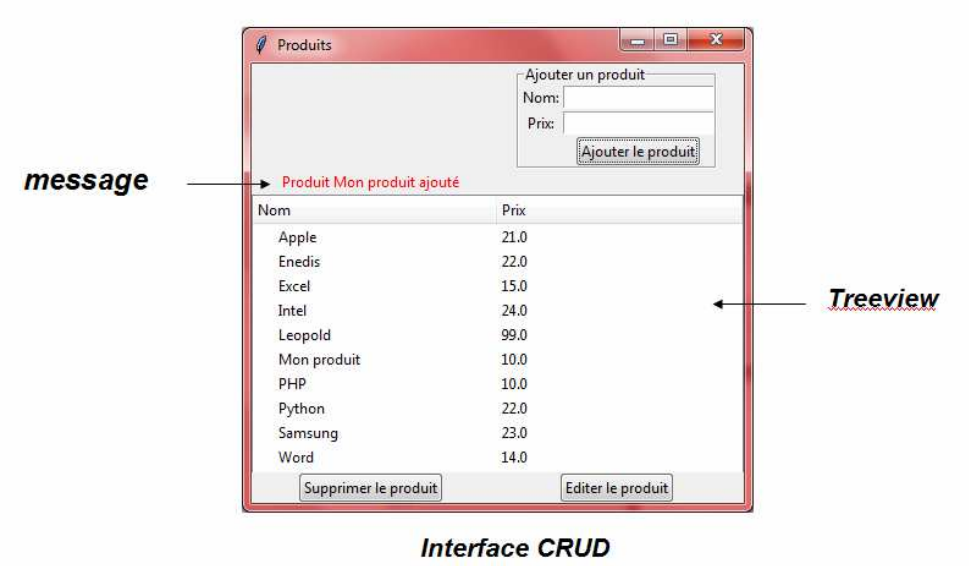
2) Créer une BDD et une table SQLite

Questions :
+ Créer la base de données SQLite nommée database.db
+ Créer la table Produits(id, nom, prix)
+ Remplir la table de quelques produits

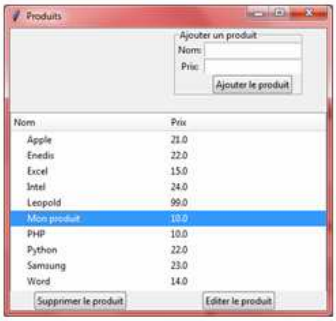
```
CREATE TABLE produit(
        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        nom TEXT,
        prix REAL
);
```

3) Présentation de l'interface de l'application

Cette interface comprend :
- Un Treeview (liste) pour afficher les données
- Un formulaire pour ajouter une donnée



Interface CRUD



Produit sélectionné

Editer le produit

```
# la fenêtre d'édition
self.edit_wind = Toplevel()
self.edit_wind.title('Editer le produit')
```

4) Créer le fichier produit.py

5) Importer les librairies

```
from tkinter import *
from tkinter import ttk
import sqlite3  :
```

6) Créer la classe Product

Attribut : db_name = 'database.sqlite'

Méthodes :

 # Le constructeur

```
def __init__(self, wind):
```

 # Connexion à la base de données (retourne un recordset)

```
def getResult(self, query, parameters=()):
```

 # Méthode affichant les données à la base de données

```
# Cette méthode utilise la méthode précédente.
# Utiliser cette méthode dans le constructeur pour
# afficher les données sur l'interface.
def showRecords(self):
```

 # Validation des saisies

```
def validation(self):
```

 # Ajouter la saisie à la base

```
# Si les données sont valides, ajoute le produit saisi
# Si le produit est ajouté, afficher 1 message,
# sinon afficher 1 message différent
# Afficher les données
# Active dans le constructeur la commande qui ajoute
```

```
# le produit
        def addProduct(self):

# Supprimer les données
# Appeler ds cette méthode getResult() pour afficher
# les données
        def deleteProduct(self):

# Modifier les données : la fenêtre de modification
        def editProduct(self):

# Modifier les données : le traitement
        def editRecord(self, nouveau_nom, nom, nouveau_prix,
```

7) Ecrire le programme principal

```
# Programme principal
# Le  fichier étant un module, on ne peux pas l'executer.
# Utiliser if __name__ == '__main__': pour exécuter le module
# et pouvoir appeler des fonctions par defaut lorsqu'on lance
# le script.
# Le module devient un script à part entière .
        if __name__ == '__main__':
                wind = Tk()
                application = Product(wind)
                wind.mainloop()
```