Projet Csv – sql- en python

Partie I

Choisir sur le site date.gouv.fr des données que vous pourrez insérer dans une base de données. Un travail de modélisation de ces données afin de le transformer en base de données sera à faire.

Vous ferez valider cette base par le professeur.

Partie 2

La création des tables et l'insertion des données se fera à l'aide d'un programme python en utilisant le module **sqlite3 (**voir Annexe1)

Partie 3

Une interface graphique permettra d'interroger la base en saisissant une ou des valeurs. Vous pourrez utiliser Tkinter ou pygame au choix (voir Annexe 2). Le programme python permettra d'exécuter les requêtes SQL et d'afficher les résultats.

Travail à rendre :

1) Un document spécifiant précisément le travail fait par chaque élève ainsi que les difficultés rencontrées, les réussites et les échecs.
2) Le fichier csv choisi (ou un lien permettant de le retrouver).
3) La base de données créée.
4) Le programme python ayant permis la création de la base ainsi que l'interrogation des données et l'affichage des réponses.

ANNEXE 1

Notre SGBD sera toujours SQLite : le module python que nous utiliserons se nomme **sqlite3**.

```
import sqlite3
```

Le module étant importé, nous devons réaliser deux actions pour pouvoir commencer à utiliser notre base :

- ouvrir le fichier de base de données
- créer un curseur

Le *curseur* est un objet python offrant des méthodes pour exécuter des requêtes et récupérer le ou les résultats de ces requêtes.

```
connexion = sqlite3.connect("nomdelabase.db")
```

```
curseur = bdd.cursor()
```

Le principe

Reste ensuite à exécuter notre première requête. Pour cela, nous utiliserons la méthode **execute()** du curseur, la requête étant une chaîne de caractères passée en paramètre.
Création de table

```
# Exécution unique
curseur.execute("""CREATE TABLE IF NOT EXISTS scores(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    pseudo TEXT,
    valeur INTEGER
)""")
```

Insertion de tuples

```
donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
# Exécutions multiples
for donnee in donnees:
    curseur.execute("INSERT INTO scores (pseudo, valeur) VALUES (?, ?)", donnee)
connexion.commit() # Ne pas oublier de valider les modifications
```

OU

```
donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
# Exécutions multiples
curseur.executemany("INSERT INTO scores (pseudo, valeur) VALUES (?, ?)", donnees)
connexion.commit() # Ne pas oublier de valider les modifications
```

```
requete = "SELECT * FROM Albums;"
```

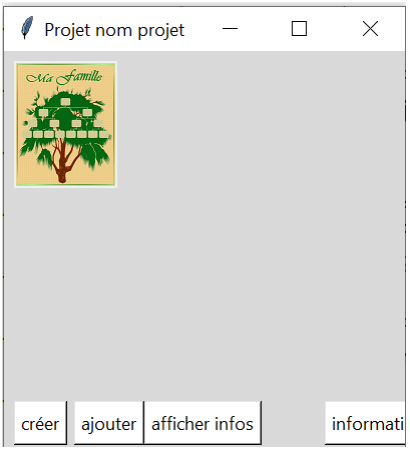
```
curseur.execute(requete)
```

Pour Finir Notre travail sur la BDD exemple est à présent terminé. Afin de fermer le fichier proprement et de s'assurer que les données saisies seront bien inscrites dans le fichier, il faut *impérativement* appeler la méthode **close()** sur l'objet *bdd* :

```
connexion.commit()
```

```
connexion.close()
```

ANNEXE 2




```
from tkinter import *
class App():
    def __init__(self, fen):
        ''' Cette méthode dit contructeur permet de créer une instance de App'''
        self.fen=fen
        self.fen.geometry('400x400')
        self.fen.configure(bg="#d9d9d9")
        self.fen.title("Projet nom projet")
        self.accueil()
    def accueil(self):
        '''page d'accueil'''
        #insertion de l'image chapeau
        self.photo1 = PhotoImage(file="images/arbre_genealogie.png")
        self.picture = Label(fen, image=self.photo1)
        self.picture.photo = self.photo1
        self.picture.place(x=10, y=10)

        #Message d'accueil
        #à compléter

        #bouton de création d'un arbre
        self.bouton_creer = Button(fen, bg="#FFFFFF", text='créer',
                           command=self.creer)
        self.bouton_creer.place(x=10, y=350)
        #bouton d'ajout
        self.bouton_ajout = Button(fen, bg="#FFFFFF", text='ajouter',
                           command=self.ajouter)
        self.bouton_ajout.place(x=70, y=350)
        #bouton d'affichage infixe
        self.bouton_aff_infixe = Button(fen, bg="#FFFFFF", text="afficher les membres",
                           command=self.afficher)
        self.bouton_aff_infixe.place(x=140, y=350)
        #bouton de supression
        self.bouton_sup = Button(fen, bg="#FFFFFF", text='informations',
                           command=self.informer)
        self.bouton_sup.place(x=320, y=350)
    def quitter(self):
        fen.destroy()
    def ajouter(self):
        '''Cette méthode permet d'accéder à la fenêtre d'ajout d'un membre de la famille
        pour compléter l'arbre généalogique'''
        #on supprime les composants précédement créés (car en réalité nous restons sur la même interface
        # dans cet exemple
        self.bouton_creer.destroy()
        self.bouton_ajout.destroy()
        self.bouton_aff_infixe.destroy()
        self.bouton_sup.destroy()

        # on crée les composants graphiques nécessaires à l'ajout d'un membre dans notre arbre généalogique
        self.label1 = Label(fen, bg="#d9d9d9",text='Nom')
        self.label1.place(x=10, y=200)
        self.nom     = StringVar(fen)
        self.txtEditNom=TextEdit = Entry(fen, textvariable=self.nom)
        self.txtEditNom.place(x=130, y=200)

        self.label2 = Label(fen,bg="#d9d9d9", text='Date de naissance')
```

```
        self.date_naissance=StringVar(fen)
        self.label2.place(x=10, y=280)

        self.txtEditDate = Entry(fen, textvariable=self.date_naissance)
        self.txtEditDate.place(x=130, y=280)
        #bouton de validation
        self.bouton_val = Button(fen, bg="grey", text ='valider',
                                 command=self.accueil)
        self.bouton_val.place(x=300, y=350)
    def creer(self):
        pass
    def afficher(self):
        pass
    def informer (self):
        pass

if __name__ =="__main__":

    fen = Tk()
    app = App(fen)
    fen.mainloop()
```