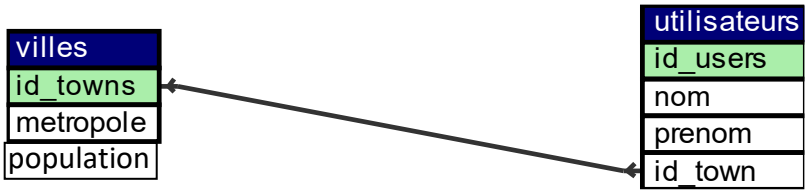
0. Objectifs

Dans le Tp_1 précédent, nous avons créé les tables ***villes*** et ***utilisateurs*** ci-dessous. Nous les avons remplis et avons exécuté des requêtes SQL de sélection afin d'en exploiter les données.



Ce travail avait été réalisé par l'intermédiaire du logiciel *DB Browser* qui permet de créer une base de données, d'exécuter des requêtes Sql et de visualiser le contenu des tables.

On se propose à présent, dans ce Tp_2, de réaliser les mêmes opérations en exécutant les mêmes requêtes Sql directement dans un code python. On utilisera ici uniquement *DB Browser*, pour vérifier l'état de la base de données créée.

1. Préparation de sa « zone de travail » :

Se placer dans un répertoire de travail et :

- Ouvrir un fichier nommé ***tp2_python.py*** qui contiendra les lignes suivantes :

Si le fichier n'existe pas, une base de données à ce nom est créée automatiquement

Importe la bibliothèque native de python *sqlite3*

connexion est un objet lié à la base de données

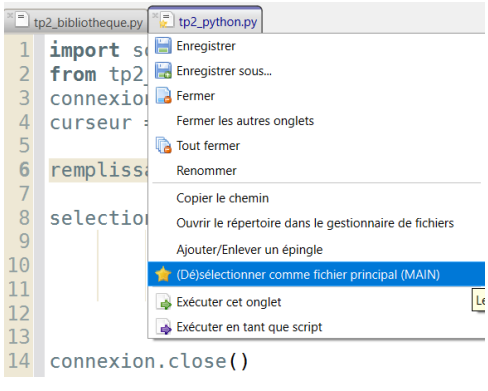
curseur est un objet de la classe sqlite3

On rompt le lien entre l'objet et la base de donnée.

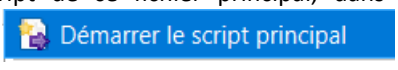
```
import sqlite3
from tp2_bibliotheque import remplissageBase , selection
connexion = sqlite3.connect("tp2_python.db")
curseur = connexion.cursor()

remplissageBase(connexion,curseur)

selection(curseur,"")

connexion.close()
```

- Ouvrir un fichier nommé ***tp2_bibliotheque.py*** qui contiendra les lignes suivantes :

```
1  def remplissageBase(connexion,curseur) :
2      pass
3
4
5  def selection(curseur,requete) :
6      pass
```



- Avec un click droit sur l'onglet du fichier ***tp2_python.py***, définir ce fichier comme *fichier principal*.

- Pour exécuter le script de ce fichier principal, dans l'onglet *Exécuter*, cliquer sur 

2. Remplissage de la base de données :

Pour plus de clarté, les instructions qui permettront de créer les 2 tables et de les remplir seront toutes placées dans la fonction *remplissageBase()*.

Pour exécuter une requête SQL de création, d'insertion, de modification de la bdd, on procédera ainsi :

```
def remplissageBase(connexion,curseur) :
    curseur.executescript("""
                          DROP TABLE IF EXISTS villes   ;
                          DROP TABLE IF EXISTS utilisateurs   ;
                      """)
    connexion.commit()

    curseur.executescript("""
                          CREATE TABLE villes  (
                              id_town INTEGER PRIMARY KEY,
                              metropole TEXT,
                              population INTEGER
                          ) ;
                          CREATE TABLE utilisateurs  (
                              id_user INTEGER PRIMARY KEY ,
                              nom TEXT,
                              prenom TEXT,
                              genre TEXT,
                              age INTEGER,
                              id_towns INTEGER,
                              FOREIGN KEY(id_towns) REFERENCES villes(id_town)
                          ) ;
                      """)
    connexion.commit()
```

Lors de sa mise au point, cette fonction sera exécutée plusieurs fois. Pour prévenir des erreurs, on supprime ici les tables qui seront recréées ensuite.

commit() permet d'appliquer les modifications sur la bdd

On peut exécuter plusieurs requêtes en même temps. Les requêtes doivent être séparées par un ;

⇨ Compléter le script de cette fonction en rajoutant les requêtes SQL établies dans le Tp1 précédent et qui permettent de remplir la bdd de la façon suivante. Ouvrir la bdd `tp2_python.db` avec le logiciel *DB Browser* afin d'en vérifier le contenu.

Table : villes

	id_town	metropole	population
	Filtre	Filtre	Filtre
1	1	LYON	513000
2	2	PARIS	2161000
3	3	MARSEILLE	861000
4	4	MACON	33000

Table : utilisateurs

	id_user	nom	prenom	genre	age	id_towns
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Chouhan	Jean	homme	50	1
2	2	Durand	Louis	homme	83	1
3	3	GranJean	Alice	femme	45	2
4	4	Bobet	Louison	homme	27	3
5	5	Champin	Arnaud	homme	23	1

3. Lecture du contenu de cette bdd :

Pour exécuter des requêtes SQL de sélection, on peut par exemple écrire dans le programme principal :

```
import sqlite3
from tp2_bibliotheque import remplissageBase , selection
connexion = sqlite3.connect("tp2_python.db")
curseur = connexion.cursor()

remplissageBase(connexion,curseur)

selection(curseur,"")
```

```
curseur.execute("SELECT * FROM villes")
resultat = curseur.fetchall()
print(resultat)
```

```
connexion.close()
```

On exécute la méthode *execute()* sur l'objet *curseur*

On exécute la méthode *fetchall()* sur l'objet *curseur*

La variable ***resultat*** qui est retournée par la méthode *fetchall()* est une liste de ***tuples*** :

```
[(1, 'LYON', 513000), (2, 'PARIS', 2161000), (3, 'MARSEILLE', 861000), (4, 'MACON', 33000)]
```

Pour lire par exemple la population de Paris, en python cela donnerait :

```
>>> resultat[1][2]
2161000
```

Afin d'avoir un code plus lisible, on propose de reporter ces lignes dans la fonction *selection()* afin de pouvoir simplement écrire dans le programme principal :

```
import sqlite3
from tp2_bibliotheque import remplissageBase , selection
connexion = sqlite3.connect("tp2_python.db")
curseur = connexion.cursor()

remplissageBase(connexion,curseur)

resultat = selection(curseur,
                     """
                         SELECT * FROM villes  ;
                     """)
print(resultat)

connexion.close()
```

⇨ Compléter le contenu de la fonction *selection()* définie dans ***tp2_bibliotheque.py*** .

⇨ Reprendre les lignes SQL de sélection définies dans le Tp1 précédent et réécrites dans le tableau donné ci-après. Les exécuter les unes après les autres et copier à chaque fois le contenu de la liste ***resultat*** associée.

Requête SQL	Liste ***resultat***
`SELECT SUM(age) AS somme_age FROM utilisateurs;`	
`SELECT COUNT(*) AS nb FROM utilisateurs;`	
`SELECT genre , COUNT(*) AS nb FROM utilisateurs GROUP BY genre ;`	
`SELECT AVG(age) AS ageMoyen FROM utilisateurs;`	
`SELECT genre,AVG(age) AS ageMoyen FROM utilisateurs GROUP BY genre;`	

`SELECT nom,prenom FROM utilisateurs`
`JOIN villes ON utilisateurs.id_towns = villes.id_town`
`WHERE metropole = 'LYON';`	

`SELECT u.nom,u.prenom, v.population FROM villes AS v`
`JOIN utilisateurs AS u ON v.id_town = u.id_towns`
`WHERE u.nom = 'Durand'`	

`SELECT u.nom,u.prenom FROM utilisateurs AS u`
`JOIN villes AS v ON u.id_towns = v.id_town`
`WHERE v.population > 1000000`	