M>r< B>r<

Fiche de TP no 4

SQLite et Python

Voici quelques informations (très) basiques permettant l'utilisation de SQLite3 (et donc d'une base de données) directement dans du code python. De nombreux sites internet permettent d'approfondir ce lien (la documentation officielle : http://docs.python.org/2/library/sqlite3.html).

Environnement

La première chose à faire (depuis un script ou depuis un environnement interactif tel que `iptyhon`) c'est d'importer le bon module :

```
1 import sqlite3
```

Comme dans le cas de la ligne de commande, il faut ensuite se connecter à une base (si elle n'existe pas, elle sera créée) :

```
2 connexion = sqlite3.connect('ma_base.db')
```

A la fin de votre programme, si vous voulez que les changements apportés à la base soient enregistrés :

```
3 connexion.commit()
```

Et puis n'oubliez pas de fermer proprement :

```
4 connexion.close()
```

Gestion des requêtes

On ne réalise pas des requêtes directement sur une connexion mais sur un objet qu'on appel curseur et qui peut contenir plusieurs lignes d'un table, quelque soit son schéma. Pour créer un curseur :

```
5 curseur = connexion.cursor()
```

Pour utiliser un curseur, on utilise la méthode `execute`. Par exemple :

```
6 curseur.execute("SELECT * FROM hotel WHERE ville='Nice'")
```

On trouve alors 0, 1 ou plusieurs lignes dans le curseur. Pour récupérer la première ligne (et 'déplacer' le curseur sur la ligne suivante), on peut écrire :

```
7 ligne = curseur.fetchone()
```

Vous pouvez aussi récupérer toutes les lignes renvoyés sous forme d'une liste (ce qui vide le curseur) :

```
8 lignes = curseur.fetchall()
```

A ce moment, il est possible de tester si aucune ligne a été renvoyée en regardant la taille de la liste.

Bien entendu, vous pouvez utiliser les boucles :

```
requete='SELECT * FROM hotel WHERE etoile = 3'
for ligne in curseur.execute(requete):
    print ligne
```

Quand on récupère une ligne dans une variable, elle se comporte comme un iterable :

```
curseur.execute('SELECT nom, prenom FROM Client')
ligne = curseur.fetchone()
nom = ligne[0]
prenom = ligne[1]
```

(Cherchez le mot clé `iterable` ici : https://docs.python.org/3/glossary.html).

Vous pouvez bien entendu utiliser des variables :

```
prenom = 'Jayme'
nom = 'Labédé'
requete='SELECT numclient FROM Client where prenom=? AND nom=?'
curseur.execute(requete, (prenom,nom))
```

Attention, même si ça peut éventuellement marcher, il est très déconseillé d'accomplir ça de cette façon :

```
requete="SELECT numclient FROM Client where prenom='{}' AND nom='{}'"
curseur.execute(requete.format(prenom,nom))
```

Une des difficultés principales vient de la gestion des types, en particulier quand ils sont complexes (ex. : coordonnées, date, ...). Mais ça c'est une autre histoire !

Les exercices

Dans ce TP, nous allons (encore !) travailler sur la base Hôtellerie des TPs précédents. L'idée est de gérer la base directement via du code python. Les scritps `sql` pour creer la base et l ;inhabiter se trouvent toujour par la :

http://pageperso.lis-lab.fr/~luigi.santocanale/teaching/BD/code/BD-Hotel-SQLite.zip

Notes :

- n'hésitez pas à écrire des fonctions auxiliaires pour découper la difficulté ;
- n'hésitez pas non plus à chercher la documentation sur le web ;
- testez le bon fonctionnement des chaque fonction que l'on vous demande d'écrire.

Question 1. Se rendre dans le répertoire du TP1 et écrire son code python à cet endroit. Vérifier qu'on a bien un fichier `hotellerie.db` ; si ce n'est pas le cas, le créer.

Question 2. Écrire une fonction `affiche_hotel(ville)` qui prend en entrée le nom d'une ville et affiche à l'écran les hôtels de cette ville.

Question 3. Écrire une fonction python `ajout_client(nom, prenom)` qui prend en entrée le nom et le prénom d'un nouveau client et l'ajoute à la base. La fonction affichera un message d'erreur si la personne existe déjà dans la base.

Découpez cette tache : d'abord on testera l'existence du client, et ensuite, si le client n'est pas dans la table, on fera son insertion dans la table.

Question 4. Écrire une fonction `bien_arrivee(numclient, numhotel, date_arrivee)` qui cherche une réservation correspondant à ce client dans cet hôtel à cette date ; si la réservation est trouvée, la fonction crée l'occupation correspondante et supprime la réservation.
Découpez cette tache en trois morceaux : `SELECT`, `INSERT`, et enfin `DELETE`.

Question 5. Écrire une fonction `reservation(client, hotel, arrivee, depart)` qui prend en entrée le numéro d'un client, le numéro d'un hôtel, une date d'arrivée, une date de départ et crée la réservation correspondante si et seulement si il y a une chambre disponible dans cet hôtel à ces dates. La fonction affiche un message si ce n'est pas possible et retourne -1 (elle retourne 0 si tout s'est bien passé).
Vous pouvez découper cette fonction en deux morceaux. Écrivez d'abord une fonction `chambre_libre(hotel, arrivee, depart)` qui teste l'existence d'une chambre libre dans un hôtel donné pour la période demandé. Cette fonction retournera `None` s'il n y a aucune chambre libre, ou bien le numéro de chambre libre pour la période.
Cette fonction sera appelée par la fonction `reservation(client, hotel, arrivee, depart)`, qui ajoutera la réservation si la valeur de retour n'est pas `None` et, autrement, affichera le message et retourne -1.

Question 6. Écrire le code python qui crée la table `Facture` contenant l'information sur : le client (numclient donc), le coût d'un séjour, l'occupation correspondante (numoccup donc), et l'hôtel du séjour (numhotel donc).

Question 7. Écrire une fonction `facture_sejours_terminés()` qui ajoute une ligne (correcte !) à la table Facture pour chaque occupation terminée.

Question 8. Écrire une fonction `faillite(numhotel)` qui supprime l'hôtel dont le numéro est passé en paramètre. Cette fonction s'assure de reloger dans un hôtel de la même ville tous les occupants actuels de l'hôtel (rappel : nous sommes le 4 avril 2014). Elle doit aussi modifier les réservations futures pour cet hôtel en les répartissant sur les autres hôtels de la même ville (vous pouvez utiliser la fonction reservation précédente). Un message doit apparaître s'il n'y a pas assez de places dans les hôtels de la ville pour reloger tous les occupants et/ou modifier toutes les réservations.