Gestion de BDD sous Python : météo aux USA

Dans ce TP, nous allons étudier comment coupler un logiciel de gestion de BDD (ici SQlite) et Python de sorte à traiter les données issues de la base.

Partie n°1: Accès à une base de données SQLite depuis Python

Le langage Python permet d'accéder facilement à des bases de données SQLite, grâce au module `sqlite3`.
Le schéma général d'un programme Python utilisant une base SQLite est le suivant :

Python
```
import sqlite3
conn = sqlite3.connect("mabase.db3")          # ouverture de la base
conn.row_factory = sqlite3.Row                # accès facile aux colonnes par leur nom
c = conn.cursor()                             # obtention d'un curseur permettant d'interroger la BDD
#
# utilisation de la base
#
c.close()
```

Le curseur `c` est un objet qui permet d'interroger la BDD `mabase.db3`. Pour effectuer une requête, on appelle la méthode `execute` appliquée au curseur `c`.
On peut alors parcourir les lignes de la table résultat, et pour chacune d'entre elles on peut facilement accéder aux colonnes :

Python
```
c.execute("SELECT titre, annee FROM livres JOIN auteurs ON livres.id_auteurs =
auteurs.id_auteurs WHERE auteur = 'duras'")
for ligne in c:
        print("Marguerite Duras a écrit " + ligne["titre"] + "en" + ligne["annee"])
c.close()
```

• Données météorologiques NOAA

La NOAA (*National Oceanic and Atmospheric Administration*), une agence des États-Unis, diffuse des données météorologiques relatives au territoire américain. Il s'agit de relevés journaliers, effectués en général dans les aéroports.
Le fichier `noaa2012.db3` contient les données journalières pour l'année 2012.
La base de données est constituée de deux tables :

- `Station` qui modélise les stations de relevés :
  - `StationId` : identifiant unique à usage interne
  - `CallSign` : indicatif, unique également, en général un code d'aéroport
  - `Name` : nom
  - `State` : code de l'état (code à deux lettres, cf. carte en fin d'énoncé)
  - `Location` : emplacement, en langage naturel
  - `Latitude`, `Longitude` : coordonnées géographiques (degrés)

- ◇ `Weather` qui modélise un relevé journalier :
  - — `StationId` : identifiant de la station
  - — `Date` : date, sous la forme AAAAMMJJ
  - — `Tmax`, `Tmin`, `Tavg` : températures maximale, minimale, moyenne (degrés Celsius)

Rq

Dans la table Weather certaines valeurs peuvent valoir NONE ou NULL, pour le cas où les données météorologiques n'ont pas été disponibles.

Table Station						
StationId	CallSign	Name	State	Location	Latitude	Longitude
03011	TEX	TELLURIDE	CO	TELLURIDE REGIONAL AIRPORT	37.95	-107.9
03012	SKX	TAOS	NM	TAOS REGIONAL AIRPORT	36.45	-105.67
03013	LAA	LAMAR	CO	LAMAR MUNICIPAL AIRPORT	38.07	-102.68

Table Weather				
StationId	Date	Tmax	Tmin	Tavg
03072	20120128	10.0	1.111	4.44444444444444
03072	20120129	16.1111111	3.33333333333333	6.66666666666667

• **Premières requêtes simples sous Python**

1. Faire la liste des noms des aéroports du Wyoming (code WY).

2. Dénombrer les aéroports du sud des USA puis du nord des USA. On prendra la limite nord/sud au dessus de 45° de latitude.

3. Afficher la température maximale atteinte aux USA. Afficher la température minimale atteinte aux USA.

4. Afficher la date correspondant au jour le plus chaud aux USA. (on pourra faire de même pour le jour le plus froid).

5. Afficher l'état des Etats-Unis où la température minimale a été atteinte.

Partie n°2: Météo de l'aéroport JFK

6. Placer dans une liste les températures journalières minimales à l'aéroport John F. Kennedy de New York (indicatif JFK).

7. Tracer sur un même graphe les températures journalières minimum, maximum et moyenne à l'aéroport JFK, sur l'année 2012. Dans un premier temps, on numérote les jours par un numéro d'ordre. Penser à ajouter une légende et des axes.

Cette fin de partie (question 8) est facultative.

Nous souhaitons maintenant afficher des noms de mois et plus seulement les numéros de jours. Pour cela, nous allons transformer la date indiquée dans la table de données en un objet Python nommé `date`, puis nous demanderons à `Matplotlib` de considérer ces dates comme telles.
Pour manipuler des dates en Python il faut commencer par importer le module `datetime` avec la commande `import datetime`. Ensuite, on peut convertir des chaînes en dates à l'aide de la fonction `strptime` à laquelle on doit en plus indiquer le format utilisé. Le format AAAAMMJJ s'écrit `"%Y%m%d"`.

Python

```
>>>print(datetime.datetime.strptime('20120101',"%Y%m%d"))
2012-01-01 00:00:00
```

L'affichage des dates sur le graphique peut se faire en utilisant les méthodes `xaxis_date` et `autofmt_xdate`, selon le schéma suivant :

```python
fig = plt.figure()              # crée une figure
sub = fig.add_subplot(111)      # crée un tracé dans la figure
sub.plot( ... )                 # effectue le tracé
sub.plot( ... )
sub.plot( ... )
sub.xaxis_date()                # configure l'axe X pour être étiqueté par des dates
fig.autofmt_xdate()             # met les dates en diagonale sous l'axe X
plt.show()                      # affiche la figure
```

8. Mettre en œuvre la gestion des dates.

Partie n°3: Filtrage des données

Nous souhaitons maintenant éliminer les variations rapides de température, et donc moyenner la température sur un certain nombre de jours. Pour cela, nous devons construire un filtre qui manipule par exemple, une fenêtre glissante de 10 jours.
Si `x` est une liste de valeurs à filtrer, et `y` la liste filtrée, un filtre glissant sur 10 éléments est :

$$y_n = \frac{1}{10}\left(x_n + x_{n-1} + \cdots + x_{n-9}\right)$$

Chaque échantillon y_n sera la moyenne des 10 x_i précédents.

9. Ecrire une fonction `filtre10` qui filtre une liste avec une fenêtre glissante de 10 jours. Pour les 10 premiers éléments, on utilisera les valeurs des derniers éléments de la liste en supposant que le phénomène observé possède une certaine périodicité (ce qui est bien sûr notre cas).

10. Appliquer ce filtre aux températures tracées précédemment et en déduire un nouveau graphe. Il pourra être nécessaire éliminer les derniers éléments des listes avec la commande `remove(element)`.

Partie n°4: Tracé de la carte des températures dans les USA

Pour une date donnée, nous souhaitons tracer géographiquement la température de toutes les stations du territoire américain.

11. Récupérer les températures moyennes, latitudes et longitudes de toutes les stations le 15 janvier 2012. Pour éviter d'avoir une carte trop grande, car tenant compte de l'Alaska et d'Hawaï, se limiter aux latitudes de 25 à 50°, et aux longitudes de -127 à -66°.

12. Pour certaines stations, la donnée de température n'est pas disponible (NONE s'affiche). Modifier le script précédent de sorte à éliminer ces stations de la liste.
Sur une carte, nous allons afficher un point pour chacune de ces stations. L'abscisse sur la carte est la longitude ; l'ordonnée est la latitude. La couleur du point indiquera la température relevée. Pour cela nous allons utiliser la fonction `scatter` de `matplotlib`.

```python
plt.scatter(long, lat, c=Tavg, s=20)
plt.xticks([])  # élimine les graduations en abscisse
plt.yticks([])  # élimine les graduations en ordonnée
plt.show()
```

13. Afficher la carte des températures moyennes des USA à l'aide de la fonction `scatter`.

Pour l'affichage sur la carte, on a utilisé une projection équirectangulaire. Cependant, cette projection introduit d'importantes déformations lorsqu'on s'écarte de l'équateur. En conséquence, on peut corriger l'abscisse par un facteur $cos\phi_1$, où ϕ_1 est la latitude moyenne sur la carte. On a donc :

$$x = \lambda \cos \phi_1 \qquad \text{et} \qquad y = \phi$$

avec λ la longitude et ϕ la latitude.

14. Modifier le script précédent pour prendre en compte le problème de projection.