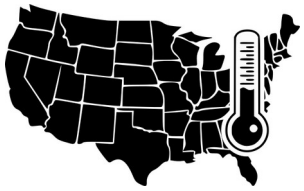
Météo des USA



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

Le but de ce TP sera de traiter et afficher sous forme graphique l'évolution des températures aux Etats Unis

1. Accès à une base de données MariadB depuis Python

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

```
import mysql.connector as sgbd      # importe la bibliothèque

mydb = sgbd.connect(                # se connecte à la DB myDatabase
        host="address_ip",
        port = 3306,
        user="user_name",
        password="secret",
        database="myDatabase"
)

mycursor = mydb.cursor()            # obtention d'un curseur permettant d'
#                                   interroger la BDD
# utilisation de la base
#
mydb.close()
```

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

```
mycursor.execute("SELECT titre, annee FROM livres JOIN auteurs ON
livres.id_auteurs = auteurs . id_auteurs WHERE auteur = 'duras'")

rows = cursor.fetchall()            # récupère la reponse

for ligne in rows :
    print("Marguerite Duras a écrit " + ligne [0] + "en" + ligne [1])
```

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 base Noaa2012 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)
- ✗ **GroundHeight**, **StationHeight** : données de hauteur

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)
- ✗ **Precip**, **Pressure**, **WindSpeed**, **WinDir** : Précipitation, pression et vent

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

Adresse de la base de données

L'accès à la base distante s'effectue avec les paramètres suivants :

- ✗ Adresse IP : **'176.181.118.75'**
- ✗ Port : **3306**
- ✗ Utilisateur : **'tp22_eleve'**
- ✗ Mot de passe : **'eleve'**
- ✗ Nom de la base : **'Noaa2012'**

Extraits des tables

Table station

StationId	CallSign	Name		Location	Latitude	Longitude	GroundHeight	StationHeight
03011	TEX	TELLURIDE	CO	TELLURIDE REGIONAL AIRPORT	37,95	-107,9	2 770,937	2 766,974
03012	SKX	TAOS	NM	TAOS REGIONAL AIRPORT	36,45	-105,667	2 161,337	2 161,337
03013	LAA	LAMAR	CO	LAMAR MUNICIPAL AIRPORT	38,07	-102,688	1 128,979	1 128,674
03014	4SL	TORREON	NM	TORREON	35,799	-107,181	2 042,16	2 105,863
03016	RIL	RIFLE	CO	GARFIELD CO REGIONAL ARPT	39,528	-107,72	1 683,106	1 689,811
03017	DEN	DENVER	CO	DENVER INTERNATIONAL AIRPO	39,833	-104,657	1 650,187	1 655,369

Table weather

	StationId	Date	Tmax	Tmin	Tavg	Precip	Pressure	WindSpeed	WinDir
8019	12896	20120121	27,222	19,444	23,333	0	30,09	[NULL]	[NULL]
8020	12896	20120122	25,556	19,444	22,778	0	30,08	[NULL]	[NULL]
8021	12896	20120123	27,222	22,222	25	0	30,07	[NULL]	[NULL]
8022	12896	20120124	27,778	22,222	25	0	30,09	[NULL]	[NULL]
8023	12896	20120125	27,778	22,778	25,556	0,01	30,09	[NULL]	[NULL]
8024	12896	20120126	27,778	23,333	25,556	0	30,04	[NULL]	[NULL]
8025	12896	20120127	28,333	23,333	26,111	0	30,02	[NULL]	[NULL]
8026	12896	20120128	26,667	22,222	24,444	0	30,09	[NULL]	[NULL]
8027	12896	20120129	25,556	20,556	23,333	0,42	30,15	[NULL]	[NULL]

2.Requêtes simples sous Python

Q1. Indiquer le nombre d'enregistrements présent dans la table weather

```
import mysql.connector as sgbd

mydb = sgbd.connect(
        host="176.181.118.75",
        port = 3306,
        user="tp22_eleve",
        password="eleve",
        database="Noaa2012"
)

mycursor = mydb.cursor()
mycursor. execute("SELECT COUNT(*) FROM station")
rows = mycursor.fetchall()
for ligne in rows :
    print (ligne [0])
```

Q2. Lister des noms des aéroports du Wyoming (code WY).

```
import mysql.connector as sgbd

mydb = sgbd.connect(
        host="176.181.118.75",
        port = 3306,
        user="tp22_eleve",
        password="eleve",
        database="Noaa2012"
)

mycursor = mydb.cursor()
mycursor. execute("SELECT Name FROM station WHERE STATE = 'WY'")
rows = mycursor.fetchall()
for ligne in rows :
    print (ligne [0])
```

Q3. Dénombrer les aéroports du sud des USA puis du nord des USA. On prendra la limite nord/sud au dessus de 45° de latitude.

```
mycursor. execute("SELECT count(StationId) FROM station WHERE Latitude >=
40")

mycursor. execute("SELECT count(StationId) FROM station WHERE Latitude <
40")
```

Q4. Afficher la température maximale atteinte aux USA. **Afficher** la température minimale atteinte aux USA.

```
mycursor. execute("SELECT MAX(Tmax) FROM weather")
```

59,44

Q5. Afficher la date correspondant au jour le plus chaud aux USA. (on pourra faire de même pour le jour le plus froid).

```
mycursor. execute("SELECT Date, MAX(Tmax) FROM weather")
```

('20120101', 59.4444444444444)

Q6. Afficher l'état des Etats-Unis où la température minimale a été atteinte.

```
mycursor. execute(" SELECT State, MIN(Tmin) FROM weather JOIN station ON
weather.StationId = station.StationId")
```

('CO', -52.2222222222222)

3.Météo de l'aéroport JFK

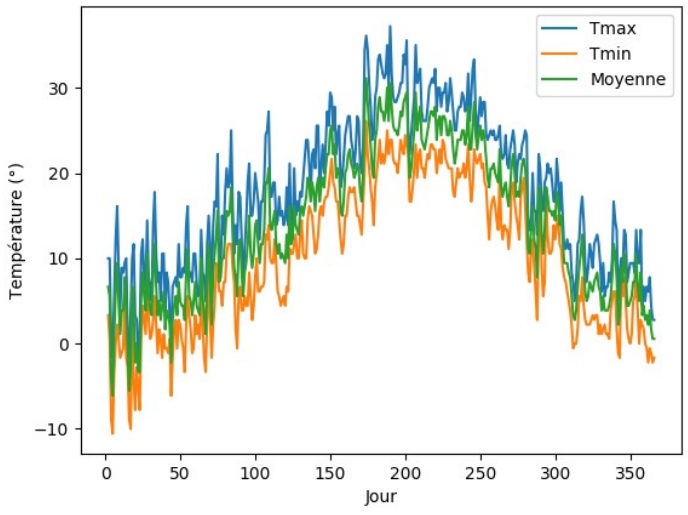
Q7. Placer dans une liste les températures journalières minimales à l'aéroport John F. Kennedy de New York (indicatif JFK).

```
mycursor. execute(" SELECT Tmin FROM weather JOIN station ON
weather.StationId = station . StationId WHERE CallSign = 'JFK'")
rows = mycursor.fetchall()
lsTmin=[]
for ligne in rows :
        lsTmin.append(ligne[0])
```

Q8. Tracer sur un même graphe les températures journalières minimum, maximum et moyenne à l'aéroport JFK, sur l'année 2012. Le module MATPLOTLIB permet de créer et afficher des graphiques. Le schéma suivant présente les principales fonctions à utiliser :

```
mycursor. execute(" SELECT Tmax, Tmin, Tavg FROM weather JOIN station ON
weather.StationId = station . StationId WHERE CallSign = 'JFK'")
rows = mycursor.fetchall()
lsTmin=[]
lsTmax = []
lsAvg = []
jour = []
i = 1
for ligne in rows :
    lsTmax.append(ligne[0])
    lsTmin.append(ligne[1])
    lsAvg.append(ligne[2])
    i+=1
    jour.append(i)

fig = plt.figure()
sub = fig.add_subplot(111)
sub.plot(jour, lsTmax, label='Tmax')
sub.plot(jour, lsTmin, label='Tmin')
sub.plot(jour, lsAvg, label='Moyenne')
plt.xlabel ('Jour')
plt.ylabel ('Température (°)')
plt.legend()
plt.show()
```



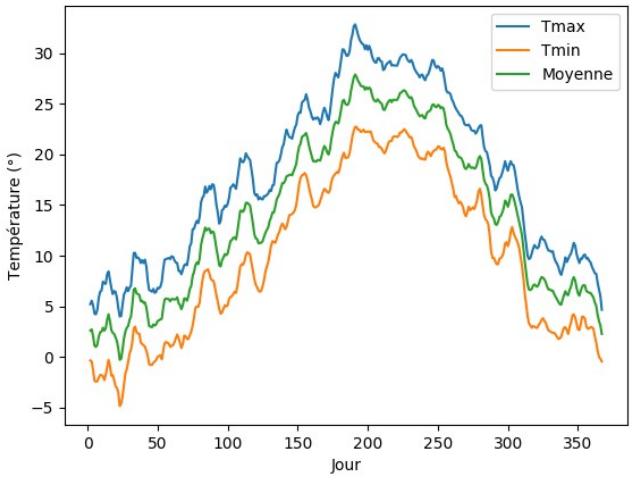
Filtrage des données

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 :

Chaque échantillon y_n sera la moyenne des 10 x_i précédents.

Q9. 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).

```
def filtre10 ( ls ) :
    """ prend en argument une liste ls ( liste de nombres) de longueur n
    renvoie une liste à n éléments.
    """
    res = [ ]
    for k in range(len(ls)) :
        tot = 0
        for j in range(k-9,k+1):
            if type(ls[j]) is not type(None) : tot +=ls [ j ]
        res .append(tot/10) #
    return res
sub = fig.add_subplot(111)
sub.plot(jour, lsTmax, label='Tmax')
sub.plot(jour, lsTmin, label='Tmin')
sub.plot(jour, lsAvg, label='Moyenne')
plt.xlabel ('Jour')
plt.ylabel ('Température (°)')
plt.legend()
plt.show()
```



Q10. Appliquer ce filtre aux températures tracées précédemment et **en déduire** un nouveau graphe.

Mise en forme des axes

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. 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".

```
>>> 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 :

```
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
```

Q11. Mettre en œuvre la gestion des dates.

```
mycursor. execute(" SELECT Tmax, Tmin, Tavg, Date FROM weather JOIN
station ON weather.StationId = station . StationId WHERE CallSign =
'JFK'")
rows = mycursor.fetchall()
lsTmin=[]
lsTmax = []
lsAvg = []
lsTminF10 = []
lsTmaxF10 = []
lsAvgF10 = []
lsDate = []
i = 1
for ligne in rows :
    lsTmax.append(ligne[0])
    lsTmin.append(ligne[1])
    lsAvg.append(ligne[2])
    lsDate .append(datetime.datetime.strptime ( ligne [3], "%Y%m%d"))
lsTminF10 = filtre10(lsTmin)
lsTmaxF10 = filtre10(lsTmax)
lsAvgF10 = filtre10(lsAvg)

fig = plt.figure()
sub = fig.add_subplot(111)
```

```
sub.plot(lsDate, lsTmaxF10, label='Tmax')
sub.plot(lsDate, lsTminF10, label='Tmin')
sub.plot(lsDate, lsAvgF10, label='Moyenne')
sub.xaxis_date()
fig .autofmt_xdate()
plt.ylabel ('Température (°)')
plt.legend()
plt.show()
```