Gestion de BDD sous Python : météo aux USA – Corrigé

Dans les requêtes suivantes (méthode `execute`), le nom des attributs correspond à des noms de variables. Il ne faut donc surtout pas ajouter de guillemets ou autres délimiteurs (dans ce cas, il s'agirait de chaînes de caractères...).

Par contre, pour accéder aux résultats de la requête, on peut parcourir une liste ou un tableau dont les colonnes ne sont pas repérées par leur index mais par des chaînes de caractères correspondant à la projection effectuée lors de la requête.

1.

Python
```
c.execute("SELECT Name FROM Station WHERE State = 'WY'")
for ligne in c:
    print(ligne["Name"])
c.close()
```

2.

Python
```
c.execute("SELECT count(StationId) FROM Station WHERE Latitude >= 40")
for ligne in c:
    print(ligne["count(StationId)"])
c.close()
```

Autre solution :

Python
```
c.execute("SELECT StationId FROM Station WHERE Latitude >= 40")
lsIdNd = []
for ligne in c:
    lsIdNd.append(ligne["StationId"])
print(len(lsIdNd))

c.execute("SELECT StationId FROM Station WHERE Latitude < 40")
lsIdSd = []
for ligne in c:
    lsIdSd.append(ligne["StationId"])
print(len(lsIdSd))
c.close()
```

3.

```python
c.execute("SELECT MAX(Tmax) FROM Weather")
for ligne in c:
    print(ligne["MAX(Tmax)"])
c.close()
```

4.

```python
c.execute("SELECT Date, MAX(Tmax) FROM Weather")
for ligne in c:
    print(ligne["MAX(Tmax)"],ligne["Date"])
c.close()
```

5. Pour cette requête, il faut faire une jointure sur l'attribut `StationId` présent dans les deux tables. On pourrait utiliser la commande `USING` comme vu en cours mais cette commande ne fonctionne pas ici. Dans ce cas, on doit préciser la table à chaque fois : par exemple : `Station.StationId`.

```python
c.execute(" SELECT State, MIN(Tmin)
            FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId")
for ligne in c:
    print(ligne["MIN(Tmin)"],ligne["State"])
c.close()
```

On peut aussi renommer les tables avec la commande `AS` pour alléger les syntaxes :

```python
c.execute(" SELECT State, MIN(Tmin)
            FROM Weather AS w JOIN Station as S
            ON w.StationId = s.StationId")
for ligne in c:
    print(ligne["MIN(Tmin)"],ligne["State"])
c.close()
```

Autre possibilité moins élégante :

```python
c.execute(" SELECT State FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId
            WHERE Tmin = (SELECT MIN(Tmin) FROM Weather )")
for ligne in c:
    print(ligne["State"])
c.close()
```

6.

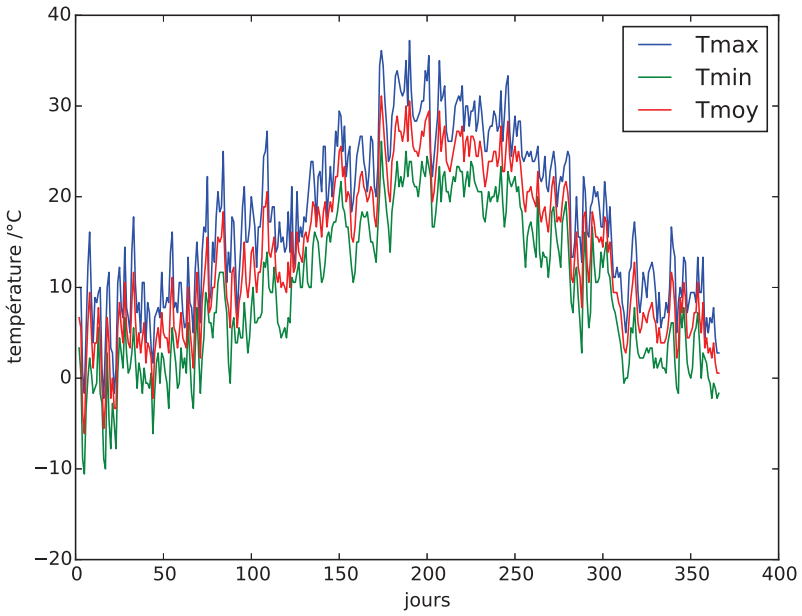
Python
```
c.execute(" SELECT Tmin FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId
            WHERE CallSign = 'JFK'")
lsTmin =[]
for ligne in c:
    lsTmin.append(ligne["Tmin"])
c.close()
```

7. On réalise ici une requête qui renvoie directement un "tableau" contenant les différentes données qui nous seront utiles pour la suite. Par contre, on ne peut parcourir l'élément c qu'une seule fois (lors d'une 2e boucle, c semble vide). Il faut donc, dans ce cas, construire les différentes listes `lsTmax`, `lsTmin`, `lsTavg`, `lsDate` dans une seule boucle.

Pour le tracé des courbes, le format de la date n'est pas très pratique 20120131. En effet, si on place la date en abscisse, il y a des problèmes lorsque l'on passe d'un mois à l'autre. par exemple l'abscisse passe de 20120131 à 20120201 soit un saut qui ne correspond plus à 1 jour...
Il existe différentes solutions :

- le plus simple : on trace les liste de températures `lsTmax`, `lsTmin`, `lsTavg` en fonction de leur indice avec la commande par exemple `plt.plot(lsTmax)`. On n'a ainsi pas de souci de vecteurs de dimensions différentes !
- on construit une liste `jour` comme proposé dans le corrigé ci-dessous (cette liste n'apporte pas grand chose...).

Python
```
c.execute(" SELECT Tmax, Tmin, Tavg FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId
            WHERE CallSign = 'JFK'")
lsTmax =[]
lsTmin =[]
lsTavg =[]
jour =[]
cpt =1

for ligne in c:
    lsTmax.append(ligne["Tmax"])
    lsTmin.append(ligne["Tmin"])
    lsTavg.append(ligne["Tavg"])
    cpt +=1
    jour.append(cpt)

c.close()

plt.plot(jour,lsTmax, label ='Tmax')
plt.plot(jour,lsTmin, label ='Tmin')
plt.plot(jour,lsTavg, label ='Tmoy')
plt.xlabel('jours')
plt.ylabel('température /°C')
plt.legend()
plt.show()
```



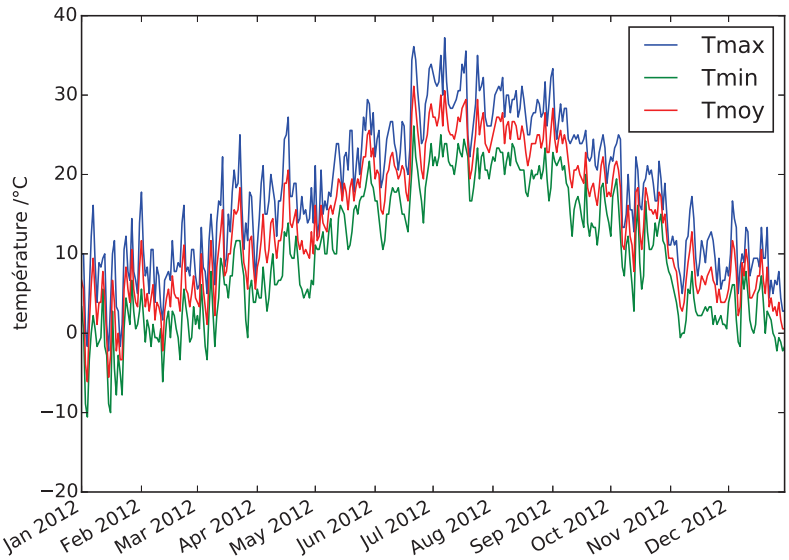
On remarque les listes de températures construites ci-dessus contiennent toutes une valeur `None` (dernier élément). ON peut d'ores et déjà supprimer ce dernier élément :

- ◇ méthode `remove` : par contre on modifie la longueur de la liste et on prend le risque d'avoir 3 listes de longueurs différentes (ce n'est pas forcément gênant).
- ◇ On peut mettre un test lors de la construction des listes du type :
  `if ligne["Tmax"] != None and ligne["Tmin"] != None and ligne["Tavg"] != None` et si cette proposition est vraie remplir les différentes listes.

8.

Python

```
c.execute(" SELECT Date, Tmax, Tmin, Tavg FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId
            WHERE CallSign = 'JFK'")
lsTmax =[]
lsTmin =[]
lsTavg =[]
lsDate =[]

for ligne in c:
    lsTmax.append(ligne["Tmax"])
    lsTmin.append(ligne["Tmin"])
    lsTavg.append(ligne["Tavg"])
    lsDate.append(datetime.datetime.strptime(ligne["Date"], "%Y%m%d"))

c.close()

fig = plt.figure()              # crée une figure
sub = fig.add_subplot(111)      # crée un tracé dans la figure
sub.plot(lsDate,lsTmax, label ='Tmax')
sub.plot(lsDate,lsTmin, label ='Tmin')
sub.plot(lsDate,lsTavg, label ='Tmoy')
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.ylabel('température /°C')
plt.legend()
plt.show()
```



9. Pour la fonction filtrage, on utilise le fait que les relevés météo soient périodiques (période = 1 année) et aussi que l'on peut facilement faire appel aux derniers éléments d'une liste par des indices négatifs. Par exemple, avec une liste `Ls` de `n` éléments, on aura : `Ls[n-1] = Ls[-1]` et quel que soit l'entier `0<k<n` : `Ls[n-k] = Ls[-k]`.

Pour cette fonction, il faut bien comprendre l'algorithme proposé, et le principe de la fenêtre glissante. Cette fonction doit prendre en argument une liste de nombres, et elle doit renvoyer une nouvelle liste de nombres de même dimension (même nombre d'éléments que la liste prise en argument).

Python
```
def filtre(ls):
    """ prend en argument une liste ls (liste de nombres) de longueur n
        renvoie une liste à n éléments.
    """
    res = []          # initialisation : liste vide
    for k in range(len(ls)): # On parcourt la liste passée en argument
        tot = 0              # initialisation, variable locale
        for j in range(k-9,k+1):    # on parcourt les 9 éléments précédents
                                    # l'élément d'indice k (et lui même)
            tot +=ls[j]     # on fait la somme dans le but de faire la moyenne (division par 10 ensuite)
        res.append(tot/10)  #
    return res
```

10.

Python
```
# Il faut retirer les derniers éléments des listes qui affiche la valeur NONE
lsTmax.remove(None)
lsTmin.remove(None)
lsTavg.remove(None)
lsDate.remove(datetime.datetime(2012, 12, 31, 0, 0))

fig = plt.figure()            # crée une figure
sub = fig.add_subplot(111)  # crée un tracé dans la figure
sub.plot(lsDate, filtre(lsTmax), label ='Tmax')
sub.plot(lsDate, filtre(lsTmin), label ='Tmin')
sub.plot(lsDate, filtre(lsTavg), label ='Tmoy')
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.ylabel('température /°C')
plt.legend()
plt.show()
```

11.

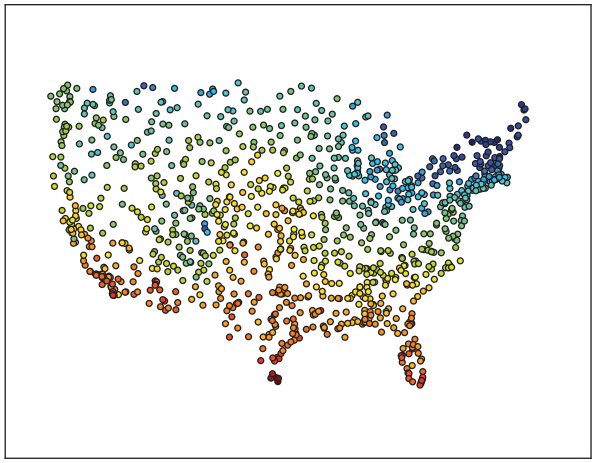
```python
c.execute(" SELECT Tavg, Latitude, Longitude FROM Weather JOIN Station
            ON Weather.StationId = Station.StationId
            WHERE Latitude > 25 AND Latitude < 50 AND Longitude >-127
            AND Longitude <66 AND Date = '20120115'")
lsTavg = []
lsLong = []
lsLat = []

for ligne in c:
    lsTavg.append(ligne["Tavg"])
    lsLong.append(ligne["Longitude"])
    lsLat.append(ligne["Latitude"])
c.close()
```

12.

```python
for ligne in c:
    if ligne["Tavg"] != None:
        lsTavg.append(ligne["Tavg"])
        lsLong.append(ligne["Longitude"])
        lsLat.append(ligne["Latitude"])
```

13.

Python
```
plt.scatter(lsLong , lsLat, c=lsTavg, s=20)
plt.xticks([])
plt.yticks([])
plt.show()
```



14.

Python
```
# calcul de la latitude moyenne
phi1 =0
for k in lsLat:
    phi1+=k
phi1 =phi1/len(lsLat)

# modification de la carte
plt.scatter(np.array(lsLong)*np.cos(phi1*np.pi/180) , lsLat, c=lsTavg, s=20)
plt.xticks([])
plt.yticks([])
plt.show()
```