Table des matières

Python, Importation, Sqlite3, exemple

Revenir à l'accueil Base de données	Amélioration de l'affichage et Manipulations de base sur les données	Echanges de données entre logiciels

Importation-exportation CSV en utilisant un programme Python

Le problème

Lorsque vous faites une exportation contenant des chaînes complexes (plusieurs mots avec des caractères séparateurs ou spéciaux), le contenu de ces chaînes est "entouré" de guillemets (voir ici).

Vous aurez le même résultat si le séparateur utilisé est la virgule et si les données sont des nombres réels avec vigule au format français.

Il peut être pratique d'utiliser un script écrit en Python pour extraire les données du fichier, les formater et les insérer "au bon format" dans la base de données.

Le programme Python

Ce programme va vous permettre de faire connaissance avec la programmation orientée bases de données avec Python. Il vous présente, d'abord, des instructions permettant de connaître l'environnement base de données.

Ensuite il présente le parcours du fichier, le formatage des données et enfin l'insertion des données dans une table.

Exemple 1 : introduction à la programmation Python avec les bases de données

Utiliser les modules appropriés et imposer un espace de nommage.

```
import sqlite3 as lite
import sys
```

On indique à Python :

- qu'il doit charger (copier) dans notre script et utiliser du code qui est contenu dans le module **sqlite3** (et aussi dans le module **sys**) ;
- que nous allons utiliser ce code dans notre script grâce à "l'alias" : **lite**.

Se connecter à la base

- La documentation du module sqlite3 de python

```
To use the module, you must first create a Connection object that
represents the database.
```

- Le code

```
connexion = None

try:
 connexion = lite.connect('emplacement/de/la/base/test.db')
```

L'espace de nommage **lite** permet d'indiquer à Python où il peut trouver la définition de la méthode *connect()* : ici, elle est dans **sqlite3**.

Création d'un objet curseur

La documentation

```
To retrieve data after executing a SELECT statement, you can either treat
the cursor as an iterator,
call the cursor’s fetchone() method to retrieve a single matching row,
or call fetchall() to get a list of the matching rows.
```

Un curseur agit sur une base de données (une connexion) et permettra **d'interroger les données ligne par ligne à partir d'une requête**.

Connaître la version de sqlite

Sqlite, comme tous les SGBDR (Systèmes de Gestion de Bases de Données Relationnelles), gère ses propres informations de stockage de la base de données (nom des tables, nom des colonnes, ...) sous forme de tables. Il suffit donc d'interroger la table système adéquate, ici grâce à une fonction enregistrée dans le catalogue de SQLite pour connaître la version de sqlite utilisée.

Le code

```
cur = connexion.cursor()
cur.execute('SELECT SQLITE_VERSION()')
data = cur.fetchone()
```

On demande à exécuter la requête puis de lire la première ligne du curseur. Ensuite, il faudrait

récupérer les données de cette ligne et les utiliser pour un affichage ou un traitement.

Connaître les tables de la base de données

```
curseur.execute('select * from sqlite_master');
lignes = curseur.fetchall();
for ligne in lignes:
    print ("SQLite master: {0}".format(ligne) )
```

Le résultat

```
SQLite master: ('table', 'livre2', 'livre2', 3, 'CREATE TABLE livre2(titre
varchar(50), auteur varchar(255),annee_parution int)')
```

Le code complet

```
import sqlite3 as lite
import sys

connexion = None
cheminDeLabase='C:\\pgms\\sqlite3\\livre\\v2\\livre2AvecPython.db'
try:
 connexion = lite.connect(cheminDeLabase)
 cur = connexion.cursor()
 cur.execute('SELECT SQLITE_VERSION()')
 data = cur.fetchone()
 print ("SQLite version: {0}".format(data) )
 curseur.execute('select * from sqlite_master');
 lignes = curseur.fetchall();
 for ligne in lignes:
     print ("SQLite master: {0}".format(ligne) )
except lite.Error as e:
 print ("Error %s:" % e.args[0])
 sys.exit(1)

finally:

 if connexion :
   connexion .close()
```

Résultat de l'exécution de ce script

```
SQLite version: ('3.7.12',)
```

```
SQLite master: ('table', 'livre2', 'livre2', 3, 'CREATE TABLE livre2(titre
```

```
varchar(50), auteur varchar(255),annee_parution int)')
```

Le même script écrit différemment et avec une instruction en plus

```
import sqlite3 as lite
import sys

connexion = None

connexion = lite.connect('c:\pgms\sqlite3\essais.db')
with connexion:

 cur = connexion.cursor()                ## Création d'un objet nommé cur de
type "cursor"
                                         ## Objet permettra d'interroger les
tables de la base.
 cur.execute('SELECT SQLITE_VERSION()')  ##
 data = cur.fetchone()
 print ("SQLite version: {0}".format(data) )
  curseur.execute('select * from sqlite_master');
 lignes = curseur.fetchall();
 for ligne in lignes:
     print ("SQLite master: {0}".format(ligne) )
 cur.execute('SELECT * from livre')
  ....
```

Le résultat

```
Traceback (most recent call last):
 File "F:/__2012_2013/_PREPAS/python/essais/bdd1", line 37, in <module>
 cur.execute('SELECT * from livre')
sqlite3.OperationalError: Could not decode to UTF-8 column 'lvr_titre' with
text 'Apprendre  programmer avec Python 3'
>>>
```

- le problème ?
- la solution ?

Quelques éléments de réponse : la solution est à rechercher du côté de l'encodage des caractères. L'invite de commandes travaille en mode "MS-DOS". SQLite 3 et Python travaillent en Unicode avec un encodage UTF-8.

- Expliquez clairement le problème et indiquez comment est encodé le caractère à, si on utilise l'invite de commandes et l'UTF-8.

<spoiler | Une solution possible >

- Supprimer la ligne posant problème

```
sqlite> SELECT * FROM livre;
 lvr_titre                              lvr_annee_parution
-----------------------------------  ------------------
Apprendre à programmer avec Python 3  2012
sqlite> DELETE FROM livre where lvr_annee_parution=2012 ;
sqlite> .tables livre
sqlite> SELECT * FROM livre;
sqlite> .mode column
sqlite> .headers on
sqlite> SELECT * FROM livre;
sqlite>
```

</spoiler>

Lire un fichier, traiter ses données et écrire dans une table

Générer le fichier de données

Nous partirons de la table *donnees_mesures* dans laquelle des valeurs ont été insérées précédemment et injecterons ces données dans la table *donnees_mesures2*.

Création de la table donnees_mesures2

```
CREATE TABLE donnees_mesures2(x DOUBLE, y DOUBLE, t DOUBLE);
```

```
sqlite> .tables
donnees_mesures   donnees_mesures2  livre
sqlite> .schema donnees_mesures2
CREATE TABLE donnees_mesures2(x double, y double, t double);
sqlite> .exit
```

Exportation CSV des données de la table donnees_mesures

```
sqlite> .mode csv
sqlite> .separator ","
sqlite> .output x_y.csv
sqlite> select * from donnees_mesures;
sqlite> .output stdout
sqlite> select * from donnees_mesures;
"1,1","2,4468240195"
"2,1","2,8420511522"
.....
sqlite> .mode column
```

Après la commande *.output stdout*, l'affichage est encore en mode CSV. La dernière instruction

SELECT nous montre comment sont stockées les données dans le fichier.

SQLite 3 a détecté la présence de virgules dans les lignes de données de la table et a donc entouré ces données avec des guillemets pour les distinguer des virgules de séparation des champs.

```
sqlite> SELECT * FROM donnees_mesures2;
sqlite>
```

Le fichier de données

<spoiler | Afficher le contenu du fichier> Ce fichier a été obtenu en faisant une exportation CSV standard à partir de la table *donnees_mesures2*.

x_y.csv

```
x,y
"1,1","2,4468240195"
"2,1","2,8420511522"
"3,1","0,6243095624"
"4,1","-2,1674193599"
"5,1","-2,9664329183"
"6,1","-1,038121732"
"7,1","1,8446337872"
"8,1","3,0314415094"
"9,1","1,4311558881"
"10,1","-1,4849278566"
"11,1","-3,035775778"
"12,1","-1,7955454494"
"13,1","1,0955010849"
"14,1","2,9793489738"
"15,1","2,1239971562"
"16,1","-0,6841478515"
"17,1","-2,8632904797"
"18,1","-2,4099370456"
"19,1","0,2591013942"
"20,1","2,6899232071"
"21,1","2,6476420286"
"22,1","0,1711309792"
"23,1","-2,4627171032"
"24,1","-2,8323544384"
"25,1","-0,5979381649"
"26,1","2,1862196998"
"27,1","2,9603772548"
1.0,"0,8414709848"
"1,1","0,8912073601"
"1,2","0,932039086"
"1,3","0,9635581854"
"1,4","0,98544973"
```

```
"1,5","0,9974949866"
"1,6","0,999573603"
"1,7","0,9916648105"
"1,8","0,9738476309"
"1,9","0,9463000877"
2.0,"0,9092974268"
"2,1","0,8632093666"
"2,2","0,8084964038"
"2,3","0,7457052122"
"2,4","0,6754631806"
"2,5","0,5984721441"
"2,6","0,5155013718"
"2,7","0,4273798802"
"2,8","0,3349881502"
"2,9","0,2392493292"
3.0,"0,1411200081"
"3,1","0,0415806624"
"3,2","-0,0583741434"
"3,3","-0,1577456941"
"3,4","-0,255541102"
"3,5","-0,3507832277"
"3,6","-0,4425204433"
"3,7","-0,5298361409"
"3,8","-0,6118578909"
"3,9","-0,6877661592"
4.0,"-0,7568024953"
"4,1","-0,8182771111"
"4,2","-0,8715757724"
"4,3","-0,9161659367"
"4,4","-0,9516020739"
"4,5","-0,9775301177"
"4,6","-0,9936910036"
"4,7","-0,9999232576"
```

</spoiler>

Le fichier a été enregistré avec une première ligne contenant les noms des deux colonnes de données.

Insérer les données à partir de python

- Ouvrir la base
- parcourir le fichier csv
- formater la ligne
- insérer des lignes dans la table à partir des données formatées

```
import sqlite3 as lite
import sys
```

```
nomFichier='C:\\pgms\\sqlite3\\livre\\v1\\x_yTmp.csv'
lignes=[ligne.strip() for ligne in open(nomFichier)]

ligne1=lignes.pop(0)
print("==================")

connexion = None

try:
 connexion = lite.connect('C:\\pgms\\sqlite3\\livre\\v1\\essaisImport.db')
 leCurseur = connexion.cursor()
 leCurseur.execute('insert into donnees_mesures2(x,y) values (3,4)')
 for ligne in lignes:
##  Remplacer les .0," par des .0","
    ligne=ligne.replace('.0,\"','.0\",\"')
##  Remplacer les virgules séparatrices de données par des point-virgules
    ligne=ligne.replace("\",\"","\";\"")
##  Remplacer les virgules dans les nombres par des points
    ligne=ligne.replace(',','.')
##  Remplacer les point-virgules de données par des virgules séparatrices
    ligne=ligne.replace(';',',')
##  Remplacer les guillemets par "rien"
    ligne=ligne.replace('"','')
## A partir d'une ligne construire une liste de deux éléments x et y
    x_et_y = ligne.split(',')
## Les convertir en réels
    x_et_y[0]=float(x_et_y[0])
    x_et_y[1]=float(x_et_y[1])
    leCurseur.execute('insert into donnees_mesures2(x,y) values (?,?)',
x_et_y)
    connexion.commit()
except lite.Error as e:
 print ("Error %s:" % e.args[0])
 sys.exit(1)

finally:
if connexion:
   connexion .close()
```

Les lignes de la table

```
sqlite> select * from donnees_mesures2;
x           y             t
----------  ------------  ----------
1.1         2.4468240195
2.1         2.8420511522
3.1         0.6243095624
4.1         -2.167419359
```

```
5.1         -2.966432918
6.1         -1.038121732
7.1         1.8446337872
8.1         3.0314415094
9.1         1.4311558881
10.1        -1.484927856
11.1        -3.035775778
12.1        -1.795545449
13.1        1.0955010849
14.1        2.9793489738
15.1        2.1239971562
16.1        -0.684147851
17.1        -2.863290479
18.1        -2.409937045
19.1        0.2591013942
20.1        2.6899232071
21.1        2.6476420286
22.1        0.1711309792
23.1        -2.462717103
24.1        -2.832354438
25.1        -0.597938164
26.1        2.1862196998
27.1        2.9603772548
1.0         0.8414709848
1.1         0.8912073601
1.2         0.932039086
1.3         0.9635581854
1.4         0.98544973
1.5         0.997494986
1.6         0.999573603
1.7         0.991664810
1.8         0.973847630
1.9         0.946300087
2.0         0.909297426
2.1         0.863209366
2.2         0.808496403
2.3         0.745705212
2.4         0.675463180
2.5         0.598472144
2.6         0.515501371
2.7         0.427379880
2.8         0.334988150
2.9         0.239249329
3.0         0.141120008
3.1         0.041580662
3.2         -0.05837414
3.3         -0.15774569
3.4         -0.25554110
3.5         -0.35078322
3.6         -0.44252044
3.7         -0.52983614
```

```
3.8         -0.61185789
3.9         -0.68776615
4.0         -0.75680249
4.1         -0.81827711
4.2         -0.87157577
4.3         -0.9161659367
4.4         -0.9516020739
4.5         -0.9775301177
4.6         -0.9936910036
4.7         -0.9999232576
```