REINVENT ENGINEERING

CENTRALE NANTES

Qualité, conception, modélisation

Accéder aux données - Python

JY Martin

Novembre 2023

Plan

Se connecter en Python

- API spécifiques
- Des bibliothèques de fonctions pour de nombreux types de base de données
- Nécessite l'importation et l'installation de modules correspondant au type de base de données utilisée

Se connecter en Python : les étapes

Comme la plupart des langages, connexion et exploitation d'une base de donénes se font en plusieurs étapes :

- Importer le module correspondant au SGBD
- Ouvrir une connexion à la base
- Créer un curseur pour une requête SQL
- Exécuter une requête SQL via le curseur
- Récupérer les résultats
- Clore le curseur
- Clore la connexion

Les APIs de connexion

L'API de connexion dépend de la base de données. Il faut :

- Que le module ait été importé (pip install)
  python -m pip install ...
  - mysql : mysql-connector
  - postgresql : postgres
  - oracle : ... ben non, il faut le télécharger depuis le site d'Oracle et l'installer à la main avec les variables d'environnement.
- Que le module soit déclaré en début de script
  import ...
  - mysql : mysql.connector
  - postgresql : psycopg2
  - oracle : cx_oracle

Les APIs de connexion

Si vous utilisez Anaconda

Allez dans “Environnements”. Sélectionnez l’environnement de travail (“base” probablement).

Clic sur la petite flèche à droite du nom de l’environnement. Dans le terminal / invite de commande qui s’est ouvert, tapez :

```
conda install nom-module
```

nom-module = mysql-connector ou psycopg2 ou …

Vérifiez lorsque le script se termine que le moule apparaît dans l’environnement.
Si ce n’est pas la cas, utilisez la procédure du slide précédent.

CENTRALE NANTES

La connexion

La connexion se fait à partir de la fonction **connect**.

4 éléments à indiquer dans le paramètre :

- host : le serveur
- dbname : le nom de la base de données
- user : le login de connexion
- password : son mot de passe

Toute connexion ouverte doit être fermée via **close**

Les curseurs

Pour effectuer une requête on passe par la définition d'un curseur.

- Un curseur est défini à partir d'une connexion à une base de données
- Il s'appuie sur une requête SQL
- Il permet d'exécuter une requête, et d'en récupérer le résultat
- Il permet enfin de récupérer le résultat soit en totalité, soit ligne par ligne.
  - fetchone : la ligne suivante
  - fetchall : toutes les lignes

Tout curseur ouvert doit être fermé (via close) avant d'être à nouveau utilisé.

NB : attention à la nécessité de commit et de rollback de certaines requêtes

Exemple 1

```
import psycopg2
conn = psycopg2.connect("host=localhost dbname=test user=prweb password=prweb")
cursor = conn.cursor()
cursor.execute("SELECT * FROM Personne")
rows = cursor.fetchall()
for row in rows:
    print(row)
cursor.close()
conn.close()
```

Exemple 2

On peut aussi utiliser le curseur dans les boucles

```
import psycopg2
conn = psycopg2.connect("host=localhost dbname=test user=prweb password=prweb")
cursor = conn.cursor()
cursor.execute("SELECT * FROM Personne")
for row in cursor:
        print(row)
cursor.close()
conn.close()
```

Exemple 3

On peut aussi utiliser le curseur dans les boucles

```
import psycopg2
conn = psycopg2.connect("host=localhost dbname=test user=prweb password=prweb")
cursor = conn.cursor()
cursor.execute("INSERT INTO Personne(nom, prenom) VALUES ('DUBOIS', 'Jacques')")
cursor.close()
conn.commit()
conn.close()
```

Parametrer les curseurs : les dictionnaires

Pour éviter les injections SQL, on utilise des dictionnaires en corrélation avec les curseurs

```
cursor = comm.cursor()
data = {"nom" : "DUBOIS", "prenom" : "Jacques"}
cursor.execute("INSERT INTO personne(nom, prenom) VALUES (:nom, :prenom)",
data)
cursor.close()
comm.commit()
```

Exemple

```
import psycopg2

host = "pfe-postgres.ec-nantes.fr"
username = "sinbad"
password = "sinbad"
database = "absence"

print("-----------------------------")
conn = psycopg2.connect("host=" + host + " dbname=" + database + " user=" + username + " password=" + password + "")
cursor = conn.cursor()
data = {"nom": "DUPONT"}
cursor.execute("SELECT * FROM Eleve WHERE nomEleve=%(nom)s", data)
for row in cursor:
  print(row)
cursor.close()
conn.close()
print("-----------------------------")
```

Exemple

```
import psycopg2
import csv

host = "localhost"
username = "prweb"
password = "prweb"
database = "absence"

print("----------------------------")
conn = psycopg2.connect("host=" + host + " dbname=" + database + " user=" + username + " password=" + password + "")

myFile = open('data.csv', "rt")
readFile = csv.reader(myFile, delimiter=";")
for row in readFile :
  if row[0] != 'numeroeleve':
    print(row)
    cursor = conn.cursor()
    cursor.execute("INSERT INTO eleve(numeroeleve, nomeleve, prenomeleve) VALUES (%s, %s, %s)", row)
    cursor.close()
conn.commit()
conn.close()
print("----------------------------")
```

Exemple

```
import psycopg2
import sqlalchemy
from sqlalchemy import create_engine
import csv
import pandas

host = "pfe-postgres.ec-nantes.fr"
username = "sinbad"
password = "sinbad"
database = "absence"

print("-----------------------------")
connStr = "postgresql://" + username + ":" + password + "@" + host + "/" + database + ""
engine = create_engine(connStr)

df = pandas.read_csv("data.csv", sep=';')
print(df)
df.to_sql("eleve", engine, if_exists='append', index=False)
print("-----------------------------")
```


Success
NEXT EXIT