Introduction au python (suite)

Master IMST "Système d'information et d'adchivage numérique" (IMST-SIAN)

Fabien Rico

Univ. Claude Bernard Lyon 1

21 octobre 2022

Plan

Mysql

Pour accéder aux base mysql, python utilise le connecteur mysql paquet `mysql-connector-python` pour pip.

```
import mysql.connector

conn = mysql.connector.connect(
  host="localhost",
  user="lenomdelutilisateur",
  password="lemotdepasse",
  database="lenomdelabase")
cursor = conn.cursor()

#... utilisation de la base

conn.close()
```

Cela permet d'exécuter toutes les requêtes, mais nous n'utiliserons que des intérrogations.

La lecture de données

Il faut générer la requete sous la forme d'une chaine de carractère.

```
import mysql.connector

conn = mysql.connector.connect(
  host="localhost",
  user="lenomdelutilisateur",
  password="lemotdepasse",
  database="lenomdelabase")
cursor = conn.cursor()

#... utilisation de la base

conn.close()
```

Cela permet d'exécuter toutes les requêtes, mais nous n'utiliserons que des intérrogations.

Faire une requete select

```
cur.execute("SELECT * FROM nomtable WHERE 1;")

# affiche les nom de colonnes (c'est un nupplet)
print(cur.column_names)
# ('colonne1', 'colonne2', ...)
results = cur.fetchall()
#
print(results)
# [(val1ligne1, val2ligne1, ...),
#  (val1ligne2, val2ligne2, ...),
#  ...
#  ]
```

- On obtient chaque ligne sous la forme d'un `n-uplet` python.
- Il faut donc connaitre l'ordre des colonnes ou utiliser leur liste donnée par `cur.column_names`.
- `fetchall` permet de récuperer une liste contenant tous les résultats.
- *Une fois qu'ils ont été obtenus*, il est possible de connaitre leur nombre avec `rowcount`

Récupération au fur et à mesure

```
cur.execute("SELECT * FROM nomtable WHERE 1;")

while True:
  res = cur.fetchone()
  if res is None:
    break
  print(res)
  # (val1ligne_n, val2ligne_n, ...)
```

- On peut lire les résultats petit à petit.
- `fetchone()` permet de lire une ligne à la fois, `fetchmany(nb)` permet de lire par lot de `nb` résultats.
- Pourquoi est-ce intéréssant ?

Ajout de parametre

La requète peut être construite à partir de variables :

```
mot = "math"
requete = f"SELECT * FROM Courses WHERE Courses.title = '{mot}'"
cur.execute(requete)
# exécution de la requete
# SELECT * FROM Courses WHERE Courses.title = 'math'
```

- La requete est une chaine de caractère, je peux utiliser les outils python pour créer celle-ci à partir de variable
- Dans l'exemple, `{name}` est remplacé par la valeur de la variable `name`
- Mais le forma de la requete est très contraint, que ce passe-t'il si la variable contient le carractère ' ?

```
mot = "l'experience"
requete = f"SELECT * FROM Courses WHERE Courses.title = '{mot}'"
cur.execute(requete)
# erreur la requete
# SELECT * FROM Courses WHERE Courses.title = 'l'experience'
# est invalide
```

Injection SQL

Lorsqu'un programme utilise des variables pour construire les requetes, il est possible que le contenu des variables *modifie la structure de la requete*.

- En effet, ces variable peuvent très bien contenir des caractères spéciaux de mysql.
- Si le contenu de la variable provient d'un utilisateur du programme, ce dernier peut par exemple l'écrire de manière a exécuter une autre requete.
- Cette dernière sera exécutée comme si elle provenait de votre programme.
- C'est ce qu'on appelle *l'injection SQL*
- En plus des problèmes de sécurité, les carractères spéciaux peuvent simplement déteriorer les données
  - problème de l'encodage des accents
  - erreur dans l'exécution de la requète
  - perte d'une partie de l'information

En général, rien de ce qui provient des utilisateur ou d'une source extérieure non maitrisée ne peut être transmis dans une requête sans que les éventuels carractères spéciaux soient modifiés.

Paremetre des requetes

Il existe en des fonctions pour rendre compatible les carractères spéciaux, mais ces dernier dépendent du système de base de données, de la configuration de la base ... Le plus efficace est donc d'utiliser les outils de la bibliothèque qui sont justement prévus pour.
Par exemple, en mysql, on peut utiliser des substitutions dans les requêtes :

```
name = ...
requete = f"SELECT * FROM Courses WHERE Courses.title = '%(valeur)s'"
cur.execute(requete, {"valeur": name})
```

Les autres base de données

Il y a un grand nombre de base de données. `python` propose plusieurs librairie pour les utiliser

- `Psycopg` pour les bases de données postgresql
- `cx_Oracle` pour les bases oracles
- `pyodbc` pour les différentes bases dont Microsorf sql server
- `sqlite3` pour les base sqlite

Ces librairies ont un fonctionnement similaire mais les différences rendent très difficile de maintenir un code qui fonctionnerait de manière transparente sur différents serveur.

ORM

Un ORM (*Object Relational Mapping*) est une bibliothèque qui se place entre votre programme et la base de données

- Il gère directement la sauvegarde et la récupération des objets.
- Il s'occupe de créer et modifier les tables.
- Il est capable d'utiliser plusieurs serveurs différents.
- Mais cela réduit les performances, parfois une simple requête est bien plus efficace.