Python et SQL

Ihab ABADI - Antoine DIEUDONNE/ UTOPIOS

Python et interaction base de données

- Il existe de nombreuses bonnes raisons d'utiliser Python pour programmer des applications qui interagissent avec de base de données :
- Python est célèbre pour sa portabilité.
- Il est indépendant de la plate-forme.
- Python prend en charge les curseurs SQL.
- Dans de nombreux langages de programmation, le développeur d'applications doit prendre en charge les connexions ouvertes et fermées de la base de données, afin d'éviter d'autres exceptions et erreurs. En Python, ces connexions sont prises en charge.
- Python prend en charge les systèmes de bases de données relationnelles.
- Les API de base de données Python sont compatibles avec diverses bases de données, il est donc très facile de migrer des base de données

DB-API (SQL-API) pour Python

- Python DB-API est indépendant de tout moteur de base de données, elle permet d'écrire des scripts Python pour accéder à n'importe quel moteur de base de données.
- L'implémentation de l'API Python DB pour MySQL est MySQL.
- Pour PostgreSQL, il prend en charge psycopg, PyGresQL et pyPgSQL Modules.
- Les implémentations DB-API pour Oracle sont dc_oracle2 et cx_oracle.
- Pydb2 est l'implémentation DB-API pour DB2.
- La DB-API de Python se compose d'objets de connexion, d'objets de curseur, d'exceptions standard et de certains autres contenus de module

DB-API (SQL-API) pour Python

- L'API de base de données fournit standard pour interagir avec des bases de données utilisant des structures et une syntaxe Python. Cette API inclut les éléments suivants :
- Importation du module API.
- Acquisition d'une connexion avec la base de données.
- Émission d'instructions SQL et de procédures stockées.
- Fermeture de la connexion.

Fonctions et Attributs de DB-API

- connect(parameters...)Constructor pour créer un connexion à la base de données.
- Renvoie un objet de connexion.
- La valeur par défaut de chaque paramètre est NULL ou zéro
- Les paramètres importants sont :
- Hôte : nom de l'hôte auquel se connecter. Par défaut : utilisez l'hôte local via un
- Socket UNIX (le cas échéant)
- User.
- Password.
- database.
- Port Par défaut : port standard (3306)

```
import mysql.connector

db = mysql.connector.connect(
host="localhost",
user="login",
password="mot_de_passe",
database="demo_python"
)
```

Méthode de l'objet connection

- Les objets de connexion sont renvoyés par la fonction connect().
- commit() : Si la base de données et les tables supportent les transactions, cela valide la transaction en cours ; sinon cette méthode ne fait rien.
- rollback() : Si la base de données et les tables supportent les transactions, cela annule la transaction en cours ; sinon un NotSupportedError est déclenché.
- cursor([cursorclass]).
- begin() pour démarrer une transaction, chaque exécution de requête démarre une transaction implicite.

Méthode de l’objet cursor

- close() :Ferme le curseur. Les opérations futures génèrent ProgrammingError, il est très important de fermer le curseur lorsque avant d'en créer un nouveau.
- insert_id() : Renvoie la dernière valeur du champ AUTO_INCREMENT insérée dans base de données
- info() : Retourne des informations sur la dernière requête. soulevé. Si vous utilisez une classe de curseur sans avertissements.
- nextset() :Avance le curseur au jeu de résultats suivant, en supprimant le reste lignes dans le jeu de résultats actuel. S'il n'y a pas d'ensembles de résultats supplémentaires, elle renvoie none.

Méthode de l'objet cursor

- Cursor.execute(operation[, parameters])
  - Permet de préparer et exécuter une opération sur une base de données (requête ou commande)
  - Paramètres : paramètre de la requête ou la commande.
  - Renvoie :le curseur, vous pouvez donc enchaîner les commandes
  - Pour exécuter une requête préparée, nous utiliserons le paramètre prepared du cursor à true
- Cursor.executemany(operation[, seq_of_parameters])
  - Permet de préparer et exécuter de nombreuses opérations sur une base de données (requêtes ou commandes)
  - Paramètres :– une séquence ou mappage de tuples de paramètres
  - Renvoie : le curseur, vous pouvez donc enchaîner les commandes

Méthode de l'objet cursor

```
## Sans paramètre
with db.cursor(Prepared=True) as c:
c.execute("insert into utilisateur (nom, prenom) values ('toto', 'tata')")
db.commit()

##Avec paramètre
request = """insert into utilisateur
(nom, prenom)
values (%s, %s)"""
params = ("toto", "tata")
with db.cursor() as c:
c.execute(request, params)
db.commit()

##Avec execute many
request = """insert into utilisateur
(nom, prenom)
values (%s, %s)"""
params = [("toto", "tata"),("titi", "minet")]
with db.cursor() as c:
c.executemany(request, params)
db.commit()
```

Méthode de l'objet cursor

- Cursor.callproc(self, procname, [parameters]) :
  - appelle un procédure stockée d'une base de données avec le nom donné
  - procname (str) – le nom de la fonction de base de données
  - paramètres – une séquence de paramètres (peut être vide ou omis)
  - La séquence de paramètres doit contenir une entrée pour chaque entrée argument que la fonction attend.
  - Le résultat de l'appel est le même que celui de la procédure
  - La fonction peut également fournir un ensemble de résultats en sortie. Ceux-ci peuvent être demandé via les méthodes de récupération standard du cursor.

Méthode de l’objet cursor

- Cursor.fetchone()
  - Récupère la ligne suivante d'un ensemble de résultats de requête
  - Renvoie : la ligne suivante du jeu de résultats de la requête
  - Type de retour : tuple nommé ou None Récupère la ligne suivante d’un ensemble de résultats de requête, renvoyant un seul tuple nommé, ou Aucun lorsqu'il n'y a plus de données disponibles.
  - Les noms de champ du tuple nommé sont les mêmes que les noms de colonne de la requête sur une base de données tant qu'ils sont valides
  - Une exception Error (ou sous-classe) est déclenchée si l'appel précédent à Cursor.execute() ou Cursor.executemany() n'a pas produire un ensemble de résultats

Méthode de l'objet cursor

```
request = "select nom, prenom from utilisateur"
with db.cursor() as c:
c.execute(request)
while True:
utilisateur = c.fetchone()
if utilisateur is None:
break
print(utilisateur)
```

Méthode de l’objet cursor

- curseur.fetchall()
  - Récupère toutes les lignes (restantes) d'une requête résultat
  - Renvoie : l'ensemble de toutes les lignes du résultat de la requête
  - Type de retour : liste de tuples nommés Récupérer toutes les lignes d’une de requête, les renvoyant sous forme de liste de noms tuples.
  - Les noms de champ du tuple nommé sont les mêmes que les noms de colonne de la requête de base de données tant qu'ils sont valides.

Méthode de l'objet cursor

```
request = "select nom, prenom from utilisateur"
with db.cursor() as c:
c.execute(request)
resultats = c.fetchall()
for utilisateur in resultats:
print(utilisateur)
```

Méthode de l’objet cursor

- Cursor.fetchmany([size=None][, keep=False])
  - Récupère l'ensemble de lignes suivant d'un résultat de requête
  - Paramètres : size (int ou None) – le nombre de lignes à récupérer
  - keep - si défini sur true, conservera la taille de tableau transmise
  - Renvoie : le prochain ensemble de lignes du résultat de la requête
  - Type de retour : liste de tuples nommés. récupère le prochain ensemble de lignes d’une requête, renvoyant une liste de tuples nommés. Une séquence vide est renvoyée lorsqu'il n'y en a plus lignes sont disponibles. Les noms de champ du tuple nommé sont les noms de colonne de la requête sur la base de données tant qu’ils sont des identifiants valides

Méthode de l'objet cursor

```
request = "select nom, prenom from utilisateur"
with db.cursor() as c:
c.execute(request)
while True:
resultats = c.fetchmany(10)
if not resultats:
break
for utilisateur in resultats:
print(utilisateur)
```

Méthode de l’objet cursor

- La méthode execute permet d’exécuter des requêtes pour créer ou altérer des entités de base de données.
- Table, Vue, … .

Type d’erreurs

- En cas d’erreur d’exécution, une exception est levée.
- Voici les exceptions les plus utilisées :
  - DataError :
    - Cette exception est levée en raison de problèmes avec les données traitées (par exemple, valeur numérique hors plage, division par zéro, etc).
  - IntegrityError
    - Si l'intégrité relationnelle de la base de données est en jeu (par exemple un échec de la vérification de la clé étrangère, clé dupliquée, etc.), cette exception est soulevé
  - InternalError
    - Cette exception est déclenchée lorsqu'il y a une erreur interne dans MySQL base de données elle-même (par exemple, un curseur invalide, la transaction est désynchronisé, etc.)
  - NotSupportedError
    - MySQL pour Python lève cette exception lorsqu'une de l’API n’est pas prise en charge
  - ProgrammingError
    - Exception déclenchée pour les erreurs de programmation réelles (pour exemple, une table est introuvable ou existe déjà, il y a une erreur de syntaxe dans l'instruction MySQL, un un nombre incorrect de paramètres est spécifié, et donc sur.)

Exercice

Exercice

- Réalisez une application d'annuaire au moyen de MySQL et de `mysql.connector`
- L'application devra permettre un CRUD basique en rapport avec une liste de contacts. Il vous sera possible de voir, modifier, supprimer et ajouter des contacts.
- Les contacts seront constitués d'un prénom, d'un nom, d'une date de naissance, d'un numéro de téléphone et d'une adresse mail.
- Un menu permettra de naviguer entre les différentes sections de l'application

Exercice

TP

Réalisez une application chenil au moyen de MySQL et de `mysql.connector`

- Vous réaliserez une application de chenil permettant à un utilisateur de manipuler des chiens et des propriétaires afin de permettre l'adoption ou le retour du meilleur ami de l'homme
- Les chiens devront avoir comme attributs leur nom, leur race, leur date de naissance, leur âge, leur genre et l'id de leur maître en cas d'adoption
- Les propriétaires devront avoir de leur côté un prénom, un nom, une date de naissance, un âge, un genre, un numéro de téléphone et un email
- Les adoptions se feront un chien à la fois, de même que pour les retour.

L'application disposera de plusieurs menus et sous-menus pour permettre la navigation dans les différentes catégories

- Le menu de gestion des chiens ainsi que celui des maître devront permettre le visionnage, l'ajout, la modification et la suppression des ensembles qui les concernent
- De son côté, le menu de gestion des adoptions devra permettre d'accéder à la liste des chiens disponibles à l'adoption, d'en réaliser une ou de faire un retour

Bien entendu, les changements effectués dans le programme devront être retranscris en base de données