SQLite 8-9

Python and SQLite

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Sunday 23 April 2017

Table of Contents

Introduction

This report is about using Python with SQLite, a lightweight SQL database library. All files for this hand in (and all others) are available at:
https://github.com/deadbok/eal_programming/

8. How to use SQLite with a Python application

The documentation the Python 3 SQLite module is available at:
https://docs.python.org/3/library/sqlite3.html.

To use SQLite in Python3:

- Import the sqlite3 module.
- Use sqlite3.connect() to connect to the database file, the file name is given as an argument. This creates a Connection object.
- Create a Cursor object by calling the cursor() method on the Connection object.
- Use this cursor to perform SQL actions using the method execute() from the Cursor object. This method takes an SQL command as argument.
- To commit the changes to the database call the commit() method of the Cursor object.

The SQL commands that are execute to create the database and insert the data are the same as in the last assignment, which is one of the points of SQL. If the program was ported to C and still using SQLite the SQL commands would remain unchanged. There are SQL syntax differences between implementations, SQLite, MySQL, PostgreSQL, and Microsoft SQL server all have variations and extensions.

create.py

```
# -*- coding: utf-8 -*-
"""
Name: SQLite example of creating tables and inserting rows.
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-04-22)
"""

# Import the SQLite module
import sqlite3

# Start an exception block to catch errors.
try:
    # Create a connection to a database with the file name 'example.db'. The
    # file is created if missing.
    conn = sqlite3.connect('example.db')

    # Get a Cursor object to used to perfom SQL commands on the database.
    cursor = conn.cursor()

    # Create the customerTable table.
    print('Creating customerTable...', end='')
    cursor.execute(
        """
        CREATE TABLE customerTable(
          idCust INT NOT NULL UNIQUE,
          name VARCHAR(100),
          email VARCHAR(50),
          address VARCHAR(50),
          city VARCHAR(30),
          PRIMARY KEY (idCust)
        )
```

```
        """
    )

    # Insert the data in to the table.
    cursor.execute(
        """
        INSERT INTO customerTable VALUES(
        1,
        'Per',
        'pda@eal.dk',
        'Mystreet1',
        'Odense'
        )
        """
    )
    # Can we make the street name Opampstreet to fit the number? ;-)
    cursor.execute(
        """
        INSERT INTO customerTable VALUES(
        2,
        'Artur',
        'at@hotmail.com',
        'Allstreet 741',
        'Vilnius'
        )
        """
    )
    cursor.execute(
        """
        INSERT INTO customerTable VALUES(
        3,
        'Alice',
        'ab@gmail.com',
        'Topstreet 56',
        'London'
        )
        """
    )
    # Commit the data to the database.
    conn.commit()
    print('done')

    print('\nCurrent entries in the database:')
    # Select all rows in the customerTable table.
    cursor.execute('SELECT * FROM customerTable')
    # A counter to make things look nice in the output.
    row_nr = 0
    # Run trhough all rows.
    for row in cursor.fetchall():
        # Next row number.
        row_nr += 1
        # Print row number.
        print('Row {}: '.format(row_nr), end='')
        # Run through all fields in the row.
        for field in row:
            # Print the fields.
            print('{}, '.format(field), end='')
        # Next line.
        print()
```

```
except sqlite3.Error as sql_error:
    print('\n\nAn database error occurred!')
    print('Error is: ' + str(sql_error))
```

Result

```
$ python3 create.py
Creating customerTable...done

Current entries in the database:
Row 1: 1, Per, pda@eal.dk, Mystreet1, Odense,
Row 2: 2, Artur, at@hotmail.com, Allstreet 741, Vilnius,
Row 3: 3, Alice, ab@gmail.com, Topstreet 56, London,
```

Successfully creating the database and inserting the data.

The program uses an exception handler to catch errors during the SQL operations, an error where the database exists when the program is called is illustrated below:

```
$ python3 create.py
Creating customerTable...

An database error occurred!
Error is: table customerTable already exists
```

Example of an error when creating the database.

8. Updating data in the database

This program reuses the structure of the above. The printing of the entries in the database has been moved in to a function, since it is called twice to show that the contents of the database has changed.

The SQL command for updating is UPDATE:

```
cursor.execute('UPDATE customerTable SET address = ? WHERE idCust = ?',
('Opampstreet 741', 2))
```

This line updates the ‘address’ field of the ‘customerTable’ where ‘idCust’ is equal to 2. Both the data to update and the value of the WHERE clause, are supplied as a tuple. The address field is replaced with ‘Opampstreet 741’.

update.py

```
# -*- coding: utf-8 -*-
"""
Name: SQLite example of updating tables.
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-04-22)
"""

# Import the SQLite module
import sqlite3


def printAllEntries():
    """
    Print all entries in the database.
    """
    print('\nCurrent entries in the database:')
    # Select all rows in the customerTable table.
    cursor.execute('SELECT * FROM customerTable')
    # A counter to make things look nice in the output.
    row_nr = 0
    # Run trhough all rows.
    for row in cursor.fetchall():
        # Next row number.
        row_nr += 1
        # Print row number.
        print('Row {}: '.format(row_nr), end='')
        # Run through all fields in the row.
        for field in row:
            # Print the fields.
            print('{}, '.format(field), end='')
        # Next line.
        print()


# Start an exception block to catch errors.
try:
    # Create a connection to a database with the file name 'example.db'. The
    # file is created if missing.
    conn = sqlite3.connect('example.db')
```

```
    # Get a Cursor object to used to perfom SQL commands on the database.
    cursor = conn.cursor()

    # Print all entries before changing them.
    printAllEntries()

    #Update the second and third fields address information.
    print('\nUpdating data in the database...', end='')
    # Yes we can have crummy chip references.
    cursor.execute('UPDATE customerTable SET address = ? WHERE idCust = ?',
                   ('Opampstreet 741', 2))

    cursor.execute('UPDATE customerTable SET address = ? WHERE idCust = ?',
                   ('Timerstreet 555', 3))
    # Commit the data to the database.
    conn.commit()
    print('done')

    # Print all entries again to show the changes.
    printAllEntries()


except sqlite3.Error as sql_error:
    print('\n\nAn database error occurred!')
    print('Error is: ' + str(sql_error))
```

Result

```
$ python3 update.py

Current entries in the database:
Row 1: 1, Per, pda@eal.dk, Mystreet1, Odense,
Row 2: 2, Artur, at@hotmail.com, Allstreet 741, Vilnius,
Row 3: 3, Alice, ab@gmail.com, Topstreet 56, London,

Updating data in the database...done

Current entries in the database:
Row 1: 1, Per, pda@eal.dk, Mystreet1, Odense,
Row 2: 2, Artur, at@hotmail.com, Opampstreet 741, Vilnius,
Row 3: 3, Alice, ab@gmail.com, Timerstreet 555, London,
```

Successfully changing the data in the database.

The program uses an exception handler to catch errors during the SQL operations, an error where the database does not exists when the program is called is illustrated below:

```
$ python3 update.py

Current entries in the database:


An database error occurred!
Error is: no such table: customerTable
```

An example of an error in the update program.