How to connect to MySQL using Python

Python provides several ways to connect to a MySQL database and process data. This article describes three methods.

.

▼ Table of Contents

Connecting to MySQL using Python

Before you can access MySQL databases using Python, you must install one (or more) of the following packages in a virtual environment:

- *MySQL-python*: This package contains the **MySQLdb** module, which is written in C. It is one of the most commonly used Python packages for MySQL.
- *mysql-connector-python*: This package contains the **mysql.connector** module, which is written entirely in Python.
- *PyMySQL*: This package contains the **pymysql** module, which is written entirely in Python. It is designed to be a drop-in replacement for the *MySQL-python* package.

All three of these packages use Python's portable SQL database API. This means that if you switch from one module to another, you can reuse almost all of your existing code (the code sample below demonstrates how to do this).

Setting up the Python virtual environment and installing a MySQL package

To set up the Python virtual environment and install a MySQL package, follow these steps:

1. Log in to your account using SSH.
2. To create a virtual environment, type the following commands:

   ```
   cd ~
   virtualenv -p /usr/bin/python2.7 sqlenv
   ```

   The *virtualenv* command creates a virtual environment named *sqlenv*, and subsequent commands in this procedure assume that the environment is named *sqlenv*. You can use any environment name you want, but make sure you replace all occurrences of *sqlenv* with your own environment name.

3. To activate the virtual environment, type the following command:

   ```
   source sqlenv/bin/activate
   ```

   The command prompt now starts with **(sqlenv)** to indicate that you are working in a Python virtual environment. All of the following commands in this procedure assume that you are working within the virtual environment. If you log out of your SSH

session (or deactivate the virtual environment by using the **deactivate** command), make sure you reactivate the virtual environment before following the steps below and running the sample code.

4. Type the command for the package you want to install:

    - To install the *MySQL-python* package, type the following command:

      ```
      pip install MySQL-python
      ```

    - To install the *mysql-connector-python* package, type the following command:

      ```
      pip install mysql-connector-python
      ```

    - To install the *pymysql* package, type the following command:

      ```
      pip install pymysql
      ```

Code sample

After you install a MySQL package in the virtual environment, you are ready to work with actual databases. The following sample Python code demonstrates how to do this, as well as just how easy it is to switch between the different SQL package implementations.

In your own code, replace *USERNAME* with the MySQL database username, *PASSWORD* with the database user's password, and *DBNAME* with the database name:

```
#!/usr/bin/python

hostname = 'localhost'
username = 'USERNAME'
password = 'PASSWORD'
database = 'DBNAME'

# Simple routine to run a query on a database and print the results:
def doQuery( conn ) :
    cur = conn.cursor()

    cur.execute( "SELECT fname, lname FROM employee" )

    for firstname, lastname in cur.fetchall() :
        print firstname, lastname


print "Using MySQLdb…"
import MySQLdb
myConnection = MySQLdb.connect( host=hostname, user=username, passwd=password,
db=database )
doQuery( myConnection )
myConnection.close()

print "Using pymysql…"
import pymysql
myConnection = pymysql.connect( host=hostname, user=username, passwd=password,
db=database )
doQuery( myConnection )
myConnection.close()

print "Using mysql.connector…"
import mysql.connector
myConnection = mysql.connector.connect( host=hostname, user=username,
passwd=password, db=database )
doQuery( myConnection )
myConnection.close()
```

This example creates a series of **Connection** objects that opens the same database using different MySQL modules. Because all three MySQL modules use the portable SQL database API interface, they are able to use the code in the **doQuery()** function without any modifications.

When you have a **Connection** object associated with a database, you can create a **Cursor** object. The **Cursor** object enables you to run the **execute()** method, which in turn enables you to run raw SQL statements (in this case, a *SELECT* query on a table named *employee*).

As you can see, Python's portable SQL database API makes it very easy to switch between MySQL modules in your code. In the sample above, the only code changes necessary to use a different module are to the **import** and **connect** statements.