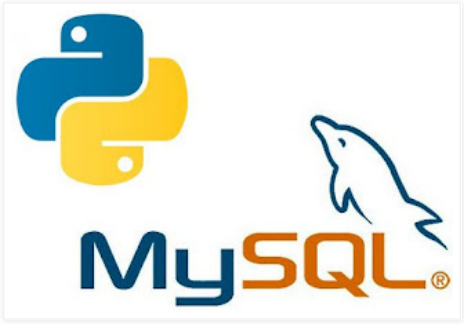
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

domingo, 13 de mayo de 2018

Bases de datos MySQL (y MariaDB) con PyMySQL



MySQL es un popular sistema de gestión de bases de datos multiplataforma de tipo relacional que proporciona un excelente rendimiento, seguridad y flexibilidad.

MySQL lo utilizan en infinidad de empresas y desde hace años es uno de los productos estrellas en el juego de herramientas LAMP para servidores y aplicaciones web.

Por su parte Python, debido a su carácter todoterreno, es una de las opciones para trabajar con MySQL y algunos frameworks basados en este lenguaje, como Django o Pyramid, soportan también esta base de datos. MySQL da soporte a aplicaciones muy conocidas de Internet: desde la Wikipedia a las redes sociales Reddit, Twitter, Youtube... y muchos más.

En la sección de anexos de *Python para impacientes* hemos incluido una **Guía urgente de MySQL** con información esencial para un programador Python para instalar MySQL en un servidor GNU/Linux Debian/Ubuntu y operar con una base de datos desde el Shell. La base de datos `personal` que se utiliza en los ejemplos de la guía es la que vamos utilizar también en los ejemplos de este artículo. Si no tienes experiencia con MySQL recomendamos la realización de los ejemplos que se proponen en dicha guía.

A continuación, contando la infraestructura necesaria instalada, con MySQL funcionando, vamos a mostrar varios ejemplos que utilizan el paquete Python **PyMySQL** para conectarse y operar con la base de datos `personal` de la mencionada guía.

PyMySQL es un paquete desarrollado por el japonés Yutaka Matsubara que permite trabajar tanto con bases de datos del gestor **MySQL** como de **MariaDB**, un derivado del primero con licencia GPL que utiliza los mismos comandos, interfaces, APIs y bibliotecas.

Instalar PyMySQL con PIP

Para instalar el paquete **PyMySQL**:

```
$ pip install pymysql
```

Conectar con base de datos y obtener datos de una tabla

El siguiente ejemplo muestra cómo conectar con PyMySQL con la base de datos `personal` y leer todos los registros de la tabla `Usuarios`:

```
import pymysql

# Conectar con base de datos
conexion = pymysql.connect(host="localhost",
                           user="alejandro",
                           passwd="2018_alejandro",
                           database="personal")
```

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

```
cursor = conexion.cursor()

# Recuperar registros de la tabla 'Usuarios'
registros = "SELECT * FROM Usuarios;"

# Mostrar registros
cursor.execute(registros)
filas = cursor.fetchall()
for fila in filas:
   print(fila)

# Finalizar
conexion.commit()
conexion.close()
```

Añadir una tabla nueva

Este ejemplo muestra el modo de agregar una tabla llamada **`Oficinas`** a la base de datos **`personal`** con PyMySQL.

```
import pymysql

# Conectar con base de datos
conexion = pymysql.connect(host="localhost",
                           user="alejandro",
                           passwd="2018_alejandro",
                           database="personal")
cursor = conexion.cursor()

# Agregar nueva tabla 'Oficinas' a la base de datos 'personal'
TablaOficinas = """CREATE TABLE Oficinas(
denom CHAR(20),
provin CHAR(10),
PRIMARY KEY (denom))"""

cursor.execute(TablaOficinas)
print("Se ha agregado la tabla 'Oficinas' a la base de datos")

# Cerrar conexión
conexion.close()
```

Insertar registros en la tabla nueva

En el código siguiente se insertan tres registros en la tabla **`Oficinas`**.

```
import pymysql

# Conectar con base de datos
conexion = pymysql.connect(host="localhost",
                           user="alejandro",
                           passwd="2018_alejandro",
                           database="personal")
cursor = conexion.cursor()

# Definir comandos para insertar registros
registro1 = "INSERT INTO Oficinas VALUES ('Central', 'Sevilla');"
registro2 = "INSERT INTO Oficinas VALUES ('Norte', 'Bilbao');"
registro3 = "INSERT INTO Oficinas VALUES ('Extremadura', 'Badajoz');"

# Ejecutar comandos
cursor.execute(registro1)
cursor.execute(registro2)
cursor.execute(registro3)

# Finalizar transacción y cerrar
conexion.commit()
conexion.close()
```

Mostrar tablas y obtener datos de la tabla nueva

En el siguiente ejemplo se listan las tablas de **`personal`** y los registros de la tabla **`Oficinas`**.

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

mayo 2018 (1)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

```
import pymysql

# Conectar con base de datos
conexion = pymysql.connect(host="localhost",
                           user="alejandro",
                           passwd="2018_alejandro",
                           database="personal")
cursor = conexion.cursor()

# Recuperar registros de la tabla 'Oficinas'
tablas = "SHOW TABLES;"
registros = "SELECT * FROM Oficinas;"

# Mostrar tablas
cursor.execute(tablas)
filas = cursor.fetchall()
print("Tablas de 'personal':")
for fila in filas:
   print(fila)

# Mostrar registros
cursor.execute(registros)
filas = cursor.fetchall()
print("Registros de 'Oficinas':")
for fila in filas:
   print(fila[0], ":", fila[1])

# Finalizar
conexion.commit()
conexion.close()
```

Borrar una tabla

Por último, un ejemplo para borrar la tabla `Oficinas` de la base de datos.

```
import pymysql

# Conectar con base de datos
conexion = pymysql.connect(host="localhost",
                           user="alejandro",
                           passwd="2018_alejandro",
                           database="personal")
cursor = conexion.cursor()

# Construir comando para borrar tabla
BorrarTabla = "DROP TABLE IF EXISTS Oficinas;"

# Borrar tabla
cursor.execute(BorrarTabla)
print("Se ha suprimido una tabla de la base de datos")

# Finalizar transacción y cerrar
conexion.commit()
conexion.close()
```

Puedes consultar información de las funciones, clases y métodos de PyMySQL en la documentación de ReadTheDocs.

Ir al índice del tutorial de Python

Entrada más reciente Inicio Entrada antigua