

GUÍA 3

MYSQL Y PYTHON

MySQL : Base de datos desde Python

Cuando tenemos que almacenar gran cantidad de datos y su posterior procesamiento es muy común utilizar un gestor de bases de datos.

Con Python podemos comunicarnos con un gestor de bases de datos para enviar y recuperar datos.

Existen gran cantidad de gestores de bases de datos y el primero que veremos para ver cual es la mecánica para conectarnos desde Python será el gestor de base de datos MySQL.

Deberemos instalar primero si no lo tenemos a MySQL.

Instalación de MySQL

Para facilitar la administración del MySQL utilizaremos el programa XAMPP que entre otros instala:

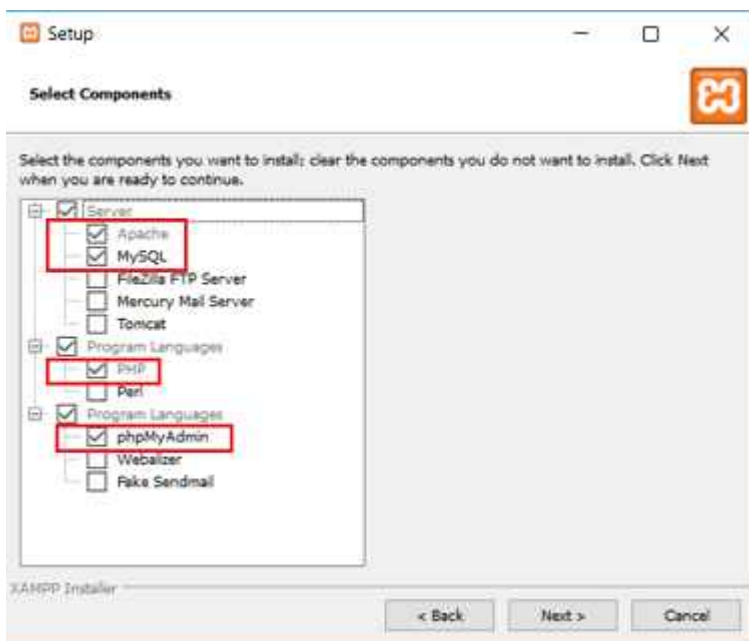
- MySQL
- PHPMyAdmin (que nos permitirá administrar las bases de datos existentes en MySQL)
- PHP (Lenguaje que nos permite ejecutar el PHPMyAdmin)
- Apache (Servidor Web que nos permite ejecutar PHPMyAdmin y PHP en un servidor)

Descargamos e instalamos [XAMPP](#) para el sistema operativo que estamos trabajando.

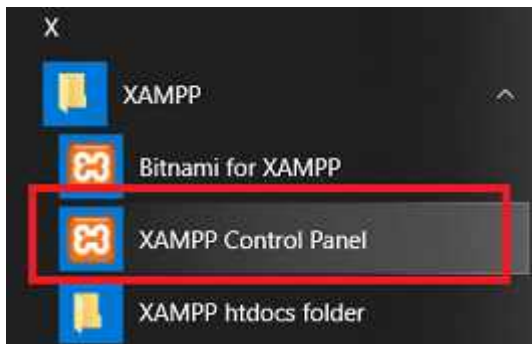
Los pasos para instalar son muy sencillos:



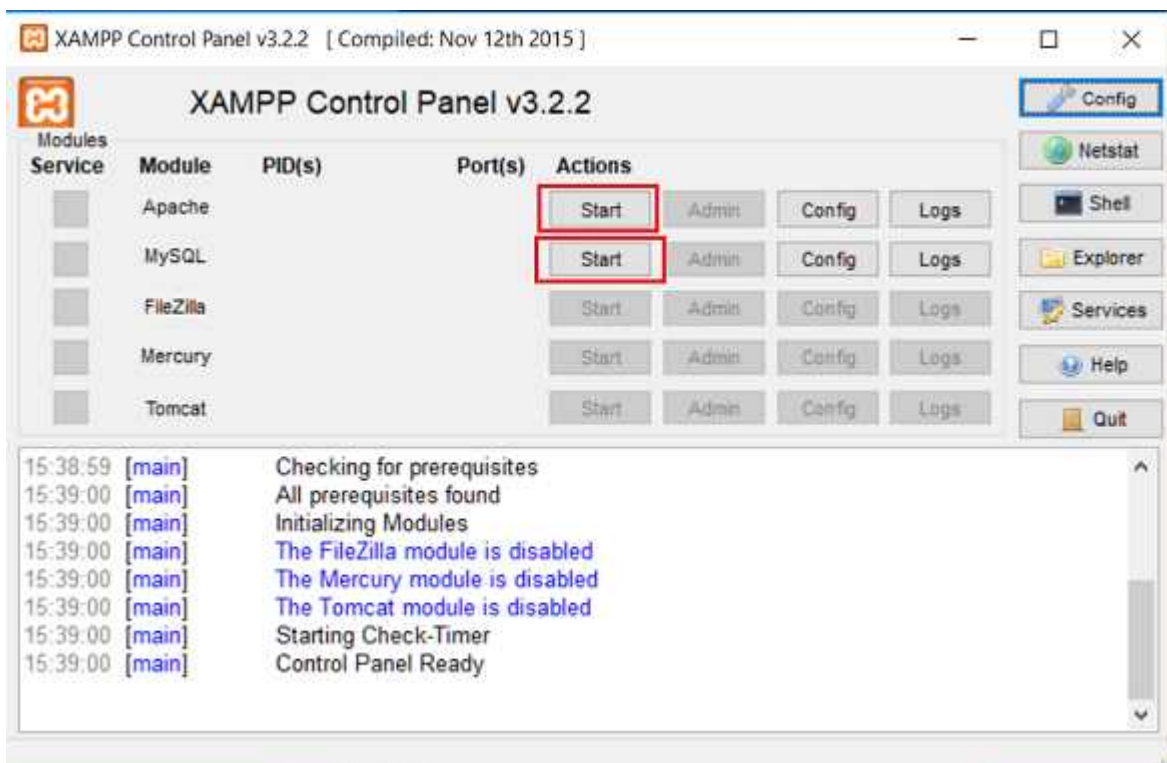
Lo mínimo que debemos instalar es:



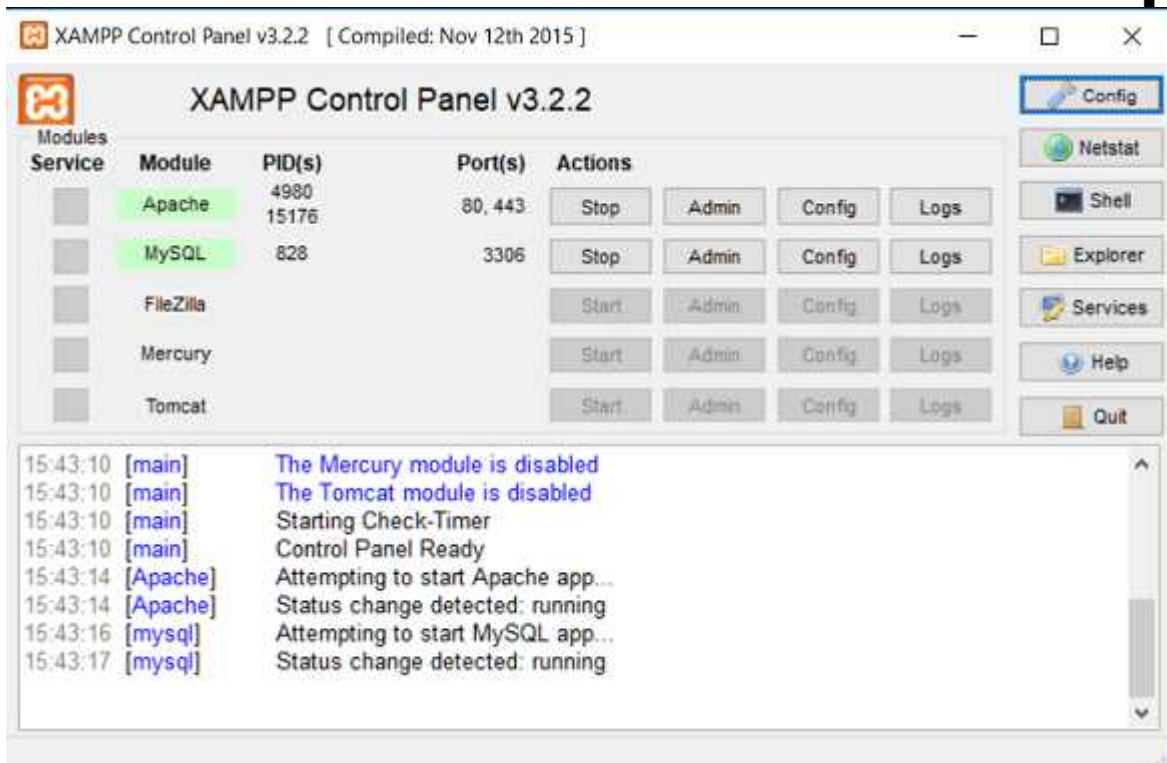
Una vez finalizada la instalación de todo el software debemos arrancar el programa "XAMPP Control Panel":



Aparece la interfaz desde donde debemos iniciar el gestor de base de datos MySQL:



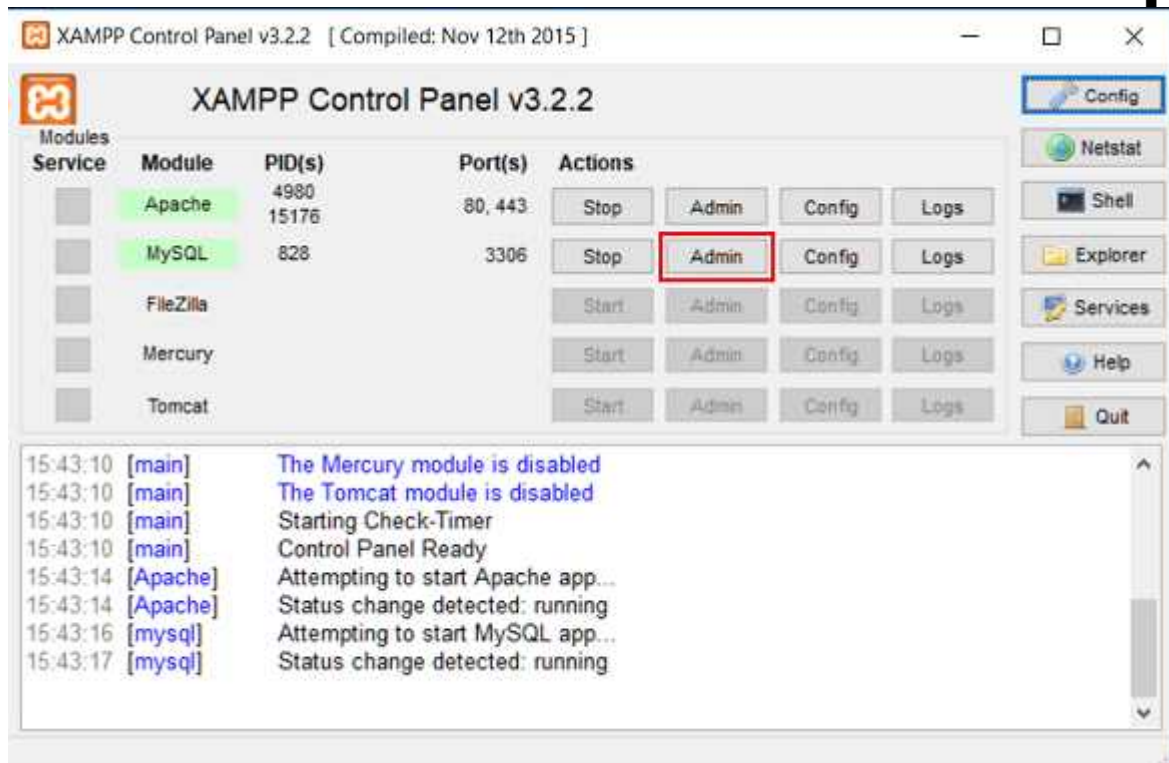
Presionamos el botón "Start" tanto para MySQL como para Apache:



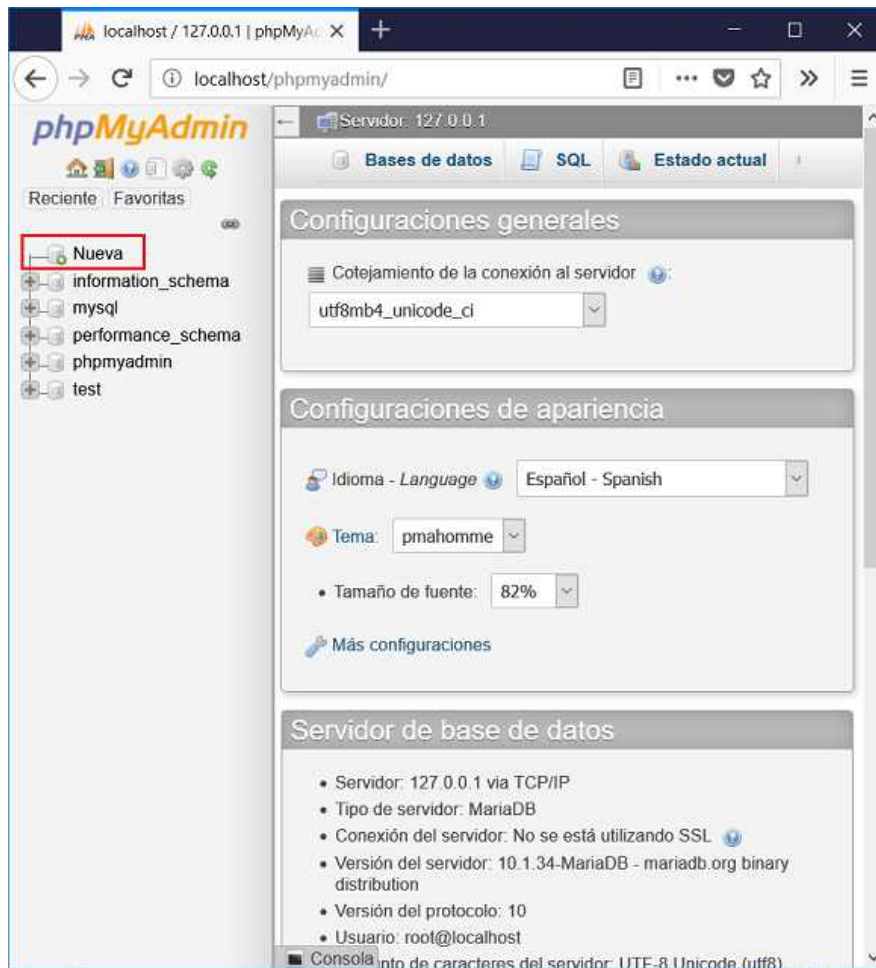
Si aparecen en verde significa que los programas se encuentran correctamente en funcionamiento.

Creación de la base de datos MySQL

Para crear la base de datos utilizaremos el programa PHPMyAdmin que lo iniciamos presionando el botón "MySQL - Admin" del "XAMPP Control Panel":



Se nos abre una aplicación web PHPMyAdmin que nos permite administrar nuestras bases de datos de MySQL:

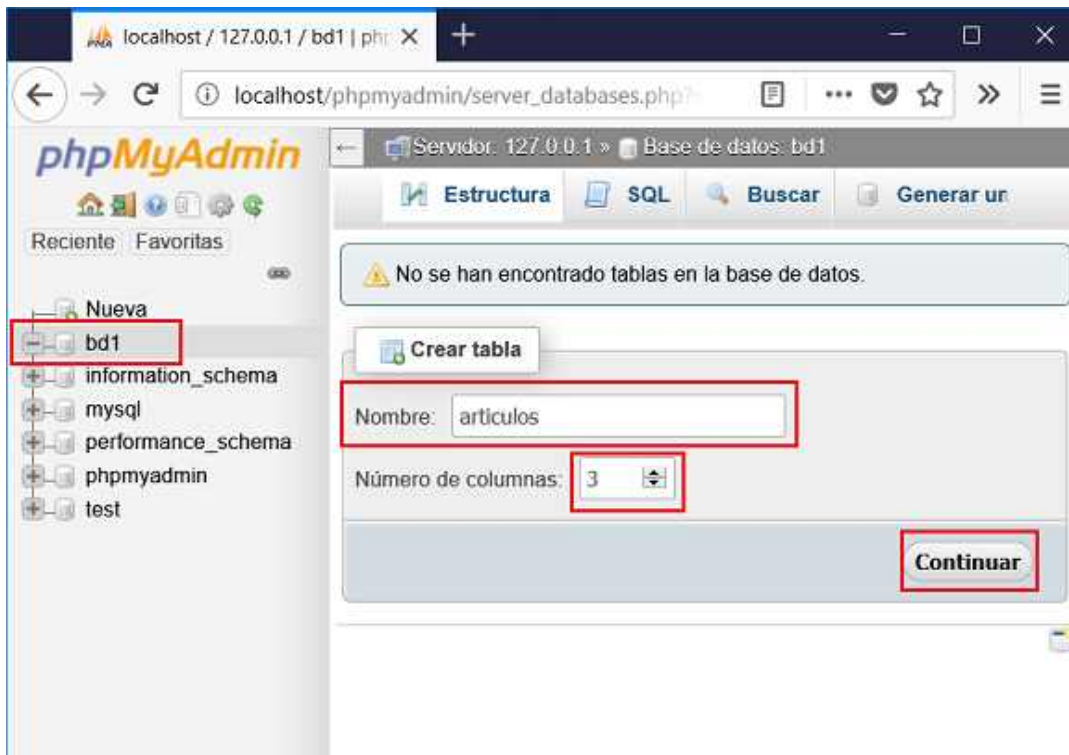


Crearemos una base de datos seleccionando la opción "Nueva" que aparece del lado izquierdo de la página.

En el siguiente diálogo debemos definir el nombre de nuestra base de datos, la llamaremos "bd1":



Una vez que se crea la podemos ver que aparece en la columna izquierda y podemos pasar a crear tablas en la misma. Crearemos la tabla artículos que tendrá 3 campos:



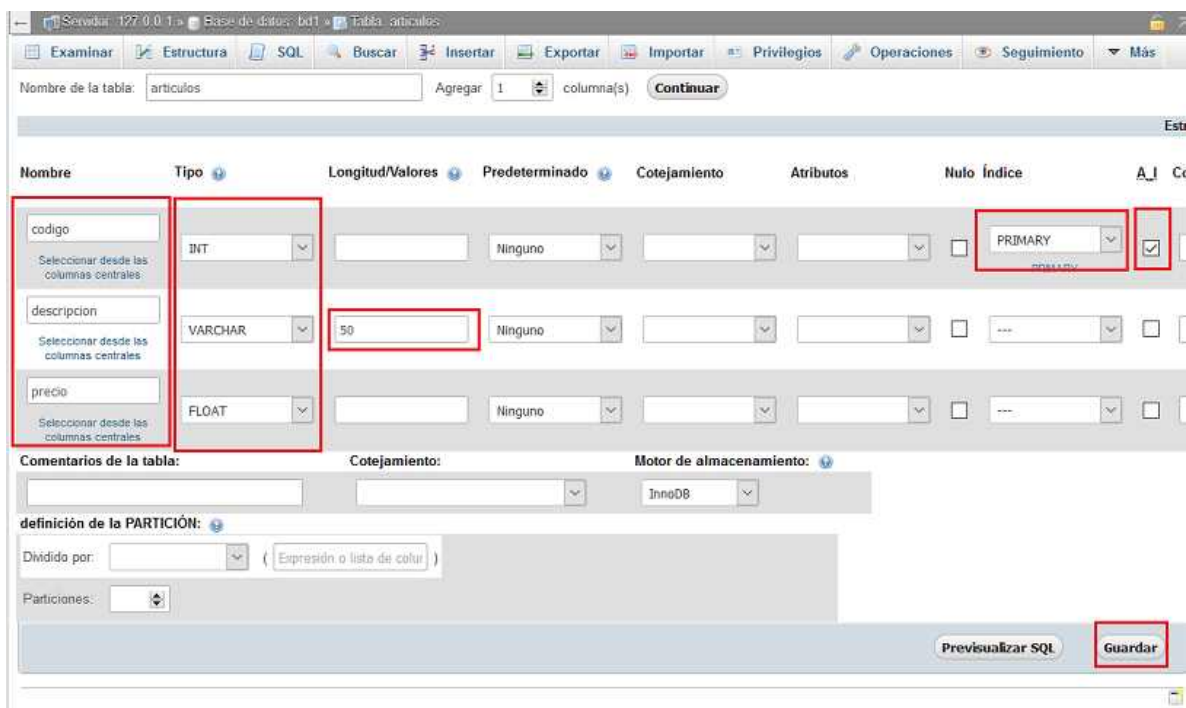
Almacenaremos en la tabla *articulos* el código, descripción y precio.

El campo 'código' será int 'clave primaria' y auto_increment.

El campo 'descripción' será varchar de 50.

El campo 'precio' será float.

Los datos a ingresar para cada campo para la creación de la tabla *articulos* son:



Ya tenemos creada la base de datos: "bd1" y en ésta la tabla "articulos":



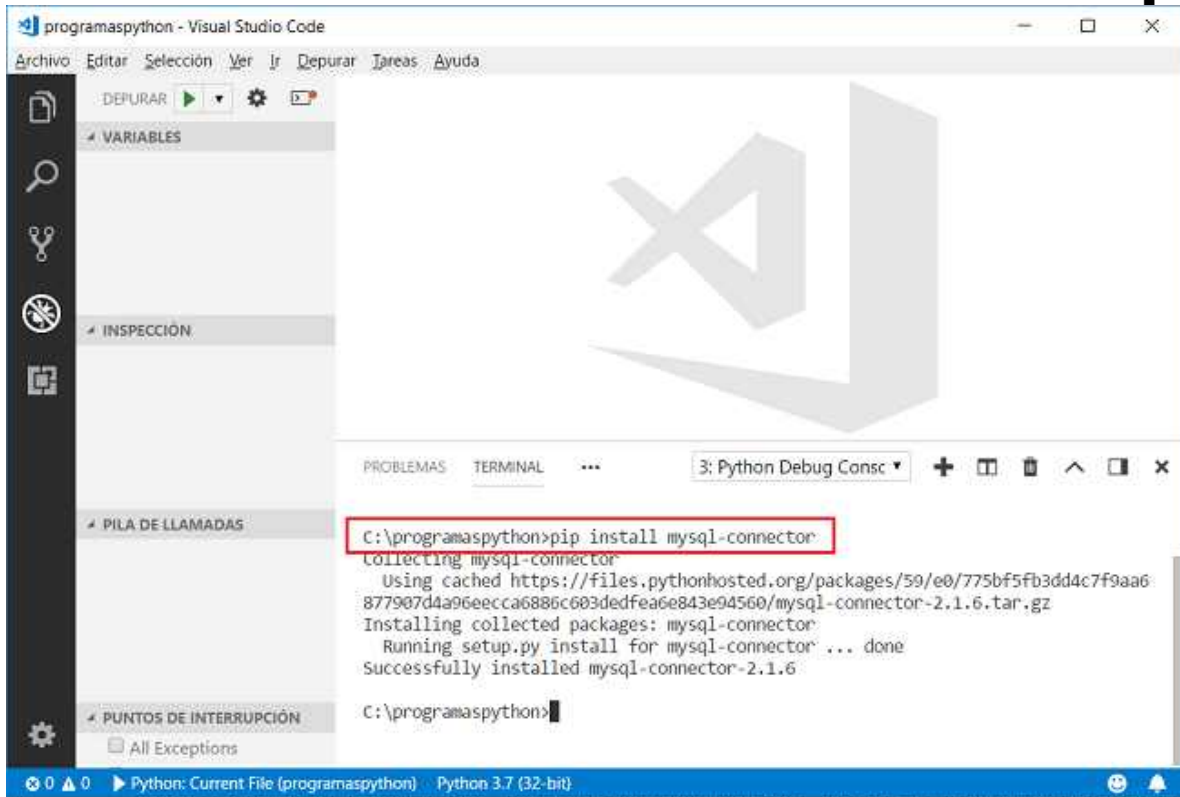
Paquete de Python necesario para conectarnos a MySQL

Utilizaremos el programa 'pip' que vimos anteriormente para instalar el paquete necesario para interconectar 'Python' y 'MySQL'.

Desde la línea de comandos ejecutamos el programa pip con el siguiente paquete a instalar:

```
pip install mysql-connector
```

Luego de ejecutar el programa pip podemos ver que nos informa de la instalación del paquete 'mysql-connector':



Conexión con el servidor de MySQL

Controlar que el "XAMPP Control Panel" se encuentre en ejecución el servidor de MySQL:

El primer programa que implementaremos nos conectaremos con el servidor de MySQL y mostraremos todas las bases de datos existentes (una de esas debería ser bd1)

```
import mysql.connector

conexion1=mysql.connector.connect(host="localhost", user="root", passwd="")

cursor1=conexion1.cursor()

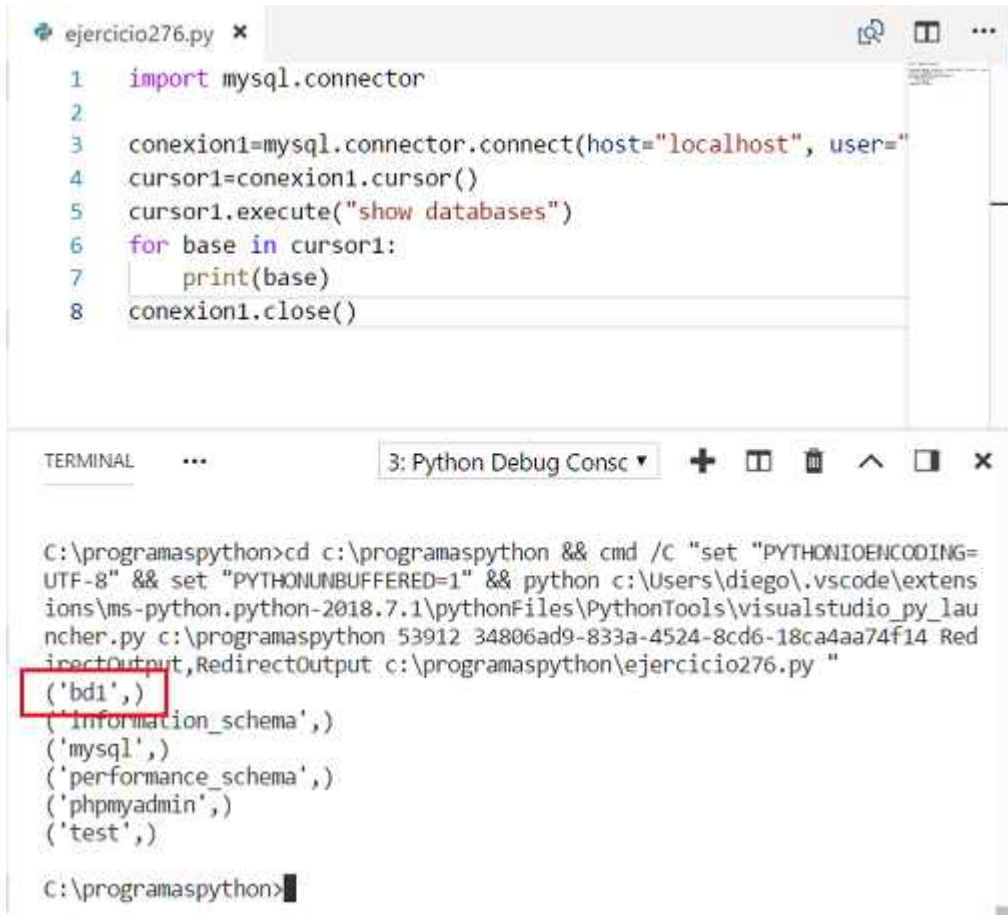
cursor1.execute("show databases")

for base in cursor1:

    print(base)

conexion1.close()
```

El resultado de ejecutar este programa es:



The screenshot shows a VS Code editor window with a file named 'ejercicio276.py'. The Python code in the editor is as follows:

```
1 import mysql.connector
2
3 conexion1=mysql.connector.connect(host="localhost", user="
4 cursor1=conexion1.cursor()
5 cursor1.execute("show databases")
6 for base in cursor1:
7     print(base)
8 conexion1.close()
```

Below the editor is a terminal window titled '3: Python Debug Consc'. It shows the command prompt execution of the script, with the output of the 'show databases' query listed below:

```
C:\programaspython>cd c:\programaspython && cmd /C "set "PYTHONIOENCODING=
UTF-8" && set "PYTHONUNBUFFERED=1" && python c:\Users\diego\.vscode\extens
ions\ms-python.python-2018.7.1\pythonFiles\PythonTools\visualstudio_py_lau
ncher.py c:\programaspython 53912 34806ad9-833a-4524-8cd6-18ca4aa74f14 Red
irectOutput,RedirectOutput c:\programaspython\ejercicio276.py "
('bd1',)
('information_schema',)
('mysql',)
('performance_schema',)
('phpmyadmin',)
('test',)

C:\programaspython>
```

Lo primero que hacemos es importar el módulo que nos permite conectarnos con MySQL:

```
import mysql.connector
```

Del módulo importado llamamos a la función `connect` pasando la ubicación de nuestro servidor que es 'localhost', el usuario que por defecto al instalar MySQL se creó el usuario 'root' y la clave de ese usuario que tiene por defecto un string vacío:

```
conexion1=mysql.connector.connect(host="localhost", user="root", passwd="")
```

Si por ejemplo el servidor de MySQL no se encuentra en ejecución el programa se detendrá en esta línea informando un error.

Luego a partir del objeto 'conexion1' que es de la clase 'MySQLConnection' llamamos al método 'cursor':

```
cursor1=conexion1.cursor()
```

A partir del objeto 'cursor1' llamamos al método `execute` y le pasamos como parámetro un comando SQL, en este caso 'show databases':

```
cursor1.execute("show databases")
```

Mediante un `for` podemos ver todas las bases de datos existentes en nuestro servidor de MySQL:

```
for base in cursor1:
```

```
    print(base)
```

Finalmente cerramos la conexión con el servidor de MySQL:

```
conexion1.close()
```

Listado de todas las tablas de una base de datos de MySQL

Ahora implementaremos un programa que recupere todas las tablas contenidos en una base de datos. Trabajaremos con la base de datos que creamos desde el PHPMyAdmin llamada 'bd1'.

```
import mysql.connector

conexion1=mysql.connector.connect(host="localhost",
                                user="root",
                                passwd="",
                                database="bd1")

cursor1=conexion1.cursor()
cursor1.execute("show tables")
for tabla in cursor1:
    print(tabla)
conexion1.close()
```

El resultado de ejecutar este programa es:



The screenshot shows a VS Code editor window with a file named 'ejercicio277.py'. The code in the editor is as follows:

```
1 import mysql.connector
2
3 conexion1=mysql.connector.connect(host="localhost",
4                                   user="root",
5                                   passwd="",
6                                   database="bd1")
7 cursor1=conexion1.cursor()
8 cursor1.execute("show tables")
9 for tabla in cursor1:
10     print(tabla)
11 conexion1.close()
12
```

Below the editor is a terminal window titled '4: Python Debug Console'. It shows the command prompt execution of the script, with the output ('articulos',) highlighted by a red box.

```
C:\programaspython>cd c:\programaspython && cmd /c "set "PYTHONIOENCODING=UTF-8" && set "PYTHONUNBUFFERED=1" && python c:\Users\diego\vscode\extensions\ms-python.python-2018.7.1\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\programaspython 54146 34806ad9-833a-4524-8cd6-18ca4aa74f14 RedirectOutput,RedirectOutput c:\programaspython\ejercicio277.py "
```

```
('articulos',)
```

```
C:\programaspython>
```

Cuando nos conectamos con el servidor de MySQL indicamos en 'database' la base de datos activa para cuando lancemos comandos SQL:

```
conexion1=mysql.connector.connect(host="localhost",  
                                  user="root",  
                                  passwd="",  
                                  database="bd1")
```

Luego de crear el cursor solicitamos que se ejecute el comando SQL 'show tables' que nos retorna todas las tablas existentes de la base de datos activa indicada cuando llamamos a 'connect':

```
cursor1=conexion1.cursor()  
cursor1.execute("show tables")
```

Imprimimos con un for todas las tablas y cerramos la conexión:

```
for tabla in cursor1:  
    print(tabla)  
conexion1.close()
```

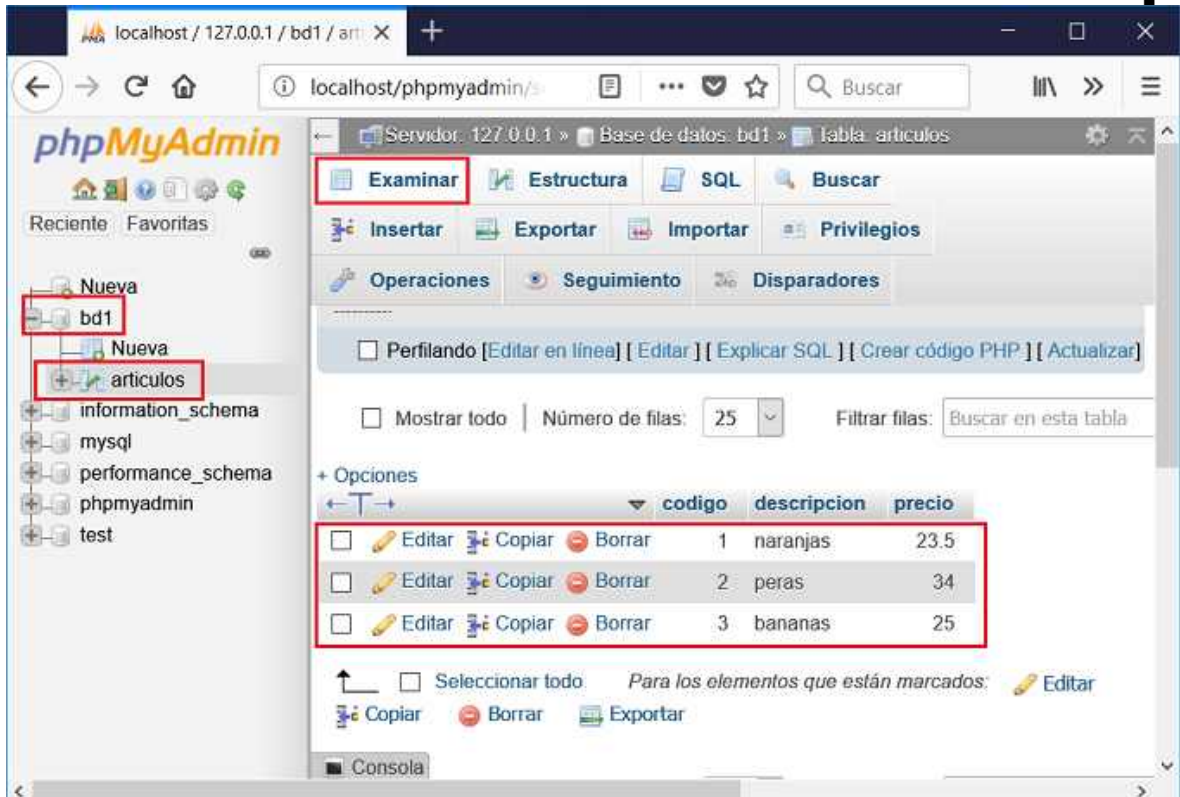
Insertar filas en una tabla

Ahora implementaremos un programa que inserte un par de filas en la tabla 'articulos'.

```
import mysql.connector

conexion1=mysql.connector.connect(host="localhost",
                                  user="root",
                                  passwd="",
                                  database="bd1")
cursor1=conexion1.cursor()
sql="insert into articulos(descripcion, precio) values (%s,%s)"
datos=("naranjas", 23.50)
cursor1.execute(sql, datos)
datos=("peras", 34)
cursor1.execute(sql, datos)
datos=("bananas", 25)
cursor1.execute(sql, datos)
conexion1.commit()
conexion1.close()
```

Por el momento si queremos controlar que se han cargado las tres filas en la tabla 'articulos' podemos abrir el 'PHPMyAdmin' y ver el contenido de la tabla:



Definimos un string con el comando SQL insert disponiendo la máscara %s donde queremos que se sustituya por un valor que le pasaremos al método execute:

```
sql="insert into articulos(descripcion, precio) values (%s,%s)"
```

La variable datos es una tupla que contiene los datos que se utilizarán en la sustitución %s:

```
datos=("naranjas", 23.50)
```

Finalmente llamamos al método 'execute' y le pasamos las dos variables que acabamos de crear:

```
cursor1.execute(sql, datos)
```

Es fundamental llamar al final al método 'commit' para que queden firmes los comandos SQL 'insert':

```
conexion1.commit()
```

Recuperar todas las filas de una tabla

Implementaremos un programa que solicite ejecutar un 'select' en la tabla 'articulos' y nos retorne todas sus filas.

```
import mysql.connector
```

```
conexion1=mysql.connector.connect(host="localhost",
```



```
        user="root",
        passwd="",
        database="bd1")

cursor1=conexion1.cursor()


cursor1.execute("select codigo, descripcion, precio from articulos")

for fila in cursor1:

    print(fila)

conexion1.close()
```

Cuando ejecutamos el programa podemos ver que se recuperan todas las filas de la tabla 'articulos':



The screenshot shows a VS Code editor window with a file named 'ejercicio279.py'. The code in the editor is as follows:

```
1 import mysql.connector
2
3 conexion1=mysql.connector.connect(host="localhost",
4                                   user="root",
5                                   passwd="",
6                                   database="bd1")
7 cursor1=conexion1.cursor()
8 cursor1.execute("select codigo, descripcion, precio from articulos")
9 for fila in cursor1:
10     print(fila)
11 conexion1.close()
12
```

Below the editor, the terminal window is open, showing the command used to run the script and its output. The output is a list of three rows from the 'articulos' table, which is highlighted with a red box:

```
C:\programaspython>cd c:\programaspython && cmd /C "set "PYTHONIOENCODING=UTF-8" && set
"PYTHONUNBUFFERED=1" && python c:\Users\diego\.vscode\extensions\ms-python.python-2018.7
.1\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\programaspython 56140 34806ad9
-833a-4524-8cd6-18ca4aa74f14 RedirectOutput,RedirectOutput c:\programaspython\ejercicio2
79.py "
```

```
(1, 'naranjas', 23.5)
(2, 'peras', 34.0)
(3, 'bananas', 25.0)
```

```
C:\programaspython>
```

Luego de conectarnos y crear un cursor procedemos a ejecutar el comando 'select', recorremos con un for el 'cursor1':

```
cursor1=conexion1.cursor()

cursor1.execute("select codigo, descripcion, precio from articulos")
```

```
for fila in cursor1:  
    print(fila)
```

Borrado y modificación de filas

Las otras dos actividades fundamentales que podemos hacer con una tabla es borrar filas y modificar datos.

Desarrollaremos un pequeño programa que borre el artículo cuyo código sea el 1 y modifique el precio del artículo cuyo código sea 3.

```
import mysql.connector  
  
conexion1=mysql.connector.connect(host="localhost",  
                                  user="root",  
                                  passwd="",  
                                  database="bd1")  
  
cursor1=conexion1.cursor()  
cursor1.execute("delete from articulos where codigo=1")  
cursor1.execute("update articulos set precio=50 where codigo=3")  
conexion1.commit()  
cursor1.execute("select codigo, descripcion, precio from articulos")  
for fila in cursor1:  
    print(fila)  
conexion1.close()
```

Cuando ejecutamos el programa podemos ver que se eliminó el artículo cuyo código es 1 y se modificó el precio del artículo con código 3:



The screenshot shows a VS Code editor window with a file named 'ejercicio280.py'. The code is a Python script that connects to a MySQL database, performs several SQL operations, and prints the results. The terminal output shows the execution of the script, with the last two lines of output, '(2, 'peras', 34.0)' and '(3, 'bananas', 50.0)', highlighted by a red box.

```
1 import mysql.connector
2
3 conexion1=mysql.connector.connect(host="localhost",
4                                   user="root",
5                                   passwd="",
6                                   database="bd1")
7 cursor1=conexion1.cursor()
8 cursor1.execute("delete from articulos where codigo=1")
9 cursor1.execute("update articulos set precio=50 where codigo=3")
10 conexion1.commit()
11 cursor1.execute("select codigo, descripcion, precio from articulos")
12 for fila in cursor1:
13     print(fila)
14 conexion1.close()
15
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL 2: Python Debug Consc

```
C:\programaspython>cd c:\programaspython && cmd /C "set "PYTHONIOENCODING=UTF-8" && set "PYTHONUNBUFFERED=1" && python c:\Users\diego\.vscode\extensions\ms-python.python-2018.7.1\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\programaspython 56259 34806ad9-833a-4524-8cd6-18ca4aa74f14 RedirectOutput RedirectOutput c:\programaspython\ejercicio280.py "
```

(2, 'peras', 34.0)
(3, 'bananas', 50.0)

C:\programaspython>

Luego de crear el cursor podemos llamar al método 'execute' varias veces y pasar distintos comando SQL:

```
cursor1=conexion1.cursor()
cursor1.execute("delete from articulos where codigo=1")
cursor1.execute("update articulos set precio=50 where codigo=3")
```

Siempre que pasemos un comando SQL: insert, delete o update debemos llamar al método commit para que quede firme los cambios en la base de datos:

```
conexion1.commit()
```

Ejecutamos finalmente un 'select' para comprobar los cambios efectuados en la tabla 'articulos':

```
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
```