****SQL**** stands for [Structured Query Language](https://en.wikipedia.org/wiki/SQL) and is a widely used programming language for managing relational databases. You may have heard of the different flavors of SQL-based DBMSs. The most popular ones include [MySQL](https://www.mysql.com/), [PostgreSQL](https://www.postgresql.org/), [SQLite](https://www.sqlite.org/index.html), and [SQL Server](https://www.microsoft.com/en-us/sql-server/sql-server-2019). All of these databases are compliant with the [SQL standards](https://docs.oracle.com/cd/B28359_01/server.111/b28286/intro002.htm) but with varying degrees of compliance.

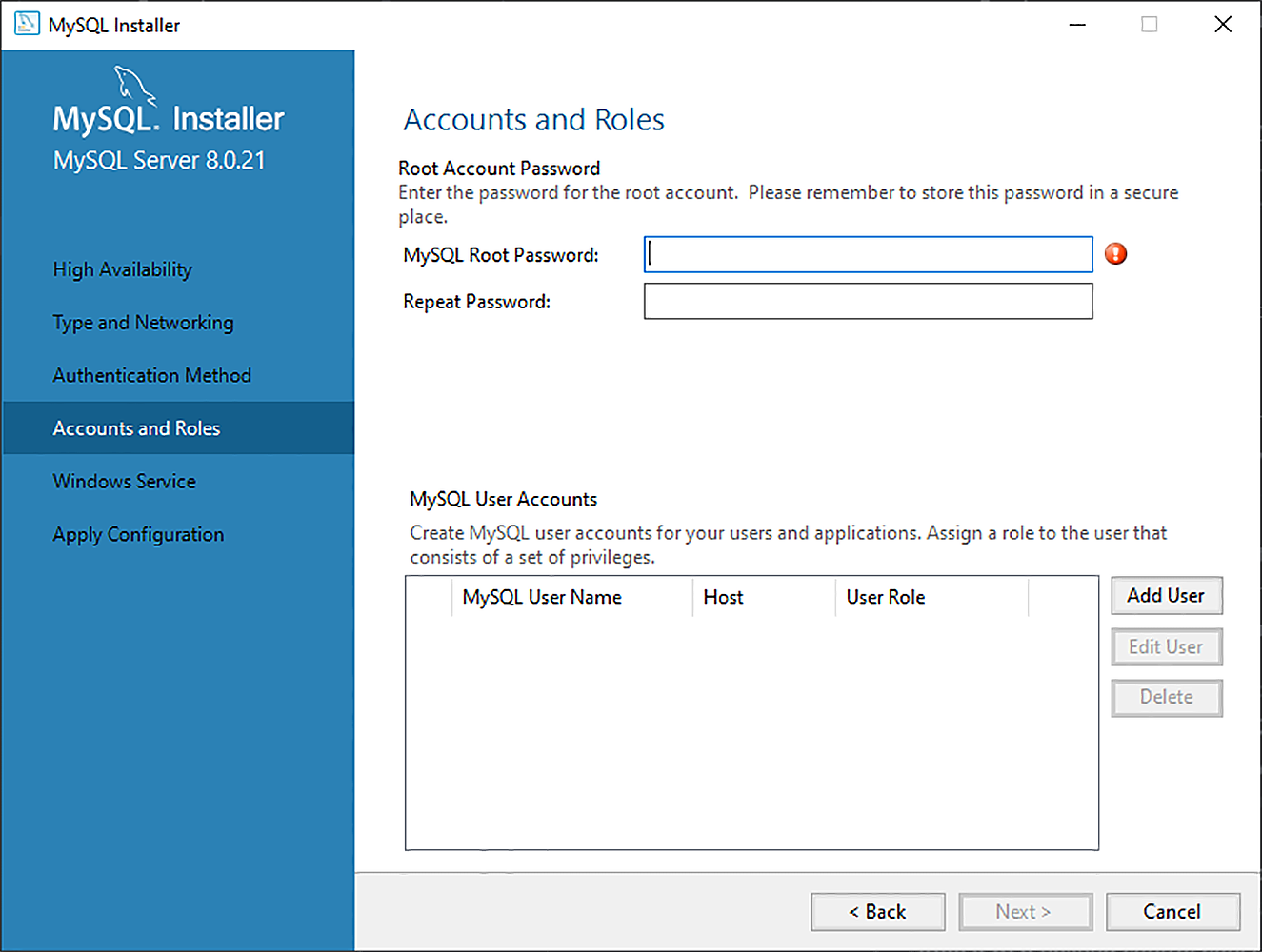
## **Installing MySQL Server and MySQL Connector/Python**

MySQL server will provide all the services required for handling your database. Once the server is up and running, you can connect your Python application with it using MySQL Connector/Python.

### Installing MySQL Server

The [official documentation](https://dev.mysql.com/doc/refman/5.7/en/installing.html) details the recommended way to download and install MySQL server. You’ll find instructions for all popular operating systems, including [Windows](https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html), [macOS](https://dev.mysql.com/doc/refman/8.0/en/osx-installation.html), [Solaris](https://dev.mysql.com/doc/refman/8.0/en/solaris-installation.html), [Linux](https://dev.mysql.com/doc/refman/8.0/en/linux-installation.html), and many more.

For Windows, the best way is to download [MySQL Installer](https://dev.mysql.com/downloads/installer/) and let it take care of the entire process. The installation manager also helps you configure the security settings of the MySQL server. On the Accounts and Roles page, you need to enter a password for the ****root**** (admin) account and also optionally add other users with varying privileges:

[](https://files.realpython.com/media/root_user_pass.cf4a9c6ea5f5.png)MySQL Installer Account Setup

While you must specify credentials for the root account during setup, you can modify these settings later on.

****Note:**** Remember the hostname, username, and password as these will be required to establish a connection with the MySQL server later on.

### Installing MySQL Connector/Python

A [database driver](https://docs.microsoft.com/en-us/sql/odbc/reference/dbms-based-drivers) is a piece of software that allows an application to connect and interact with a database system. Programming languages like Python need a special driver before they can speak to a database from a specific vendor.

These drivers are typically obtained as third-party modules.

Similarly, in Python you need to install a Python MySQL connector to interact with a MySQL database. Many packages follow the DB-API standards, but the most popular among them is [MySQL Connector/Python](https://dev.mysql.com/doc/connector-python/en/). You can get it with [pip](https://realpython.com/what-is-pip/):

$ pip install mysql-connector-python

pip installs the connector as a third-party module in the currently active virtual environment. It’s recommended that you set up an [isolated virtual environment](https://realpython.com/python-virtual-environments-a-primer/) for the project along with all the dependencies.

To test if the installation was successful, type the following command on your Python [terminal](https://realpython.com/terminal-commands/):

>>>

>>> import mysql.connector

## **Establishing a Connection With MySQL Server**

MySQL is a ****server-based**** database management system. One server might contain multiple databases. To interact with a database, you must first establish a connection with the server. The general workflow of a Python program that interacts with a MySQL-based database is as follows:

1. Connect to the MySQL server.
2. Create a new database.
3. Connect to the newly created or an existing database.
4. Execute a SQL query and fetch results.
5. Inform the database if any changes are made to a table.
6. Close the connection to the MySQL server.

This is a generic workflow that might vary depending on the individual application. But whatever the application might be, the first step is to connect your database with your application.

### Establishing a Connection

The first step in interacting with a MySQL server is to establish a connection. To do this, you need [connect()](https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysql-connector-connect.html) from the mysql.connector module. This function takes in parameters like host, user, and password and returns a [MySQLConnection](https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html) object.

import mysql.connector  
  
con = mysql.connector.connect(  
  host="localhost",  
  user="yourusername",  
  password="yourpassword"  
)

The code above uses the entered login credentials to establish a connection with your MySQL server. In return, you get a MySQLConnection object, which is stored in the connection variable. From now on, you’ll use this [variable](https://realpython.com/python-variables/) to access your MySQL server.

You’ve now established a connection between your program and your MySQL server, but you still need to either create a new database or connect to an existing database inside the server.

### Creating a New Database

In the last section, you established a connection with your MySQL server. To create a new database, you need to execute a SQL statement:

CREATE DATABASE mydatabase;

To execute a SQL query in Python, you’ll need to use a [cursor](https://en.wikipedia.org/wiki/Cursor_(databases)), which abstracts away the access to database records. MySQL Connector/Python provides you with the [MySQLCursor](https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlcursor.html) class, which instantiates objects that can execute MySQL queries in Python. An instance of the MySQLCursor class is also called a cursor.

cursor objects make use of a MySQLConnection object to interact with your MySQL server. To create a cursor, use the .cursor() method of your connection variable:

cursor = connection.cursor()

The above code gives you an instance of the MySQLCursor class.

A query that needs to be executed is sent to [cursor.execute()](https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlcursor-execute.html) in [string](https://realpython.com/python-strings/) format. In this particular occasion, you’ll send the CREATE DATABASE query to cursor.execute():

import mysql.connector  
  
con= mysql.connector.connect(  
  host="localhost",  
  user="yourusername",  
  password="yourpassword"  
)  
  
mycursor = mydb.cursor()  
  
mycursor.execute("CREATE DATABASE mydatabase")

### Connecting to an Existing Database

In the last section, you created a new database called mydatabse. However, you still haven’t connected to it. In many situations, you’ll already have a MySQL database that you want to connect with your Python application.

You can do this using the same connect() function that you used earlier by sending an additional parameter called database:

import mysql.connector  
  
con = mysql.connector.connect(  
  host="localhost",  
  user="yourusername",  
  password="yourpassword",  
****database=**"mydatabase"**  
)

The above code is very similar to the connection script that you used earlier. The only change here is an additional database parameter, where the name of your database is passed to connect(). Once you execute this script, you’ll be connected to the  mydatabase.