# Setting Up Python for MySQL

*AiCE Software Engineering Foundations — MySQL Programming*

You already know Python. In this lesson, you'll connect it to the exact same aice_books database you built directly in MySQL — the same table, the same books, but now controlled entirely from Python code instead of typed by hand at the mysql> prompt.

## 1. Starting a New Project

Every project you build from here on gets its own GitHub repository. Create a new one and name it aice-mysql-python, generate a personal access token if you haven't already, then clone it and move into the folder.

From now on, every Python project you create gets its own dedicated folder inside your repository. Create one for this project:

```
mkdir aice_books_app
cd aice_books_app
```

## 2. Why Every Python Project Gets Its Own Virtual Environment

A virtual environment is a private, self-contained copy of Python that lives inside a single project folder. Instead of installing packages globally — where every project on your machine shares the same versions, and one project's needs can quietly break another's — a virtual environment keeps each project's dependencies isolated from everything else on your computer.

## 3. Creating and Activating a Virtual Environment

Now that you're inside your project directory, create one:

```
python3 -m venv venv
```

This creates a folder named venv inside aice_books_app, containing a fresh, isolated copy of Python. Creating it is not enough on its own — it also has to be activated. On Ubuntu:

```
source venv/bin/activate
```

On PowerShell:

```
.\venv\Scripts\Activate.ps1
```

Your prompt now starts with (venv) — that tells you the virtual environment is active. From this point on, any package you install goes into this isolated copy of Python, not your system-wide one. When you're done working, leave it with:

```
deactivate
```

**Reactivate It Every Time**

*(venv) does not stay active between terminal sessions. Every time you open a new terminal to work on this project, activate it again before running any Python or pip command.*

## 4. Keeping venv Out of Git

One important rule: never commit the venv folder to source control. It can be regenerated from requirements.txt on any machine, and committing it bloats your repository with thousands of unnecessary files. Create a .gitignore file with vim:

```
vim .gitignore
```

Write the name of anything you don't want pushed to GitHub. Since venv is a folder, add a trailing slash:

```
venv/
```

Save and quit. Git now ignores the entire venv folder on every commit.

## 5. Installing the MySQL Connector

With (venv) active, install the Python module that lets your code talk to MySQL — the MySQL connector:

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

Press Enter and wait for it to finish installing.

## 6. Recording Dependencies With requirements.txt

It's good practice to record every module you've installed, so anyone else — or you, on a new machine — can recreate the exact same setup:

```
pip freeze > requirements.txt
```

This writes every installed package and its exact version into requirements.txt. On a new machine, someone could recreate your entire environment with a single pip install -r requirements.txt.

## 7. Writing Your First Connection Script

Create your Python file, named app.py:

```
vim app.py
```

Type the following:

```
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="your_password",
    database="aice_books"
)
```

```
cursor = conn.cursor()

print("Connected to aice_books successfully!")
```

The value of password here is the same MySQL root password you set when securing your installation — replace "your_password" with your actual password.

## 8. Running It

```
python3 app.py
```

You should see "Connected to aice_books successfully!" printed. Python is now talking directly to MySQL.

> **A Real Password in Code — For Now**
> *Keeping a real password typed directly into app.py like this is fine while you're just getting the connection working, but it is not something you'd ever want to push to GitHub for the world to see. This unit leaves it in the code; a later unit fixes it properly.*

## Key Takeaways

You gave this project its own folder, its own isolated virtual environment, and a .gitignore that keeps venv out of version control. You installed mysql-connector-python, recorded it in requirements.txt, and wrote a connection script that logs in to aice_books directly from Python — the same database you built and populated by hand in the previous unit. With the connection working, the next lesson puts it to use: performing every CRUD operation — Create, Read, Update, Delete — from Python code instead of the mysql> prompt.