Database Connection

Why We Use Database Connection in Python

Python is a powerful programming language, but it doesn't **store data permanently** by itself. That's where databases come in!

When you **connect Python to a database**, you can:

✅ **Store data permanently**
Example: Save user registration details, product listings, student records, etc.

✅ **Retrieve data later**
Example: Load data from a table to display in a web app or desktop app.

✅ **Manage large volumes of structured data**
Databases like MySQL, PostgreSQL, or SQLite handle thousands (or millions) of records efficiently.

🎯 Real-Life Examples

Use Case	How Python + DB helps
Login/Signup system	Store usernames and passwords securely
E-commerce app	Manage product listings, orders, inventory
School management system	Track students, marks, attendance
Employee records	HR systems use databases to store details
Data analysis projects	Fetch real-time data for analysis or reports

---

Common Databases You Can Use with Python

DBMS	Python Library
MySQL	mysql-connector-python
PostgreSQL	psycopg2
SQLite	sqlite3 (built-in)
MongoDB	pymongo
Oracle	cx_Oracle

🔗 What is a Database Connection?

A **database connection** is a **bridge between your Python program and a database** (like MySQL, SQLite, PostgreSQL, etc.). It allows your Python code to **talk to the database** to:

- Save data
- Read data
- Update data
- Delete data

🧱 Components of a Database Connection

When you connect to a database, you typically need:

Component	Description
Host	The server where the database is running (localhost for local)
User	The username to access the database
Password	The password for that user
Database	The specific database you want to work with
Port	Communication port (default for MySQL: 3306)

Steps to Connect Python to a Database

Here's a typical process using **MySQL**:

1. Install the connector:

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

2. Import and Connect

```
import mysql.connector

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

3. Create a Cursor Object

```
cursor = connection.cursor()
```

4. Execute SQL Queries

```
cursor.execute("SELECT * FROM students")
results = cursor.fetchall()
for row in results:
    print(row)
```

5. Commit Changes (if needed)

```
connection.commit()
```

6. Close Connection

```
cursor.close()
connection.close()
```

✅ Features & Advantages

Feature	Why it's Important
🔄 CRUD Support	Add, edit, delete, and read data easily

Feature	Why it's Important
📁 Data Persistence	Data doesn't disappear when program ends
⚡ Efficient	Databases handle large data faster than files
🔐 Secure Access	Controlled access using users and roles
📊 Integration	Works great with data tools and dashboards
🧠 SQL Power	Use powerful queries for filtering, joins, etc.

Database Connection

```
import mysql.connector

connection=mysql.connector.connect(
   host="localhost",
   user="root",
   password="root",
   database="py_connection"
)

if connection.is_connected():
   print("connected to mysql")

connection.close()
```

CURD Operations:

1. Creation of Table

```
import mysql.connector


connection=mysql.connector.connect(
   host="localhost",
   user="root",
   password="root",
   database="py_connection"
)

if connection.is_connected():
   print("connected to mysql")

connection.close()
```

2. Inserting Data

```
import mysql.connector
from mysql.connector import Error

try:
    connection=mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="py_connection"
    )

    if connection.is_connected():
        print("connected to mysql")


    cursor = connection.cursor()

    # cursor.execute("SHOW DATABASES")  # Show all databases
    # for db in cursor:
    #     print(db)


    cursor.execute("INSERT INTO students (name, age) VALUES (%s, %s)", ("John", 20))

    sql = "INSERT INTO students (name, age) VALUES (%s, %s)"
    val = ("Alice", 22)
    cursor.execute(sql,val)

    connection.commit()
    print("Data entered")


except Error as e:
    print("error at connection ",e)


finally:
    if connection:
        cursor.close()
        connection.close()
```

3. Modifying /Updating Data

```
cursor = connection.cursor()
#  Update a record
update_sql = "UPDATE students SET age = %s WHERE name = %s"
update_val = (23, "Alice")
cursor.execute(update_sql, update_val)

connection.commit()
print(" Record updated successfully")
```

4. Removing Single record

```
cursor = connection.cursor()

# DELETE a record where name is 'John'
delete_sql = "DELETE FROM students WHERE name = %s"
delete_val = ("John",)
cursor.execute(delete_sql, delete_val)

connection.commit()
print(" Record deleted successfully")
```

5. Removing all records

```
cursor = connection.cursor()

# DELETE all records
cursor.execute("DELETE FROM students")
connection.commit()

print(" Record deleted successfully")
```

Data Fetching/ Display Table Data

```
cursor = connection.cursor()

# cursor.execute("select * from students")
# print(cursor.fetchall())

cursor.execute("select * from students")
result=cursor.fetchall()

for row in result:
    print(row)
```

Login/Signup System using Python + MySQL

Project Overview

A **console-based application** with:

1. ✅ Signup – Save new users to the database.
2. 🔐 Login – Verify user credentials.
3. ❌ Exit – Exit the app.

Database Setup (MySQL): First, open MySQL and run

```
CREATE DATABASE user_auth;

USE user_auth;

CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(255) NOT NULL
);
```

Python Script (login_signup.py)

Here's a basic working code:

```
import mysql.connector
from mysql.connector import Error


try:
   # Connect to database
   mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="root",
      database="py_connection"
   )


   cursor = mydb.cursor()

   # Signup function
   def signup():
      username = input("Enter a new username: ")
      password = input("Enter a new password: ")

      try:
         cursor.execute("INSERT INTO users (username, password) VALUES (%s, %s)", (username, password))
         mydb.commit()
         print(" ✅ Signup successful!")
      except mysql.connector.IntegrityError:
         print(" ❗ Username already exists. Try a different one.")
```

```
# Login function
def login():
    username = input("Enter your username: ")
    password = input("Enter your password: ")

    cursor.execute("SELECT * FROM users WHERE username = %s AND password = %s", (username, password))
    result = cursor.fetchone()

    if result:
        print("🔓 Login successful! Welcome,", username)
    else:
        print("❌ Invalid credentials. Try again.")


# Menu
while True:
    print("\n===== Login/Signup System =====")
    print("1. Signup")
    print("2. Login")
    print("3. Exit")

    choice = input("Choose an option: ")

    if choice == "1":
        signup()
    elif choice == "2":
        login()
    elif choice == "3":
        print("👋 Goodbye!")
        break
    else:
        print("❗ Invalid choice. Try again.")


except Error as e:
    print("error at connection ",e)

finally:
    if mydb:
        cursor.close()
        mydb.close()
```

Output:

```
PS C:\Users\abhin\OneDrive\Desktop\10 k coders\7. Python Programing Language> & "
Programing Language/.venv/Scripts/python.exe" "c:/Users/abhin/OneDrive/Desktop/10
e Connection/login_signup_form.py"

===== Login/Signup System =====
1. Signup
2. Login
3. Exit
Choose an option: 1
Enter a new username: Abhi
Enter a new password: 1234
✅ Signup successful!
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
✅ Signup successful!

===== Login/Signup System =====
1. Signup
2. Login
3. Exit
Choose an option: 2
Enter your username: Abhi
Enter your password: 1234
🔓 Login successful! Welcome, Abhi

===== Login/Signup System =====
1. Signup
2. Login
3. Exit
Choose an option:
```