Python Programming

Unit 04 – Lecture 06: Transactions and Database Error Handling

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts Demo Interactive Summary

Agenda

Learning Outcomes

- Explain what a transaction is and why it matters

Learning Outcomes

- Explain what a transaction is and why it matters
- Use `commit` and `rollback` correctly

Learning Outcomes

- Explain what a transaction is and why it matters
- Use `commit` and `rollback` correctly
- Handle common database errors (constraints, invalid input)

Learning Outcomes

- Explain what a transaction is and why it matters
- Use `commit` and `rollback` correctly
- Handle common database errors (constraints, invalid input)
- Write safer DB code using try/except and context managers

What is a Transaction?

- A transaction is a group of operations treated as one unit

What is a Transaction?

- A transaction is a group of operations treated as one unit
- Either **all succeed** (commit) OR **none apply** (rollback)

What is a Transaction?

- A transaction is a group of operations treated as one unit
- Either **all succeed** (commit) OR **none apply** (rollback)
- Prevents partial updates (data corruption)

Commit vs Rollback

- `commit()` saves changes permanently

Commit vs Rollback

- `commit()` saves changes permanently
- `rollback()` cancels changes since last commit

Commit vs Rollback

- `commit()` saves changes permanently
- `rollback()` cancels changes since last commit
- Always rollback on failure in multi-step updates

Typical Safe Pattern

```
import sqlite3

con = sqlite3.connect("app.db")
try:
    con.execute("INSERT ...")
    con.execute("UPDATE ...")
    con.commit()
except sqlite3.Error as e:
    con.rollback()
    print("DB Error:", e)
finally:
    con.close()
```

Common DB Errors

- `IntegrityError`: constraint failed (duplicate unique key)

Common DB Errors

- `IntegrityError`: constraint failed (duplicate unique key)
- `OperationalError`: SQL error, missing table, locked DB

Common DB Errors

- `IntegrityError`: constraint failed (duplicate unique key)
- `OperationalError`: SQL error, missing table, locked DB
- Input issues: wrong types, empty fields

Demo: Rollback on Failure

- File: `demo/transactions_rollback_demo.py`

Demo: Rollback on Failure

- File: `demo/transactions_rollback_demo.py`
- Inserts two rows where the second violates a UNIQUE constraint

Demo: Rollback on Failure

- File: `demo/transactions_rollback_demo.py`
- Inserts two rows where the second violates a UNIQUE constraint
- Demonstrates that rollback prevents partial insertion

Checkpoint 1

Question: What problem can happen if you do not use transactions for multi-step updates?

Checkpoint 2

Question: Which exception might occur when you insert a duplicate unique value?

Think-Pair-Share

Discuss:

- In a banking app, why is rollback critical?

Key Takeaways

- Transactions ensure all-or-nothing updates

Key Takeaways

- Transactions ensure all-or-nothing updates
- Use commit on success and rollback on failure

Key Takeaways

- Transactions ensure all-or-nothing updates
- Use commit on success and rollback on failure
- Handle constraint errors and operational errors cleanly

Key Takeaways

- Transactions ensure all-or-nothing updates
- Use commit on success and rollback on failure
- Handle constraint errors and operational errors cleanly
- Always close DB connections (use try/finally or context managers)

Exit Question

In one sentence: what does `rollback()` do?