Introduction to Programming with Python

Dr. Anatol Wegner

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

anatol.wegner@uni-wuerzburg.de

Lecture 09
Databases with SQLite

January 17, 2025

Recap

- **Version control essentials:**
  - Track changes to your files.
  - Revert to previous versions.
  - Experiment without fear.
- **Git basics:**
  - Staging and committing changes.
  - Branching for parallel development.
  - Viewing history and differences.
- **Collaboration with Git:**
  - Remote repositories (GitHub).
  - Cloning, pushing, and pulling.
  - Branching and merging.
  - Pull requests for code review.

Recap

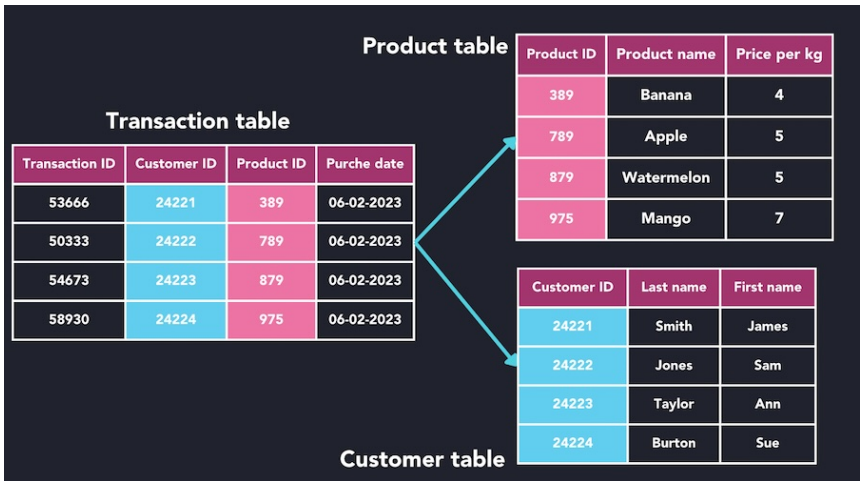
- **Version control essentials:**
  - Track changes to your files.
  - Revert to previous versions.
  - Experiment without fear.
- **Git basics:**
  - Staging and committing changes.
  - Branching for parallel development.
  - Viewing history and differences.
- **Collaboration with Git:**
  - Remote repositories (GitHub).
  - Cloning, pushing, and pulling.
  - Branching and merging.
  - Pull requests for code review.

Today: Databases with SQLite

- Basic concepts and SQL commands.
- Interacting with databases with SQlite3.

What is a Database?

- **Organized collection of data**
  - Digital filing cabinet
  - Library catalogue
  - Phone book



a database

Why Use Databases?

- **Efficient data storage and retrieval:**
  - Easily store and find the information you need.
- **Data integrity and consistency:**
  - Ensure data accuracy and prevent errors.
- **Reduced data redundancy:**
  - Avoid storing the same information multiple times.
- **Powerful querying and analysis:**
  - Extract meaningful insights from your data.

Focus on SQLite

- **Lightweight and file-based:**
  - No need for a separate server.
  - Great for learning and small projects.
- **Widely used:**
  - Found in web browsers, mobile apps, and more.

Python and SQLite

- **Built-in support:** Python has a built-in module (‘sqlite3‘) for working with SQLite databases.
- **Seamless integration:** Use Python code to connect to the database, execute queries, and manipulate data.
- **Powerful combination:** Leverage Python’s versatility and SQLite’s simplicity for your data management needs.

Relational Databases: Key Concepts

- **Tables:**
  - Organized structures that hold data in rows and columns.
  - Like spreadsheets or tables in a word processor.
- **Columns:**
  - Define the type of data stored (e.g., text, number, date).
  - Like the headers in a spreadsheet.
- **Rows:**
  - Represent individual records or entries in the table.

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	For Those A	1	1	1	Angus Young, M	343719	11170334	0.99
2	Balls to the	2	2	1	(Null)	342562	5510424	0.99
3	Fast As a Sh	3	2	1	F. Baltes, S. Kau	230619	3990994	0.99
4	Restless an	3	2	1	F. Baltes, R.A. Sı	252051	4331779	0.99
5	Princess of	3	2	1	Deaffy & R.A. S	375418	6290521	0.99
6	Put The Fir	1	1	1	Angus Young, M	205662	6713451	0.99
7	Let's Get It	1	1	1	Angus Young, M	233926	7636561	0.99
8	Inject The V	1	1	1	Angus Young, M	210834	6852860	0.99
9	Snowballed	1	1	1	Angus Young, M	203102	6599424	0.99
10	Evil Walks	1	1	1	Angus Young, M	263497	8611245	0.99
11	C.O.D.	1	1	1	Angus Young, M	199836	6566314	0.99
12	Breaking Th	1	1	1	Angus Young, M	263288	8596840	0.99
13	Night Of Th	1	1	1	Angus Young, M	205688	6706347	0.99
14	Spellbound	1	1	1	Angus Young, M	270863	8817038	0.99

Relational Databases: More Key Concepts

- **Primary Key:**
  - Uniquely identifies each row in a table.
  - Ensures that each record can be distinguished.
  - Often an auto-incrementing integer.
- **Foreign Key:**
  - A column that links related data in different tables.
  - Creates relationships between tables.
  - Refers to the primary key of another table.

id (PK)	name	nationality	birth_year
1	Jane Austen	English	1775
2	Charles Dickens	English	1812

Table 1: Authors

id (PK)	title	author_id **(FK)**
1	Pride and Prejudice	1
2	Oliver Twist	2

Table 2: Books

SQL Basics: Querying Data with SELECT

- **SQL (Structured Query Language):**
  - A language for interacting with relational databases.
- **SELECT statement:**
  - Used to retrieve data from a table.
  - Basic structure:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- **Explanation:**
  - `SELECT`: Specifies the columns you want to retrieve. Use '*' to select all columns.
  - `FROM`: Specifies the table you want to retrieve data from.
  - `WHERE`: (Optional) Specifies a condition to filter the results.
- **Example:**

```
SELECT title, author FROM books WHERE year > 1900;
```

  - This query retrieves the title and author of books published after 1900 from the "books" table.

SQL Basics: Modifying Data

- **INSERT statement:**
  - Used to add new data to a table.
- **Example:**

```
INSERT INTO books (title, author, year)
VALUES ('1984', 'George Orwell', 1949);
```

- **UPDATE statement:**
  - Used to modify existing data in a table.
- **Example:**

```
UPDATE books
SET year = 1948
WHERE title = '1984';
```

- **DELETE statement:**
  - Used to remove data from a table.
- **Example:**

```
DELETE FROM books WHERE id = 10;
```

The `sqlite3` Module

- **Built-in module:** Python comes with a built-in module called `sqlite3` for working with SQLite databases.
- **No installation needed:** You don't need to install any external libraries. Just **import sqlite3**.
- **Simple to use:** The `sqlite3` module provides a straightforward way to interact with SQLite databases using Python code.

Connecting to a Database

- **Establish a connection:** Use the `connect()` function to create a connection to the database.
- **Database file:** If the database file doesn't exist, it will be created.
- **Example:**

```
import sqlite3

conn = sqlite3.connect('mydatabase.db')
```

  - This creates a connection to the database file "mydatabase.db".
  - To execute SQL queries and fetch results we need a **cursor** object:

```
cursor = conn.cursor()
```

Using the Cursor

- **The Cursor Object:** The `cursor` object is essential for executing SQL queries and fetching results. Think of it as a pointer that allows you to traverse and manipulate data within the database. It is created using `conn.cursor()`.
- It is crucial to close the connection using `conn.close()` when you are finished interacting with the database to release resources.
- The cursor object has methods to execute SQL queries and retrieve data.
- Some important methods include:
  - `cursor.execute(sql_query)`: Executes an SQL query.
  - `cursor.fetchone()`: Fetches the next row of a query result.
  - `cursor.fetchall()`: Fetches all remaining rows of a query result.
  - `conn.commit()`: Saves changes to the database.

Creating a Table

- **Execute SQL:** Use the `execute()` method of the cursor to execute SQL statements.
- **Commit changes:** Use `conn.commit()` to save the changes to the database.
- **Example:**

```
import sqlite3

conn = sqlite3.connect('mydatabase.db')
cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS books
             (id INTEGER PRIMARY KEY, title TEXT,
             author TEXT, year INTEGER)''')
conn.commit()
```

Committing Changes to the Database

- **Database transactions:** Changes made with `INSERT`, `UPDATE`, or `DELETE` are not immediately saved to the database file.
- **Committing changes:** Use `conn.commit()` to save the changes permanently.

Example:

```
cursor.execute("INSERT INTO books (title, author, year)
VALUES (?, ?, ?)",
('The Restaurant at the End of the Universe',
'Douglas Adams', 1980))
conn.commit()
```

- Without `conn.commit()`, the new book would not be saved in the database.

Fetching Data

- **Execute a SELECT query:** Use `cursor.execute()` with a `SELECT` statement.
- **Fetch methods:**
  - `fetchone()`: Retrieves the next row of the result as a tuple.
  - `fetchall()`: Retrieves all rows of the result as a list of tuples.
  - `fetchmany(size)`: Retrieves the specified number of rows.

Example:

```
cursor.execute("SELECT * FROM books WHERE author=?",
('Douglas Adams',))

# Fetch one row
first_row = cursor.fetchone()
print(first_row)

# Fetch all rows
all_rows = cursor.fetchall()
```

Fetching Data

- **Execute a SELECT query:** Use `cursor.execute()` with a SELECT statement.
- **Fetch methods:**
  - `fetchone()`: Retrieves the next row of the result as a tuple.
  - `fetchall()`: Retrieves all rows of the result as a list of tuples.
  - `fetchmany(size)`: Retrieves the specified number of rows.

Practice Session

- Database basics with sqlite3

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Lecture_09.ipynb`

Example: Authors and Books Database

- **Two tables:**
  - `authors`: Stores information about authors (id, name, nationality, birth_year).
  - `books`: Stores information about books (id, title, author_id).
- **Foreign key:** The `author_id` column in the `books` table is a foreign key that references the `id` column in the `authors` table.
- **Creating the tables:**

```
cursor.execute('''CREATE TABLE IF NOT EXISTS authors
        (id INTEGER PRIMARY KEY, name TEXT,
        nationality TEXT, birth_year INTEGER)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS books
        (id INTEGER PRIMARY KEY, title TEXT,
        author_id INTEGER,
        FOREIGN KEY (author_id) REFERENCES authors (id))''')
conn.commit()
```

Example: Inserting Data

- **Insert authors:**

```
cursor.execute("INSERT INTO authors
(name, nationality, birth_year) VALUES (?, ?, ?)",
('Jane Austen', 'English', 1775))
cursor.execute("INSERT INTO authors
(name, nationality, birth_year) VALUES (?, ?, ?)",
('Charles Dickens', 'English', 1812))
conn.commit()
```

- **Insert books:**

```
cursor.execute("INSERT INTO books (title, author_id)
VALUES (?, ?)", ('Pride and Prejudice', 1))
cursor.execute("INSERT INTO books (title, author_id)
VALUES (?, ?)", ('Oliver Twist', 2))
conn.commit()
```

Example: Querying with Joins

- **JOIN clause:** Used to combine data from multiple tables based on related columns.
- **Retrieve book titles and author names:**

```
cursor.execute('''SELECT books.title, authors.name
    FROM books
    INNER JOIN authors ON books.author_id = authors.id''')
results = cursor.fetchall()
for row in results:
    print(row)
```

- **Output:**

```
('Pride and Prejudice', 'Jane Austen')
('Oliver Twist', 'Charles Dickens')
```

Key Takeaways

- **Databases:** Organized collections of data for efficient storage, retrieval, and analysis.
- **Relational databases:** Data is organized in tables with rows and columns.
- **SQL:** A powerful language for interacting with relational databases.
- **SQLite:** A lightweight, file-based database system ideal for learning and small projects.
- **Key concepts:** Tables, columns, rows, primary keys, foreign keys, SQL commands (‘SELECT‘, ‘INSERT‘, ‘UPDATE‘, ‘DELETE‘).
- **Python and SQLite:** Python’s ‘sqlite3‘ module allows seamless interaction with SQLite databases.
- **Python integration:** Connecting to a database, creating tables, inserting data, querying data, and committing changes.

Key Takeaways

- **Databases:** Organized collections of data for efficient storage, retrieval, and analysis.
- **Relational databases:** Data is organized in tables with rows and columns.
- **SQL:** A powerful language for interacting with relational databases.
- **SQLite:** A lightweight, file-based database system ideal for learning and small projects.
- **Key concepts:** Tables, columns, rows, primary keys, foreign keys, SQL commands (‘SELECT‘, ‘INSERT‘, ‘UPDATE‘, ‘DELETE‘).
- **Python and SQLite:** Python’s ‘sqlite3‘ module allows seamless interaction with SQLite databases.
- **Python integration:** Connecting to a database, creating tables, inserting data, querying data, and committing changes.

Exercise Session

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Exercise_L09.ipynb`

Self-Study Questions

1. What are the main advantages of using a database?
2. What is SQLite, and why is it suitable for learning?
3. Explain the difference between a table, a column, and a row in a relational database.
4. What is the purpose of a primary key?
5. How does a foreign key establish a relationship between tables?
6. Write a SQL query to retrieve all books published after a specific year.
7. How do you insert a new record into a table using SQL?
8. How do you update an existing record in a table using SQL?
9. How do you delete a record from a table using SQL?
10. How do you connect to an SQLite database and execute SQL commands using Python?

Additional Resources

- **SQLite Documentation:** `https://www.sqlite.org/docs.html`
- **Python's sqlite3 Module Documentation:** `https://docs.python.org/3/library/sqlite3.html`
- **W3Schools SQL Tutorial:** `https://www.w3schools.com/sql/`
- **SQLZoo Interactive SQL Tutorial:** `https://sqlzoo.net/`
- **DB Browser for SQLite:** `https://sqlitebrowser.org/` (A visual tool for managing SQLite databases)