# Python数据库操作 教案

周次：7

课次：1

章节内容：SQLite数据库

课时：2

## 单元教学目标

- 知识目标：能够列举SQLite的轻量级特性、嵌入式数据库优势以及与Python的集成方式，并理解其核心数据结构（如表、行、字段）   
- 技能目标：能够使用Python连接SQLite数据库，创建、查询和修改数据库表，并完成简单数据插入与查询操作   
- 素养目标：能够规范使用SQLite数据库，遵守操作规范，避免SQL注入，确保数据安全性和完整性

## 教学重点

• 掌握sqlite3模块的核心使用方法（如创建数据库、建立表、执行SQL语句）   
• 理解基本的数据库操作流程（如连接数据库、执行查询、提交事务）   
• 掌握数据插入与查询的实战技巧（如使用execute方法操作表数据）

## 教学难点

• 学生可能难以掌握如何正确使用参数化查询（如`execute()`方法）来防止SQL注入，尤其是在处理用户输入时，容易出现硬编码的SQL语句导致安全问题。   
• 如何正确关闭数据库连接（如`conn.close()`）并处理异常（如`try-except`块）是教学难点，学生可能因忽略连接管理导致程序崩溃或资源泄漏。

## 教学活动

新课导入【10分钟】   
1. 故事导入：教师讲述“小明的投票系统”案例，展示SQLite数据库如何记录投票数据，引发学生兴趣。   
2. 范例导入：展示Python代码片段（如`import sqlite3`），提问“如何用Python操作数据库？”引导学生思考。   
  
讲授新课【120分钟】   
1. \*\*讲授法（20分钟）\*\*   
 - 介绍SQLite基础概念（数据库、表、字段、主键）。   
 - 通过案例分析讲解创建数据库（`sqlite3.connect()`）和表（`create\_table()`）。   
 - 示范代码：   
 ```python   
 import sqlite3   
 conn = sqlite3.connect('votes.db')   
 cur = conn.cursor()   
 cur.execute('''CREATE TABLE IF NOT EXISTS votes (id INTEGER PRIMARY KEY, name TEXT)''')   
 conn.commit()   
 ```   
 - 重点强调：表结构设计规范（字段类型、主键设置）。   
  
2. \*\*讨论法（15分钟）\*\*   
 - 分组讨论：“如何避免数据库重复数据？”   
 - 教师引导学生思考索引、唯一约束等优化手段。   
  
3. \*\*练习法（30分钟）\*\*   
 - \*\*基础练习\*\*：学生用代码创建数据库并插入数据（如`INSERT INTO votes (name) VALUES ('Alice')`）。   
 - \*\*进阶练习\*\*：模拟事务处理（使用`try-except`块捕获异常，回滚操作）。   
 - 教师巡视指导，鼓励学生调试代码。   
  
4. \*\*案例分析法（25分钟）\*\*   
 - 教师提供“学生信息管理系统”案例，学生分组分析需求（如添加、查询、删除功能）。   
 - 通过代码片段讲解SQL语句与Python的交互逻辑。   
  
5. \*\*头脑风暴法（10分钟）\*\*   
 - 学生列举SQLite的常见操作（如`SELECT`、`UPDATE`、`DELETE`），教师总结关键点。   
  
巩固练习【30分钟】   
1. \*\*分层练习\*\*   
 - \*\*基础题\*\*：创建数据库并插入3条学生信息。   
 - \*\*进阶题\*\*：用`SELECT`查询学生姓名，并用`WHERE`条件过滤“性别为男”的记录。   
 - \*\*挑战题\*\*：用事务处理确保插入数据时的完整性（如数据丢失保护）。   
  
2. \*\*小组协作\*\*   
 - 学生分组完成练习，教师提供时间限制（如30分钟内完成），鼓励使用代码编辑器或IDE辅助。   
  
3. \*\*角色扮演\*\*   
 - 学生模拟“数据库管理员”角色，扮演开发者、测试员、运维员，讨论代码问题（如异常处理、性能优化）。   
  
归纳总结【10分钟】   
1. \*\*教师总结\*\*   
 - 强调重点：SQLite核心操作（创建、查询、更新、删除）、事务处理、异常捕获。   
 - 提问学生：“如何确保数据安全？”引导复述事务的原理。   
  
2. \*\*学生复述\*\*   
 - 学生用简短语句总结本课内容（如“SQLite是轻量级数据库，适合小型项目”）。   
 - 教师补充：SQLite的跨平台兼容性（支持Windows、Linux、macOS）。   
  
教学方法整合：讲授法+案例分析+练习法+讨论法+头脑风暴+角色扮演+游戏化练习（如“代码纠错挑战”）。

## 教学资源

</think>  
  
• \*\*教学设备\*\*：   
- 电脑或笔记本电脑（安装Python环境）   
- 屏幕显示器（用于展示教学内容）   
- 白板或投影仪（用于演示代码和操作）   
- 学生笔记本（用于记录课堂内容）   
- 数据库管理工具（如SQLite Viewer或SQLite Browser）   
  
• \*\*推荐参考资料或网站\*\*：   
- 《Python编程：从入门到实践》——适合初学者，包含数据库操作章节   
- Python官方文档（https://docs.python.org/3/）——提供完整API说明   
- SQLite官方文档（https://www.sqlite.org/docs/）——详细介绍SQLite语法和功能   
- GitHub上的Python SQLite项目（https://github.com/SQLite/SQLite）——开源代码示例   
- 网易云课堂或B站——有Python数据库操作的课程视频教程   
- 《Python数据库开发实战》——适合有一定基础的学习者   
- SQLite数据库管理工具（如SQLite Browser）——提供图形化界面操作数据库

## 教学反思

• 教学效果方面：需评估学生是否掌握SQLite基本操作（如连接、创建表、查询）及数据处理能力，是否能独立完成简单数据库操作任务   
• 学生反馈方面：关注学生对SQLite语法（如cursor使用、参数化查询）的理解程度，以及实践过程中遇到的常见问题（如安装配置障碍、数据类型错误）   
• 教学方法优化：需反思课堂讲解是否过于理论化，是否需要增加可视化演示（如使用SQLite Browser工具）或分步骤操作指导   
• 技术细节覆盖：评估是否充分讲解了数据库事务、索引等高级概念，以及是否通过代码示例帮助学生理解抽象概念   
• 实践任务设计：检查是否通过分组练习或项目任务（如学生自行创建数据库并完成增删改查）提升学习主动性   
• 差异化教学：考虑是否对不同水平学生（如基础薄弱者 vs 程序员）提供了针对性的辅助资源（如代码注释、示例对比）   
• 工具辅助使用：反思是否有效利用了SQLite的可视化工具（如Python的sqlite3模块与第三方库）提升学习体验   
• 常见错误排查：评估是否通过课堂演示或课后练习帮助学生解决典型错误（如语法错误、表结构不匹配）   
• 学习动机激发：反思是否通过实际案例（如学生管理系统的开发）增强学习相关性，提升参与度   
• 课后巩固设计：检查是否通过习题、代码调试任务或同伴互评等方式强化知识掌握程度

## 教学评价

• 课程目标与评价维度   
 - 评价维度涵盖知识掌握、技能应用、合作能力、创新思维等   
 - 结果性评价侧重项目成果（如数据库创建、SQL语句编写）   
 - 过程性评价关注学习过程（如课堂参与、作业完成度）   
  
• 过程性评价标准与方式   
 - \*\*课堂参与度\*\*   
 - 标准：主动提问、积极发言、配合小组任务   
 - 方式：教师观察记录、学生自评表（如“是否主动参与讨论”）   
 - \*\*作业完成情况\*\*   
 - 标准：按时提交、代码规范、逻辑清晰   
 - 方式：教师批改评分+学生提交的作业截图   
 - \*\*小组合作表现\*\*   
 - 标准：分工明确、沟通协作、任务完成度   
 - 方式：小组互评表（如“是否有效分配任务”）   
  
• 结果性评价标准与方式   
 - \*\*数据库操作能力\*\*   
 - 标准：能创建/查询数据库、执行INSERT/SELECT语句、处理异常   
 - 方式：教师现场测试（如编写SQL语句并验证结果）   
 - \*\*项目成果\*\*   
 - 标准：完成指定数据库设计（如学生信息表）、编写完整代码、测试运行正常   
 - 方式：提交项目代码+教师验收测试报告   
 - \*\*编程能力\*\*   
 - 标准：代码结构清晰、语法正确、注释完整   
 - 方式：教师代码审查+单元测试通过率（如使用pytest）   
  
• 评价权重与反馈机制   
 - 过程性评价占比：40%（课堂参与、作业、小组合作）   
 - 结果性评价占比：60%（项目成果、编程能力）   
 - 评价反馈：通过评分表+个性化反馈（如“需加强逻辑思维”）