Использование базы данных SQLite в Python

Python имеет встроенный модуль `sqlite3`, который позволяет:

- подключаться к базе данных;
- выполнять SQL-запросы;
- получать результаты;
- работать с транзакциями (сохранение изменений).

Так как SQLite хранит всё в одном файле, работа максимально проста: никакой настройки сервера не нужно.

Подключение к базе данных

Для начала подключимся к базе или создадим новую (если файла ещё нет).

```
    import sqlite3

# подключение к базе данных (файл создастся
автоматически, если его нет)
connection = sqlite3.connect("students.db")

# создание курсора для выполнения SQL-запросов
cursor = connection.cursor()
```

`connect("students.db")` — создаёт файл `students.db` в текущей папке;

`cursor()` — объект для выполнения SQL-запросов.

Создание таблицы из Python

Создадим таблицу студентов (как в прошлой лекции):

```
    cursor.execute('''
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER,
    email TEXT UNIQUE
)
''')
```

`IF NOT EXISTS` защищает от ошибки, если таблица уже есть.

`execute()` принимает строку с SQL-запросом.

Добавление данных

Добавим запись:

```
    cursor.execute('''
INSERT INTO students (name, age, email)
VALUES (?, ?, ?)
''', ("Иван Иванов", 20, "ivan@example.com"))

connection.commit()  # сохраняем изменения
```

Здесь используется `?` — **плейсхолдер**. Мы подставляем значения вторым параметром (`("Иван Иванов", 20, "ivan@example.com")`).

⚠ Это защищает от SQL-инъекций (важно для безопасности).

Добавление нескольких записей

```
    students = [
    ("Мария Петрова", 22, "maria@example.com"),
    ("Сергей Смирнов", 19, "sergey@example.com"),
    ("Анна Соколова", 21, "anna@example.com")
]

cursor.executemany('''
INSERT INTO students (name, age, email)
VALUES (?, ?, ?)
''', students)

connection.commit()
```

`executemany()` позволяет вставить сразу несколько строк.

Получение данных

Один результат

```
    cursor.execute("SELECT * FROM students WHERE name =
?", ("Иван Иванов",))
```

```
student = cursor.fetchone()
print(student)
```

`fetchone()` возвращает одну строку (или `None`).

Несколько результатов

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

`fetchall()` возвращает список всех строк.

Итерация напрямую по курсору

```
    for row in cursor.execute("SELECT name, age FROM
students ORDER BY age"):
    print(row)
```

Обновление данных

```
    cursor.execute('''
UPDATE students
SET age = ?
WHERE email = ?
''', (21, "ivan@example.com"))

connection.commit()
```

Удаление данных

```
    cursor.execute('''
DELETE FROM students
WHERE age < ?
''', (20,))

connection.commit()
```

Работа с транзакциями

- connection.commit() сохраняет изменения.
- connection.rollback() откатывает изменения при ошибках.

Пример с откатом:

```
    try:
    cursor.execute("INSERT INTO students (name, age,
email) VALUES (?, ?, ?)",
                   ("Пробный", 25, "test@example.com"))
    raise Exception("Ошибка во время работы")  #
симулируем сбой
    connection.commit()
except:
    connection.rollback()
    print("Изменения отменены")
```

Работа с контекстным менеджером

Чтобы не забывать закрывать соединение, используем with:

```
    import sqlite3

with sqlite3.connect("students.db") as conn:
    cursor = conn.cursor()
    cursor.execute("SELECT COUNT(*) FROM students")
    print(cursor.fetchone())
```

Здесь соединение автоматически закрывается после блока with.

Пример: простая программа «Студенты»

```
    import sqlite3

def init_db():
    conn = sqlite3.connect("students.db")
    cursor = conn.cursor()
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS students (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
        name TEXT NOT NULL,
        age INTEGER,
        email TEXT UNIQUE
    )
    ''')
    conn.commit()
    return conn

def add_student(conn, name, age, email):
    cursor = conn.cursor()
    cursor.execute("INSERT INTO students (name, age,
email) VALUES (?, ?, ?)",
                   (name, age, email))
    conn.commit()

def list_students(conn):
    cursor = conn.cursor()
    cursor.execute("SELECT id, name, age, email FROM
students")
    for row in cursor.fetchall():
        print(row)

# запуск
conn = init_db()
add_student(conn, "Алексей", 23, "alex@example.com")
list_students(conn)
```