10. 데이터 베이스(DB) 연동 #src: 8_Python/ch10_데이터베이스연동 참고

1) SQLite 데이터 베이스 연결

(1) SQLite와 Python #DB Browser for SQLite다운로드: https://sqlitebrowser.org/dl/

- SQLite DB는 별도의 서버 프로세스 없이 SQL을 사용하여 DB 액세스 할 수 있도록 만든 간단한 디스크 기반 데이터 베이스를 제공하는 C 라이브러리

(2) 데이터 베이스 연결: DB연결 객체 connection 생성

sqlite3.connect('파일명.db'); DB연결 객체 생성시 해당 파일 연결(없는 파일일 경우 파일 생성)

(3) SQLite API; 모듈은 SQLite DB를 위한 DB-API2.0 사양을 준수하는 SQL인터페이스 제공

① splite3.connection; DB연결 객체로, DB마다 객체 생성 방법이 다르다

속성/메소드	설명
cursor()	커서 객체 생성
commit()	현재 트랜잭션을 커밋(변경사항 적용)
rollback()	마지막 호출 이후 모든 변경사항을 롤백(취소)
close()	데이터베이스 연결을 닫음. commit() 하지 않고 DB연결 닫을 시 변경사항 손실

② splite3.cursor; DB질의문을 실행하고 결과를 가져오기 위한 객체 #커서 닫을 때도 close()

속성/메소드	설명
execute(sql, {parameters})	SQL문을 실행. SQL문을 매개변수화 할 수 있음 qmark: cursor.execute("insert into tableN values (?, ?)", (name, age)) named: cursor.execute("select * from tableN where name=:name", {"name":name})
fetchone()	쿼리 결과를 1행씩 가져옴 (더 이상 데이터가 없을 경우 None 반환)
fetchmany(n)	쿼리 결과를 n행씩 가져와서 목록을 반환 (데이터 없으면 빈 목록[] 반환)
fetchall()	쿼리 결과의 모든 행을 가져와서 목록을 반환 (데이터 없으면 빈 목록[] 반환)

(4) SQLite 데이터 베이스에 데이터 입력/조회하기: Create, Read(SELECT), Update, Delete

① sqlite3.connect; DB연결 객체(conn) 생성 → ② cursor.execute("SQL문") → ┌#조회는 ④생략

③ 데이터 처리; fetchall()메소드와 반복문 이용 → ④ conn.commit(); 데이터 수정時 → ⑤ close()

2) Oracle 데이터 베이스 연결 #cx_Oracle 설치: pip install cx_Oracle or conda install cx_Oracle

```
oracle_dsn = cx_Oracle.makedsn(host="localhost", port=1521, sid="xe") #Oracle 버전 등 확인必
conn = cx_Oracle.connect("scott", "tiger", dsn=oracle_dsn) #user, password 등 확 必
```

3) MariaDB 연결 #pymysql 설치(pip install pymysql)

```
pymysql.connect(host="localhost", port=3306, db="kimdb", user="root", passwd="mysql",
                charset="utf8", autocommit=True) #mySQL 의 port, user, password 등 확인必
```

※데이터(튜플, 리스트 등)를 데이터 프레임으로 변경: pandas.DataFrame("데이터") #pandas 패키지