문제 1: FastAPI 기반 간단한 CRUD API 구현

📋 문제 개요

데이터베이스를 사용하지 않고 메모리(Python 리스트와 딕셔너리)만을 활용하여 **학생 정보 관리 시스템**의 REST API를 구현하세요.

🎯 학습 목표

- **CRUD 개념 이해**: Create, Read, Update, Delete 기본 개념 습득
- **FastAPI 라우팅**: HTTP 메서드별 엔드포인트 구현 방법
- **데이터 모델링**: Pydantic을 활용한 입력/출력 모델 정의
- **메모리 저장소**: 리스트와 딕셔너리를 활용한 데이터 관리
- **에러 처리**: HTTP 상태 코드와 예외 처리 방법
- **API 문서화**: FastAPI 자동 문서 생성 활용

📊 데이터 구조

학생(Student) 정보

- **id**: 정수형, 자동 증가하는 고유 식별자
- **name**: 문자열, 학생 이름 (필수)
- **age**: 정수형, 나이 (필수)
- **major**: 문자열, 전공 (필수)
- **grade**: 문자열, 학년 (선택사항)

🛠️ 구현해야 할 API 엔드포인트

1. 루트 엔드포인트

- **경로**: `GET /`
- **기능**: API 소개 및 현재 등록된 학생 수 표시
- **응답 예시**:

```json
{
  "message": "학생 정보 관리 시스템 API",
  "total_students": 5,
  "available_endpoints": ["/students", "/students/{id}"]
}
```

2. 전체 학생 조회

- **경로**: `GET /students`
- **기능**: 등록된 모든 학생 정보를 조회
- **응답**: 학생 객체들의 리스트
- **상태 코드**: 200 OK

3. 특정 학생 조회

- **경로**: `GET /students/{student_id}`
- **기능**: ID로 특정 학생의 상세 정보 조회
- **매개변수**: `student_id` (정수형)
- **응답**: 해당 학생의 정보 객체
- **상태 코드**:
  - 200 OK (학생을 찾은 경우)
  - 404 Not Found (해당 ID의 학생이 없는 경우)

4. 새 학생 등록

- **경로**: `POST /students`
- **기능**: 새로운 학생 정보를 등록
- **요청 본문**: 학생 정보 (id 제외)
- **응답**: 생성된 학생 정보 (id 포함)
- **상태 코드**: 201 Created

5. 학생 정보 수정

- **경로**: `PUT /students/{student_id}`
- **기능**: 기존 학생의 정보를 전체 수정
- **매개변수**: `student_id` (정수형)
- **요청 본문**: 수정할 학생 정보 (id 제외)
- **응답**: 수정된 학생 정보
- **상태 코드**:
  - 200 OK (수정 성공)
  - 404 Not Found (해당 ID의 학생이 없는 경우)

6. 학생 정보 삭제

- **경로**: `DELETE /students/{student_id}`
- **기능**: 특정 학생 정보를 삭제

- **매개변수**: `student_id` (정수형)
- **응답**: 삭제 확인 메시지
- **상태 코드**:
  - 200 OK (삭제 성공)
  - 404 Not Found (해당 ID의 학생이 없는 경우)

📝 구현 요구사항

1. 필수 라이브러리

```bash
pip install fastapi uvicorn
```

2. Pydantic 모델 정의

- 입력용 모델 (`StudentCreate`): id 제외한 학생 정보
- 응답용 모델 (`StudentResponse`): id 포함한 완전한 학생 정보
- 필수 필드와 선택사항 필드 구분

3. 데이터 저장소

- 전역 변수로 학생 정보를 저장할 리스트 생성
- 자동 증가하는 ID 관리 변수

4. 에러 처리

- `HTTPException`을 사용하여 적절한 에러 응답
- 404 에러 시 한국어 메시지 포함

5. 초기 테스트 데이터

앱 시작 시 다음 학생 정보들을 미리 등록해 두세요:

- 김철수, 20세, 컴퓨터공학과, 2학년
- 이영희, 21세, 경영학과 (학년 정보 없음)
- 박민수, 19세, 전자공학과, 1학년

🔍 테스트 방법

1. 서버 실행

bash

```
uvicorn main:app --reload
```

2. 자동 API 문서 확인

브라우저에서 `http://localhost:8000/docs` 접속

3. 기본 테스트 시나리오

1. 전체 학생 목록 조회
2. 새 학생 추가
3. 특정 학생 조회
4. 학생 정보 수정
5. 학생 삭제
6. 존재하지 않는 학생 조회 (404 에러 확인)

💡 힌트

HTTP 상태 코드 사용법

python

```
from fastapi import HTTPException

# 404 에러
raise HTTPException(status_code=404, detail="학생을 찾을 수 없습니다")

# 201 상태 코드로 응답
@app.post("/students", status_code=201)
```

경로 매개변수 처리

python

```
@app.get("/students/{student_id}")
async def get_student(student_id: int):
    # student_id 사용
```

CRUD 기본 패턴

python

```python
# CREATE - 새 항목 추가
new_student = {"id": next_id, "name": name, ...}
students.append(new_student)
next_id += 1

# READ - 항목 조회
for student in students:
    if student["id"] == student_id:
        return student

# UPDATE - 항목 수정
for i, student in enumerate(students):
    if student["id"] == student_id:
        students[i] = updated_student
        return students[i]

# DELETE - 항목 삭제
for i, student in enumerate(students):
    if student["id"] == student_id:
        deleted = students.pop(i)
        return deleted
```

📋 체크리스트

완성 후 다음 항목들을 확인해보세요:

- [ ] **CREATE**: 새 학생 추가가 정상 작동하는가?
- [ ] **READ**: 전체 학생 조회가 정상 작동하는가?
- [ ] **READ**: 특정 학생 조회가 정상 작동하는가?
- [ ] **UPDATE**: 학생 정보 수정이 정상 작동하는가?
- [ ] **DELETE**: 학생 삭제가 정상 작동하는가?
- [ ] 404 에러가 적절히 처리되는가?
- [ ] API 문서가 자동 생성되는가?
- [ ] 초기 테스트 데이터가 로드되는가?
- [ ] ID가 자동으로 증가하는가?

🚀 추가 도전 과제 (선택사항)

CRUD의 확장 개념을 연습해보세요:

1. **조건부 조회**: 전공별 학생 조회 `GET /students/major/{major_name}`
2. **부분 수정**: PATCH 메서드로 일부 필드만 수정하기
3. **검색 기능**: 이름으로 학생 검색 `GET /students/search?name=검색어`

4. **정렬 기능**: 나이순, 이름순 정렬 옵션 추가
5. **페이징**: 대량 데이터 처리를 위한 페이지네이션 구현