행정 AI 기술 실습 · 통합 교재

AI로 만든 것을, 직접 연결하고 운영하기

문서·데이터 → API → 공공데이터 → 클라우드 배포 → n8n 자동화

지역 도서관 현황 대시보드로 배우는 첫 번째 기술 실습 시리즈

2026.09.20 초판 · 학습 권장 295분

웹 교재: https://aiedu.gdiaxhub.com/tech

차례

01. 마크다운·CSV·JSON 기초 · 50분

02. API를 처음 만나는 행정 실무자 · 45분

03. 공공데이터에서 보고서까지 · 70분

04. 바이브코딩 다음은 클라우드 배포 · 70분

05. n8n 첫 업무 자동화 · 60분

코드와 가상 자료는 교육·업무에 수정·재사용할 수 있습니다. 실제 공공데이터의 이용조건은 별도 확인합니다. 외부 계정의 배포·인증·n8n UI 실행은 학습 환경에서 확인해야 합니다.

LESSON 01

마크다운·CSV·JSON 기초

AI와 일하기 위한 문서와 데이터 형식

완성물: 보고서 메모와 구조화한 데이터 · 준비: 브라우저 · 편집기 · Python

완성 모습과 준비물

이번 실습의 결과물은 `report.md` 보고서와 `sample.csv`를 변환한 JSON입니다. 이어지는 API·배포·자동화 수업도 같은 도서관 자료를 씁니다. 가상 지역 3곳, 도서관 6곳, 좌석 합계 400석이 출발점입니다.

전체 실습 키트 ZIP을 내려받아 압축을 완전히 푸세요. 파일 미리보기 화면에서 실행하지 않습니다. `README.md`가 있는 `tech-practice` 폴더를 편집기로 엽니다. 편집기는 메모장도 가능하지만 코드 편집기를 쓰면 구조를 보기 쉽습니다. 변환 실습에는 Python 3.10 이상이 필요합니다.

휴대전화에서는 교재와 완성 대시보드를 볼 수 있습니다. 파일 편집과 터미널 실행은 PC를 기준으로 설명합니다. 먼저 원본 폴더를 복제해 `tech-practice-backup`으로 보관하세요.

1단계 · 문서·표·프로그램의 언어 구별하기

형식	이 실습의 파일	잘하는 일	주의할 점
Markdown	report.md	제목·목록·링크로 보고서 구조 표현	편집기마다 표·체크박스 지원이 다를 수 있음
CSV	sample.csv	열과 행으로 같은 종류의 기록 저장	쉼표가 포함된 값은 따옴표로 감싸기
JSON	libraries.json	프로그램 사이에서 구조화한 데이터 교환	큰따옴표, 숫자, null의 차이 지키기

한 자료를 세 번 작성하는 것이 아닙니다. CSV를 원본 표로 보존하고, JSON으로 화면을 만들고, Markdown으로 결과를 설명합니다. AI에게도 "보고서용 문장"인지 "컴퓨터가 읽을 JSON"인지 출력 형식을 지정해야 합니다.

2단계 · Markdown으로 검토 메모 쓰기

`report.md`를 열어 내용을 읽습니다. `#`는 큰 제목, `##`는 절 제목, `-`는 목록입니다. 아래 예시를 문서 끝에 붙이고 저장하세요. 편집기의 Markdown 미리보기가 있으면 원문과 나란히 비교합니다.

```
## 추가 검토
- 집계 대상: 해솔시 도서관 3곳
- 좌석 합계: 200석
- [ ] 원본과 합계를 비교했다
- [ ] 출처와 기준일을 적었다

> 좌석 합계만으로 지역의 독서 수요를 판단하지 않는다.

[원본 데이터](./sample.csv)
```

성공 확인: 제목의 단계가 보이고, 두 개의 할 일과 원본 링크가 구별됩니다. 체크박스가 단순 문자로 보이더라도 원문은 유지됩니다. 실제 공개 URL이 아닌 상대 경로 링크는 파일 위치에 따라 달라집니다.

AI에게 요청하기: "위 Markdown의 숫자와 출처는 바꾸지 말고, 사실·해석·추가 확인을 구분해 문장을 다듬어줘. 근거 없는 원인 설명은 추가하지 마." 결과를 붙이기 전에 3곳·200석이 유지되는지 확인하세요.

3단계 · CSV의 한 행을 이해하기

`sample.csv`를 텍스트 편집기로 열어 첫 두 줄을 봅니다. 엑셀에서는 데이터 가져오기를 사용해 UTF-8을 선택하면 한글을 확인하기 쉽습니다.

```
도서관명,시군구명,열람좌석수,데이터기준일자
해솔중앙도서관,해솔시,120,2026-09-01
```

첫 줄은 열 이름이고, 다음 줄부터 한 도서관입니다. 열 이름을 바꾸면 뒤의 변환 프로그램이 해당 열을 찾지 못합니다. 이름에 쉼표가 있는 경우 `"해솔, 중앙도서관"`처럼 값 전체를 감쌉니다. 숫자 `120`에는 단위 "석"을 붙이지 않습니다.

빈 좌석 값은 **모름**, `0`은 **좌석 없음**입니다. 이 차이를 지켜야 모르는 값을 합계 0으로 잘못 해석하지 않습니다. 원본을 고칠 때는 사본에서 작업하고, 수정한 행과 이유를 메모하세요.

4단계 · CSV를 JSON으로 변환하기

터미널의 현재 폴더가 `tech-practice`인지 확인합니다. Windows는 해당 폴더에서 터미널을 열고 아래 첫 명령을 실행합니다. macOS·Linux는 두 번째 명령을 씁니다. 두 명령을 모두 실행할 필요는 없습니다.

```
py convert_csv.py sample.csv --sample --source "교육용 가상 데이터"
```

```
python3 convert_csv.py sample.csv --sample --source "교육용 가상 데이터"
```

`완료: 6행 → dashboard/data/libraries.json` 이 나오면 성공입니다. JSON 파일의 `items` 안에는 6개 객체가 있습니다. 프로그램은 수집일을 실행 날짜로 기록하고, CSV의 기준일은 각 행에 보존합니다.

```
{"name":"해솔중앙도서관","city":"해솔시","seats":120,"date":"2026-09-01"}
```

JSON에서 `"120"` 은 문자열, `120` 은 숫자, `null` 은 값 없음입니다. 마지막 항목 뒤에 쉼표를 넣거나 작은따옴표를 쓰면 JSON 문법 오류가 납니다. 이 실습에서는 손으로 대량 수정하지 않고 CSV를 고친 뒤 다시 변환하세요.

5단계 · 숫자를 화면에서 확인하기

Windows는 `py server.py` , macOS·Linux는 `python3 server.py` 를 실행합니다. 터미널을 켜둔 채 브라우저에서 `http://127.0.0.1:8765` 를 엽니다. 전체 6곳·400석, 지역을 해솔시로 바꾸면 3곳·200석이어야 합니다. 서버 종료는 터미널에서 Ctrl+C입니다.

작은 실험: 사본 CSV에서 첫 행 좌석을 빈 칸으로 바꾸고 다시 변환합니다. 화면을 새로 고치면 전체 6곳·280석·좌석 미상 1곳이 됩니다. 도서관 수는 유지되고, 알려진 좌석만 더해지는 것을 확인하세요.

막혔을 때와 원상 복구

증상	확인 순서
py 또는 python3를 찾지 못함	Python 설치 여부 확인 → 터미널 다시 열기 → 버전 확인
파일을 찾지 못함	압축 해제 여부 → 현재 폴더에 convert_csv.py가 있는지 확인
필수 열 누락	CSV 첫 줄을 sample.csv의 열 이름과 비교
숫자 변환 중단	“120석”, 음수, 임의 문자열 제거 → 원본을 보고 수정
화면 데이터가 바뀌지 않음	변환 성공 메시지 → 브라우저 새로고침 → JSON 파일 확인

실험 전 백업한 `sample.csv` 로 교체한 뒤 같은 변환 명령을 다시 실행하면 기본값으로 돌아갑니다. 변환 도중 오류가 나면 기존 JSON은 유지되도록 만들었습니다. 오류를 무시하고 “성공했다”고 보고서에 적지 마세요.

완료 기준과 다음 실습

- Markdown에서 사실·해석·확인할 일을 구분했다.
- CSV 6행과 JSON 6개 항목을 비교했다.
- 전체 400석과 해솔시 200석을 직접 확인했다.
- 빈 값과 0의 차이를 설명할 수 있다.

다음 교재에서는 이 자료를 **주소로 요청하고 JSON으로 응답받는 API**로 읽습니다. 제출물은 `report.md`와 결과 화면 한 장이면 됩니다. 공개 게시판에 올릴 때 실제 업무자료는 제외하고 가상 데이터만 사용하세요.

참고 문서

- CommonMark 문법: Markdown의 기본 문법. 표·체크박스는 확장 문법입니다.
- Python csv: 따옴표와 열 구분을 직접 구현하지 않고 CSV 라이브러리를 쓰는 이유.
- Python json: JSON과 Python 자료형의 대응.

LESSON 02

API를 처음 만나는 행정 실무자

주소로 요청하고 JSON으로 응답받기

완성물: 조건 조회·페이지 나누기·오류 확인 · 준비: 1강 실습 키트 · Python

완성 모습과 준비물

브라우저 주소를 바꿔 도서관 목록을 조회하고, 요청과 응답의 관계를 설명하는 것이 목표입니다. 실제 서비스의 인증키를 신청하기 전에 **내 PC의 연습 API**로 성공·오류·페이지 나누기를 경험합니다. 이 API는 공공데이터포털 서버가 아닙니다.

실습 키트를 압축 해제하고 Python 3.10 이상을 준비하세요. 앞 교재에서 데이터를 수정했다면 원본 `sample.csv`를 복원하고 변환부터 다시 실행합니다. 스마트폰에서는 개념과 교재를 읽고, 실행은 PC에서 진행하세요.

1단계 · API를 요청서와 응답서로 보기

API는 프로그램이 다른 프로그램에 정해진 형식으로 요청하는 접점입니다. 이번 예제의 계약은 "GET으로 도서관을 조회하면 JSON을 돌려준다"입니다. API가 반드시 AI를 뜻하지는 않습니다.

요소	예시	뜻
메서드	GET	조회 요청
주소	/api/libraries	어떤 자료를 요청하는지
쿼리	city=해솔시	조회 조건
상태 코드	200 / 400	HTTP 수준의 성공 / 잘못된 요청
응답 본문	items 배열	실제 돌려받은 데이터

GET 조회와 POST 저장은 역할이 다릅니다. 다른 서비스에서 POST 예시를 보았다고 조회 주소에 그대로 적용하지 마세요. 실제 API는 HTTP 200이어도 본문 안에 오류 코드가 있을 수 있어, 상태 코드와 본문을 모두 확인합니다.

2단계 · 연습 서버 켜기

`tech-practice` 폴더에서 아래 중 운영체제에 맞는 명령 하나를 실행합니다.

```
py server.py
```

```
python3 server.py
```

터미널에 표시된 주소를 열고, 이어서 아래 주소를 브라우저 주소창에 입력합니다.

```
http://127.0.0.1:8765/api/libraries
```

`127.0.0.1`은 지금 쓰는 컴퓨터 자신입니다. 친구나 휴대전화에 이 주소를 보내도 내 PC에 접속하는 것이 아닙니다. 연습 서버는 외부에 공개하지 않도록 이 주소에만 연결합니다.

성공 확인: `total`은 6, `page`는 1, `limit`은 3, `items`는 3개입니다. 목록이 3개만 보인다고 전체가 3개인 것은 아닙니다.

3단계 · 조건과 페이지 바꾸기

다음 주소를 차례로 열어 응답을 비교하세요. 주소창에서 한글이 `%`가 포함된 문자열로 바뀌는 것은 URL 인코딩입니다.

```
http://127.0.0.1:8765/api/libraries?city=해솔시&page=1&limit=2
http://127.0.0.1:8765/api/libraries?city=해솔시&page=2&limit=2
```

첫 응답은 전체 3곳 중 2곳, 두 번째 응답은 나머지 1곳입니다. `?`는 조건의 시작, `&`는 조건 사이의 구분입니다. `total`은 조건에 맞는 전체 개수이고 `items.length`는 이번 응답의 개수입니다.

직접 계산: 첫 페이지 좌석 합계는 180석, 두 번째는 20석입니다. 두 페이지를 합쳐야 해솔시 200석입니다. 대량 자료를 가져올 때 첫 페이지의 일부만 보고 전체 통계라고 보고하는 실수를 피하세요.

4단계 · 오류를 일부러 만들기

```
http://127.0.0.1:8765/api/libraries?limit=0
```

응답 본문에는 limit 범위 오류가 나옵니다. 브라우저 개발자 도구의 Network(네트워크)를 열고 새로고침한 뒤 이 요청을 선택하면 상태 **400**도 확인할 수 있습니다. 정상 주소로 바꾸면 **200**으로 돌아옵니다.

`city=없는도시`는 잘못된 문법이 아니므로 200과 빈 배열이 나옵니다. "검색 결과 없음"과 "서버 고장"을 구별해야 화면에서 올바른 안내를 할 수 있습니다.

실제 서비스에서 만날 코드	첫 확인
401 / 403	인증키, 이용 신청 승인, 접근 권한

실제 서비스에서 만날 코드	첫 확인
404	문서의 정확한 주소와 경로
429	호출 한도, 재시도 간격
5xx	제공 서버 상태, 일시 장애

이 연습 서버가 위 모든 오류를 재현하는 것은 아닙니다. 400·404와 빈 결과를 직접 확인하고, 나머지는 실제 API 연동 시 구별할 기준으로 알아둡니다.

5단계 · 화면과 API의 차이 확인하기

연습 대시보드는 `dashboard/data/libraries.json` 파일을 읽습니다. `/api/libraries`는 같은 파일을 읽고 조건에 맞게 골라 응답합니다. 화면과 API가 자료를 공유하지만, 화면이 자동으로 API를 호출하도록 구성한 것은 아닙니다.

AI에게 이렇게 요청해 확장할 수 있습니다.

```
현재 app.js의 파일 조회를 /api/libraries 조회로 바꾸는 방안을 설명해줘.
limit=100이어도 모든 페이지를 가져오도록 하고,
HTTP 오류와 items 누락을 검사해줘.
먼저 파일 기반 버전과 서버가 필요한 버전의 차이를 설명해줘.
수정 전에 원본을 백업하고, 실패하면 오류를 화면에 보여줘.
```

다음 배포 교재는 서버가 필요 없는 **파일 기반 원본 버전**을 사용합니다. API 연결 버전을 정적 호스팅에 그대로 올리면 `/api/libraries`가 없어 실패합니다. 서버 코드와 정적 파일은 배포 방식이 다르다는 점이 이 단계의 핵심입니다.

막혔을 때와 종료

- 연결 거부: 서버를 실행한 터미널이 열려 있는지 확인합니다.
- 포트 사용 중: 앞 실습의 서버가 켜져 있다면 그대로 쓰거나 그 터미널에서 Ctrl+C로 종료한 뒤 다시 실행합니다.
- JSON 대신 HTML: 오타가 있는 경로의 404 페이지인지 확인합니다.
- 실제 외부 API에서 CORS 오류: 브라우저 접근 허용 정책 문제일 수 있습니다. 인증키를 프런트엔드에 넣는 우회 대신 서버에서 요청하는 구조를 검토합니다.

서버는 Ctrl+C로 끝납니다. 파일을 변경하지 않았다면 추가 복원은 필요 없습니다. 주소 전체에 인증키가 있는 실제 요청은 화면 캡처나 공개 질문에 붙이지 마세요.

완료 기준과 다음 실습

- GET·주소·쿼리·응답을 내 말로 설명했다.

- 두 페이지를 합쳐 3곳·200석임을 확인했다.
- HTTP 400과 정상적인 빈 결과를 구별했다.
- 정적 파일 배포와 API 서버 배포가 다름을 이해했다.

다음은 실제 공공데이터를 찾고, 출처를 남기며 CSV를 같은 대시보드 형식으로 바꾸는 실습입니다.

참고 문서

- MDN HTTP 개요
- MDN HTTP 응답 상태 코드
- Python http.server: 이 실습용 서버를 공용 운영 서버로 쓰지 않는 이유.

LESSON 03

공공데이터에서 보고서까지

자료의 출처를 남기고 실제 CSV로 교체하기

완성물: 출처와 기준일이 있는 현황 대시보드 · 준비: 1~2강 · 공공데이터 CSV

완성 모습과 자료의 범위

공공데이터포털에서 내려받은 도서관 CSV를 JSON으로 바꾸고, 지역별 현황 화면과 출처가 있는 검토 메모를 만듭니다. 첫 실행은 키트의 **가상 CSV**로 연습하고, 그다음 실제 CSV로 교체합니다. 연습 수치 6곳·400석을 실제 통계로 쓰지 않습니다.

전국도서관표준데이터는 이 교재의 실제 자료 탐색 출발점입니다. 포털의 화면과 제공 범위는 바뀔 수 있습니다. 2026-09-20에 자료 페이지와 열 이름을 확인했으며, 학습자 계정의 다운로드·API 승인은 직접 진행해야 합니다.

준비물은 실습 키트, Python 3.10 이상, 웹브라우저입니다. 실제 자료를 확보하지 못해도 가상 CSV로 변환·검증까지 마칠 수 있습니다.

1단계 · 데이터보다 질문부터 정하기

질문을 "우리 지역의 도서관은 몇 곳이고, 알려진 좌석은 몇 석인가?"로 좁힙니다. 전국 모든 시설을 비교하거나 수요를 예측하는 것은 이번 실습 범위를 넘습니다.

`report.md`의 사본을 `report-public.md`로 저장하고 다음 항목을 먼저 적으세요.

```
# 도서관 현황 검토
- 대상 지역: 직접 입력
- 포함 범위: 내려받은 파일의 실제 범위
- 자료명: 전국도서관표준데이터
- 출처 URL: https://www.data.go.kr/data/15013109/standard.do
- 제공기관: 선택한 자료의 표시값
- 자료 기준일: 각 행 확인
- 다운로드 날짜: 오늘 날짜
- 이용조건: 자료 페이지에서 확인한 내용
```

성공 확인: "자료 기준일"과 "다운로드 날짜"를 따로 적었습니다. 오늘 내려받았다고 자료가 오늘의 현황인 것은 아닙니다.

2단계 · 포털에서 파일 확보하기

1. 위 공식 자료 페이지를 엽니다. 자료 설명과 제공범위를 읽습니다.
2. 기관별 데이터가 필요하면 기관명으로 검색하고 해당 자료를 선택합니다. 통합 자료를 쓰면 어느 지역까지 포함되는지 확인합니다.
3. 활용 정보의 파일 다운로드에서 CSV를 찾습니다. 다운로드 창에서 계정 로그인을 요구하면 직접 로그인합니다.
4. 원본 파일은 별도 보관하고, 작업용 사본을 `tech-practice/portal.csv`로 저장합니다. 확장자가 `.csv.csv`가 되지 않게 확인합니다.
5. 파일에 필요한 네 열이 있는지 확인합니다: 도서관명, 시군구명, 열람좌석수, 데이터기준일자.

포털에서 조회·다운로드한 결과가 일부 범위라면 그 범위를 보고서에 적습니다. 목록 페이지의 기관 수와 CSV의 도서관 수를 같은 값으로 취급하지 않습니다. 통합 자료와 개별 기관 자료는 갱신 시점이 다를 수 있으므로 같은 날짜의 숫자라고 가정하지 마세요.

3단계 · 가상 자료로 변환기 먼저 확인하기

ZIP을 압축 해제한 폴더에서 실행합니다. Windows에서는 아래 `python3`를 `py`로 바꿉니다.

```
python3 convert_csv.py sample.csv --sample --source "교육용 가상 데이터"
python3 server.py
```

브라우저에서 `http://127.0.0.1:8765`를 열면 6곳·400석입니다. 이것이 변환기와 화면이 함께 작동한다는 기준선입니다. 화면이 안 나오면 실제 파일을 넣기 전에 이 상태부터 해결하세요.

프로그램은 다음처럼 이름을 바꿉니다. 날짜는 원본 값을 보존하며, 빈 좌석은 `null`로 유지합니다.

원본 CSV 열	JSON 필드	화면에서 의미
도서관명	name	시설 이름
시군구명	city	지역 선택 조건
열람좌석수	seats	알려진 좌석 합계
데이터기준일자	date	행별 자료 기준일

4단계 · 실제 자료로 교체하기

서버 터미널은 그대로 두고 같은 폴더에서 새 터미널을 엽니다. 실제 `portal.csv`가 준비되면 다음을 실행합니다. 실제 자료에는 `--sample`을 붙이지 않습니다.

```
python3 convert_csv.py portal.csv --source "전국도서관표준데이터 ·
https://www.data.go.kr/data/15013109/standard.do"
```

완료 메시지의 행 수를 기록하고 대시보드를 새로 고칩니다. 화면의 표시가 `[공공데이터]`로 바뀌어야 합니다. 좌석 합계는 실제 파일에 따라 달라지며, 400석과 같을 필요가 없습니다.

한글 인코딩 오류가 나고 원본이 CP949라면 다음처럼 다시 시도합니다. 인코딩을 바꿔 읽는 것이며 원본을 수정하는 명령은 아닙니다.

```
python3 convert_csv.py portal.csv --encoding cp949 --source "전국도서관표준데이터 ·
https://www.data.go.kr/data/15013109/standard.do"
```

행 수 확인: CSV를 표로 열어 머리글을 제외한 기록 수를 세고 변환 결과와 비교합니다. 중복 도서관이 있는지 이름·지역·주소를 함께 검토하세요. 이 변환기는 중복을 자동 삭제하지 않습니다. 같은 이름이 반드시 같은 시설이라는 보장이 없기 때문입니다.

5단계 · 화면에서 보고서로 연결하기

대상 지역을 선택하고 도서관 수·좌석 합계·미상 개수를 기록합니다. 원본에서 3개 행을 직접 찾아 값이 같은지 대조하세요. 자료 기준일이 여러 개라면 그 범위도 적습니다.

AI에는 확인한 숫자만 전달해 초안을 요청합니다.

```
아래 집계로 행정 검토 메모를 작성해줘.
[내가 확인한 지역, 도서관 수, 좌석 합계, 미상 개수, 출처, 기준일]
구성은 사실 / 해석의 한계 / 추가 확인 순서로 해줘.
제공하지 않은 수치와 원인을 만들지 마.
이용자 수·운영시간이 없으므로 수요나 효율을 단정하지 마.
```

AI 답변과 원본 숫자를 다시 비교하고 `report-public.md`에 반영합니다. 검토 책임은 사람에게 있습니다. 도서관 수 증가·감소를 말하려면 같은 범위와 정의로 집계한 과거 자료가 추가로 필요합니다.

확장 · CSV 다운로드를 API로 바꾸려면

자료 페이지의 오픈 API 안내에서 활용 신청, 서비스 URL, 요청 변수, 응답 예시, 호출 한도를 확인합니다. 같은 포털의 API라도 주소와 응답 구조가 같다고 가정하지 마세요. 이번 교재는 특정 인증키를 발급하거나 실제 인증 API 호출에 성공했다고 주장하지 않습니다.

실제 연결 순서는 다음과 같습니다.

1. 해당 API의 활용 신청과 승인을 확인합니다.

2. 문서에 제시된 최소 요청을 서버나 로컬 프로그램에서 한 번 실행합니다.
3. HTTP 상태와 본문의 서비스 결과 코드를 함께 확인합니다. 오류가 XML로 오는 경우도 구별합니다.
4. 페이지 크기·전체 건수·종료 조건을 확인하고 제한된 범위부터 수집합니다.
5. 받은 필드를 이 키트의 name·city·seats·date로 매핑합니다.
6. CSV 집계와 비교한 후 자동 갱신으로 확장합니다.

인증키는 브라우저 JS나 공개 저장소에 넣지 않습니다. 문서에서 인코딩된 키와 원문 키를 구별하고, URL 생성 과정에서 이중 인코딩하지 않게 확인합니다. 오류 상담 시 키가 포함된 주소는 가립니다.

오류 해결·복원·완료 기준

증상	해결
필수 열 누락	실제 머리글을 확인하고 작업용 CSV의 열 이름을 맞춤. 다른 의미의 열을 억지로 대응하지 않음
좌석 숫자 오류	표시된 행을 원본과 대조. 알 수 없는 값은 빈 칸으로 남기고 변경 이력 기록
숫자가 너무 큼	중복·전국 범위 포함·필터 조건 확인
JSON이 이전 내용	변환 오류로 기존 파일이 보존됐는지 확인

가상 데이터로 복원하려면 원본 `sample.csv`에 대해 `--sample`을 붙여 변환합니다. 실제 파일 원본과 검토 메모는 별도 보존하세요.

완료 기준은 **출처·범위·기준일이 있는 대시보드**, **원본과의 행 수 대조**, **사실과 한계를 구분한 메모**입니다. 다음 교재는 이 결과물을 공개 주소로 배포합니다. 공개 전에는 파일의 이용조건과 공개 범위를 확인하세요.

참고 문서

- 전국도서관표준데이터 · 공공데이터포털
- Python CSV 읽기

LESSON 04

바이브코딩 다음은 클라우드 배포

내 PC의 웹앱을 실제 HTTPS 주소로

완성물: 공개 주소 · 확인 기록 · 복원용 백업 · 준비: Node.js · Cloudflare 계정

완성 모습과 배포 범위

내 PC에서만 열리던 도서관 대시보드를 HTTPS 주소로 공개하고 스마트폰에서 확인합니다. 이 사이트에 게시한 완성 예제를 먼저 열어보세요. 교재의 별도 Cloudflare 계정 배포는 학습자가 직접 진행하는 단계이며, 예제 사이트의 주소가 학습자의 배포 결과는 아닙니다.

이번에는 HTML·CSS·JS·JSON으로만 구성한 **정적 웹앱**을 배포합니다. Python API 서버, 로그인, 데이터베이스, 실시간 수집기는 포함하지 않습니다. 처음부터 모든 운영 요소를 섞지 않고 "어떤 파일을 어디에 올리는지"부터 확인합니다.

준비물: 실습 키트, Node.js 지원 LTS 버전, npm, 본인의 Cloudflare 계정. 계정의 요금·한도는 배포 시 확인합니다. 별도 도메인 구매 없이 제공되는 주소로 실습할 수 있습니다.

1단계 · AI가 만든 것을 점검하기

AI가 코드를 만들어준 것과 실제 서비스가 완성된 것은 다릅니다. 먼저 `python3 server.py` (Windows는 `py server.py`)로 원본을 실행합니다. 전체 6곳·400석, 해솔시 3곳·200석을 확인하세요.

파일 역할은 다음과 같습니다.

파일	역할	공개 여부
dashboard/index.html	화면 구조	공개
dashboard/styles.css	디자인	공개
dashboard/app.js	필터·집계 동작	공개
dashboard/data/libraries.json	화면에 표시할 자료	공개
server.py	PC 연습용 서버	이번 배포에서 제외
wrangler.jsonc	배포 폴더 설정	배포 도구가 읽음

`dashboard` 안의 파일은 방문자가 볼 수 있습니다. 인증키·기관 내부 자료·개인정보가 들어 있지 않은지 확인합니다. 첫 배포는 가상 자료 그대로 진행하세요.

2단계 · AI에게 작은 변경 맡기기

폴더를 복사해 원본을 보존하고 `ai-request.md`를 읽습니다. 원하는 변경을 하나만 지정하면 오류의 원인을 찾기 쉽습니다. 예를 들어 도서관 이름 검색을 추가합니다.

```
기존 dashboard 파일에 도서관 이름 검색 입력창을 추가해줘.
지역 선택과 검색어가 함께 적용되어야 해.
해솔시 + 어린이 → 1곳·60석,
검색어 지우기 → 3곳·200석이 완료 기준이야.
외부 라이브러리나 API 키 없이 구현해줘.
바꾼 파일, 확인 방법, 되돌리는 방법을 알려줘.
```

코드를 적용한 뒤 완료 기준을 직접 확인합니다. 실패하면 오류 메시지와 해당 코드만 전달해 수정하고, 동작이 확인되기 전에는 배포하지 않습니다. 이 변경은 선택 실습이며 키트 원본에는 이름 검색이 들어 있지 않습니다.

3단계 · 배포 도구와 폴더 확인하기

터미널에서 `node --version`, `npm --version`으로 설치를 확인합니다. `tech-practice` 폴더에서 제공된 `wrangler.jsonc`를 열면 다음 구조입니다.

```
{
  "name": "my-library-dashboard",
  "compatibility_date": "2026-09-20",
  "assets": { "directory": "./dashboard" }
}
```

`name`은 본인 계정 안에서 구별할 프로젝트 이름입니다. 영문 소문자·숫자·하이픈으로 수정할 수 있습니다.
`assets.directory`가 `./dashboard`인지 확인하세요. 프로젝트 전체를 올리는 설정으로 바꾸지 않습니다.

이 교재는 Workers의 정적 자산 기능을 사용합니다. 복잡한 프레임워크 빌드 없이 준비된 파일을 올리는 경로입니다. 기존 Next.js 앱 전체를 같은 설정으로 옮길 수 있다는 뜻은 아닙니다.

4단계 · 로그인하고 미리보기

```
npx wrangler@4 login
npx wrangler@4 dev
```

처음 실행할 때 패키지 설치 확인이 나올 수 있습니다. 로그인 명령은 브라우저에서 Cloudflare 계정 승인을 진행합니다. 여러 계정이 있으면 배포 대상 계정을 확인합니다. 기관 공용 계정이라면 계정 운영 규칙을 따르세요.

`dev`가 출력하는 로컬 주소를 열어 원본과 같은 화면이 나오는지 확인합니다. Python 서버의 8765와 다른 주소일 수 있습니다. 터미널에 나온 주소를 그대로 사용하세요. 검사가 끝나면 Ctrl+C로 미리보기를 종료합니다.

성공 확인: 전체·지역 필터가 동작하고 개발자 도구에 JSON 파일 404 오류가 없습니다. Wrangler는 외부 도구이므로 설치·로그인·실행 결과는 PC와 계정 환경에서 확인해야 합니다.

5단계 · 실제 주소로 배포하고 확인하기

```
npx wrangler@4 deploy
```

완료 출력의 HTTPS 주소를 복사해 새 브라우저 창에서 엽니다. 주소를 임의로 조합하지 말고 도구가 표시한 실제 주소를 사용합니다. 이어서 스마트폰에서도 열어봅니다.

- 첫 화면에 연습 자료 표시와 출처가 있는가?
- 전체 6곳·400석, 해솔시 3곳·200석이 맞는가?
- 작은 화면에서 지역 선택과 표를 사용할 수 있는가?
- 새로고침해도 JSON 파일을 정상적으로 읽는가?

이 구조는 배포한 JSON의 스냅샷입니다. PC의 CSV를 바꿔도 공개 사이트가 자동으로 갱신되지 않습니다. 다시 변환하고 검증한 뒤 배포해야 바뀝니다. 자동 갱신은 이후 API·자동화 실습에서 확장할 별도 기능입니다.

6단계 · 업데이트와 되돌리기

변경 전 `dashboard` 폴더를 날짜가 포함된 이름으로 복사해 보관합니다. CSV를 수정·변환하고 로컬에서 확인한 뒤 동일한 `deploy` 명령으로 올립니다. 작업 기록에는 변경 내용·배포 날짜·검증 결과를 남깁니다.

문제가 생기면 정상 작동하던 백업으로 `dashboard`를 복원하고 다시 배포합니다. 파일 기반 실습에서 이해하기 쉬운 복원 방법입니다. Cloudflare의 배포 이력과 rollback 기능도 있지만 데이터베이스 변경까지 자동으로 되돌려주는 것은 아닙니다.

공개를 끝내려면 Cloudflare 대시보드에서 **이번에 만든 프로젝트 이름**을 확인한 후 해당 프로젝트의 삭제 절차를 따릅니다. 연결한 사용자 도메인이 있다면 연결 상태도 함께 확인하세요. 다른 서비스 프로젝트를 선택하지 않도록 주의합니다.

막혔을 때

증상	확인 순서
npm 명령 없음	Node.js 설치 → 터미널 다시 열기
로그인 또는 권한 오류	브라우저 계정 → 선택한 계정 → Workers 권한
assets 폴더 없음	현재 위치와 wrangler.jsonc의 상대 경로 확인

증상	확인 순서
화면은 뜨는데 표가 비어 있음	data/libraries.json 요청 상태 → JSON 구조 → 브라우저 오류
API 주소만 404	파일 기반 원본인지 확인. Python 서버는 배포되지 않음
공개 주소에 이전 자료가 보임	로컬 JSON → 배포 성공 → 새로고침 순서로 확인

완료 기준과 다음 과정

실제 HTTPS 주소와 스마트폰 확인 결과, 수정 이력, 복원용 백업이 있으면 완료입니다. 외부 계정 배포를 진행하지 않았다면 "로컬 완료"로 기록하고 공개 완료로 표시하지 않습니다.

다음 n8n 교재에서는 반복 집계를 흐름으로 만들고, 사람이 검토할 결과를 생성합니다. 더 깊은 서버·컨텍스트·에이전트 학습은 엔지니어링 심화, 더 많은 코딩 사례는 바이브코딩으로 이어집니다.

공식 문서

- Cloudflare Workers 정적 자산
- 정적 사이트 시작하기
- Wrangler 설정
- Wrangler 명령과 rollback
- Node.js 다운로드

공식 문서 확인일: 2026-09-20. 명령·계정 화면은 버전에 따라 달라질 수 있습니다.

LESSON 05

n8n 첫 업무 자동화

입력·처리·결과를 연결하는 반복 업무

완성물: 수동 실행 가능한 도서관 집계 워크플로 · 준비: n8n 작업 공간

완성 모습과 준비물

버튼을 누르면 가상 도서관 6개 기록을 읽고, 해솔시 3곳·200석을 한 개 결과로 만드는 자동화입니다. 외부 메일 전송과 예약 실행을 넣기 전에 **입력 → 처리 → 결과**가 맞는지 확인합니다.

실습 키트의 `n8n-library-summary.json`을 사용합니다. 워크플로 파일만 다운로드할 수도 있습니다. 준비물은 접속 가능한 본인의 n8n 작업 공간입니다. n8n Cloud 또는 기관에서 운영하는 환경을 사용할 수 있으며, 요금·권한은 해당 환경에서 확인합니다.

워크플로 JSON 구조와 내부 집계 코드를 점검한 예제입니다. 학습자의 n8n 화면에서 가져오기·실행하는 과정은 직접 확인해야 합니다. 이 교재에서 n8n 계정 생성이나 실제 예약 작업을 대신 실행하지는 않습니다.

1단계 · 흐름을 읽기

노드	역할	예상 출력
직접 실행	사람이 테스트 시작	다음 노드 실행
연습 데이터	가상 도서관 자료 생성	6 items
해솔시 집계	지역 필터와 좌석 합계	1 item: count 3, seats 200

노드는 작업 한 개, 연결선은 다음 작업으로 넘기는 순서입니다. n8n에서 item은 하나의 데이터 묶음입니다. 마지막 출력이 1 item이어도 도서관이 한 곳이라는 뜻은 아닙니다. 결과 객체의 `count`가 도서관 수입니다.

이 예제는 모두 결정된 규칙으로 계산하며 AI 모델을 호출하지 않습니다. 숫자를 더하는 일은 코드로 처리하고, 필요하면 확인된 결과의 문장화에 AI를 추가하는 편이 검증하기 쉽습니다.

2단계 · 워크플로 가져오기

1. n8n에서 새 워크플로를 엽니다.
2. 워크플로 메뉴의 **Import from File**을 선택합니다. 버전에 따라 메뉴 위치와 표시 언어가 다를 수 있습니다.

3. `n8n-library-summary.json`을 선택합니다.
4. 세 노드가 왼쪽에서 오른쪽으로 연결됐는지 확인합니다.
5. "나의 도서관 집계 연습"처럼 이름을 바꾸고 저장합니다.

이 파일에는 인증정보·Webhook·메일 발송 노드가 없습니다. 가져오기 후 자동으로 예약 실행되는 구성도 아닙니다. 알 수 없는 노드 오류가 나면 n8n의 Code 노드를 지원하는 버전인지 확인하세요.

3단계 · 처음 실행하고 숫자 확인하기

Execute Workflow 또는 화면의 실행 버튼을 누릅니다. 실행이 끝나면 "연습 데이터" 노드를 선택하고 Output의 Table 또는 JSON 탭을 봅니다. 6개 항목에 name·city·seats·date가 있어야 합니다.

"해솔시 집계" 노드의 출력은 다음 핵심 값을 가집니다.

```
{
  "sample": true,
  "city": "해솔시",
  "count": 3,
  "seats": 200,
  "summary": "교육용 가상 데이터 · 검토 후 보고서에 반영"
}
```

성공 확인: count 3, seats 200입니다. 노드에 초록 표시가 뜨는 것만으로 숫자가 맞다는 뜻은 아니므로 원본 CSV와 대조합니다. 같은 흐름을 다시 실행해도 같은 결과가 나오는지 확인하세요.

4단계 · 집계 코드를 한 줄씩 읽기

"해솔시 집계" Code 노드는 JavaScript의 **Run Once for All Items** 모드입니다. 모든 입력을 한 번에 받아 합계를 계산합니다.

```
const rows = $input.all()
  .map(item => item.json)
  .filter(row => row.city === '해솔시');

return [{ json: {
  sample: true,
  city: '해솔시',
  count: rows.length,
  seats: rows.reduce((sum, row) =>
    sum + (Number.isFinite(row.seats) ? row.seats : 0), 0)
} }];
```

`$input.all()`은 입력 전체, `map`은 각 항목의 내용 추출, `filter`는 지역 선택, `reduce`는 합산입니다. 문자열 `"120"`은 숫자 120과 다르므로 이 예제에서는 숫자인 좌석만 합산합니다. 실제 자료를 연결할 때는 앞 단계에서 형식을 검증해야 합니다.

사본 워크플로에서 필터와 출력 city의 `해솔시`를 모두 `들꽃군`으로 바꾸고 실행하세요. 결과는 2곳·110석입니다. 필터만 바꾸고 표기 지역을 그대로 두면 내용과 제목이 어긋납니다.

5단계 · 실제 조회를 붙이는 설계

다음 단계는 "연습 데이터"를 **HTTP Request** 노드로 바꾸는 것입니다. 곧바로 운영 API를 붙이기 전에 아래 공개 가상 JSON으로 응답 구조를 연습할 수 있습니다.

```
https://aiedu.gdiaxhub.com/tech-demo/data/libraries.json
```

새 워크플로 사본에서 직접 실행 → HTTP Request → Code로 연결합니다. HTTP Request는 GET, URL은 위 주소, 응답 형식은 JSON으로 설정합니다. 출력이 하나의 객체이고 그 안의 `items`가 도서관 배열인지 먼저 확인하세요. 노드 설정에 따라 응답 본문이 감싸져 나오면 실제 출력 구조에 맞춰야 합니다.

응답 본문이 그대로 출력되는 설정이라면 Code 앞부분은 다음과 같이 바뀝니다.

```
const body = $input.first().json;
if (!Array.isArray(body.items)) {
  throw new Error('items 배열이 없습니다. HTTP 응답을 확인하세요.');
}
const rows = body.items.filter(row => row.city === '해솔시');
return [{ json: {
  sample: body.sample,
  city: '해솔시',
  count: rows.length,
  seats: rows.reduce((sum, row) =>
    sum + (Number.isFinite(row.seats) ? row.seats : 0), 0)
} }];
```

n8n Cloud에서 `127.0.0.1:8765`를 호출하면 학습자의 PC로 연결되지 않습니다. 외부에서 접속 가능한 주소가 필요합니다. 실제 공공 API 연결 시에는 인증정보를 Credentials로 관리하고, 조회 한도·페이지 수·오류 응답을 검사하는 단계를 추가합니다.

6단계 · 예약과 발송은 검증 뒤에 붙이기

수동 실행을 통과한 사본에서 Schedule Trigger를 추가합니다. 워크플로의 시간대를 `Asia/Seoul`로 확인하고, 처음에는 하루 한 번처럼 실행 빈도를 낮게 잡습니다. 예약 실행은 버전에 맞는 게시·활성화 절차를 완료해야 동작합니다. 저장만으로 실행된다고 가정하지 마세요.

예약 전에 결정할 항목은 다음과 같습니다.

- 실패하면 어디에서 확인할 것인가? 실행 이력과 Error Workflow를 연결할 담당자를 정합니다.
- 같은 날짜에 두 번 돌면 어떻게 할 것인가? 날짜·자료 버전 같은 식별자로 중복 결과를 구별합니다.
- 자료가 비어 있으면 0으로 보고할 것인가? 수집 실패와 정상 0건을 구분하고 실패 시 보고를 멈춥니다.
- 누구에게 전송할 것인가? 처음에는 결과 생성까지만 확인하고 검토한 뒤 발송을 붙입니다.

이 키트는 스케줄러·중복 방지 저장소·메일 발송을 구현한 운영 시스템이 아닙니다. 이 항목들을 직접 추가하고 검증한 뒤 반복 실행으로 전환하세요.

오류 해결과 되돌리기

증상	확인
가져오기 오류	ZIP 전체가 아니라 .json 파일을 선택했는지, 파일이 잘리지 않았는지
1 item만 보임	요약 결과 하나인지 확인. 내부 count 값을 봄
0곳으로 집계	지역명 철자와 입력 데이터의 city 값 비교
좌석 합계 0	seats 값이 숫자인지 문자열인지 확인
이전 실행 값이 계속 보임	고정 데이터(pin data) 여부와 새 실행 결과 확인
HTTP 오류	URL·응답 상태·JSON 구조·인증정보 확인

처음으로 돌아가려면 원본 JSON을 **새 워크플로**로 다시 가져옵니다. 직접 추가한 예약 작업이 있다면 기존 워크플로의 예약 활성화를 먼저 해제합니다. 복사본을 만들어도 기존 예약이 자동으로 꺼지지 않습니다.

완료 기준과 더 배우기

- 원본 6 items에서 해솔시 3곳·200석을 만들었다.
- 지역을 바꿔 들꽃군 2곳·110석을 확인했다.
- 입력 데이터와 요약 결과의 item 수 차이를 설명했다.
- 예약·외부 발송 전 검증할 내용을 적었다.

다섯 교재를 마치면 문서·데이터·API·배포·자동화가 하나의 작업으로 연결됩니다. 이후에는 실제 API 수집기, 서버의 비밀키 관리, 변경 이력, 오류 알림을 순서대로 확장할 수 있습니다. AI 비서 운영은 헤르메스 교재, 도구 연결과 에이전트 설계는 엔지니어링 심화에서 이어가세요.

공식 문서

- 워크플로 가져오기와 내보내기

- Code 노드와 실행 모드
- HTTP Request 노드
- Schedule Trigger와 시간대
- 오류 처리

공식 문서 확인일: 2026-09-20. n8n 계정 화면에서의 가져오기와 예약 실행은 사용 환경에서 별도 확인합니다.