Week 9. 파일 입출력(File I/O) + LLM 연습

데이터를 “저장하고 다시 불러오는” 프로그램 만들기

학습 목표

- 텍스트 파일을 열고 읽기/쓰기를 할 수 있다.
- with 문으로 파일을 안전하게 다룰 수 있다.
- CSV/JSON 기본 포맷을 읽고 저장할 수 있다.
- LLM을 활용해 파일 처리 코드를 생성하고 예외/경계값을 검증할 수 있다.

오늘의 구성

- 파일/경로 기본
- open 모드와 with 문
- 읽기(read) / 쓰기(write)
- CSV/JSON 기초
- With AI 실습 + 연습문제

왜 파일 입출력이 필요한가?

- 프로그램을 껐다 켜도 데이터가 남아야 한다(영속성)
- 로그/결과 보고서/설정 파일 저장
- 데이터 분석(다음 주) 전 단계: CSV 읽기

경로(path) 기초

- 상대경로: 현재 폴더 기준(예: data/input.txt)
- 절대경로: 드라이브/루트부터 전체 경로
- 실습에서는 프로젝트 폴더 구조를 먼저 정한다.

파일 열기(open) 모드

```
# r: 읽기(기본)
# w: 쓰기(덮어쓰기)
# a: 이어쓰기(append)
# x: 새 파일 생성(있으면 에러)
# b: 바이너리(이번 수업은 text 위주)
```

핵심 포인트

- w는 기존 파일을 지움(주의)
- 문자 인코딩(utf-8) 지정 권장
- 예외 처리(FileNotFoundError)

with 문(권장)

```
1 path = "data.txt"
2 with open(path, "r", encoding="utf-8") as f:
3     text = f.read()
4 print(text)
```

핵심 포인트

- 파일을 자동으로 닫아준다
- 예외가 나도 자원 정리
- 파일 누수 방지

읽기: read / readline / readlines

```
with open("data.txt", "r", encoding="utf-8") as f:
    all_text = f.read()
    # line = f.readline()
    # lines = f.readlines()
```

핵심 포인트

- open() as f: 파일을 객체 f로 열기
- read(): 전체를 한 번에
- readline(): 한 줄
- readlines(): 줄 리스트

쓰기: write / writelines

```
 1 lines = ["Hello
", "World
"]
 2 with open("out.txt", "w", encoding="utf-8") as f:
 3     f.writelines(lines)
```

핵심 포인트

- write는 문자열 1개
- writelines는 여러 줄
-

(줄바꿈)을 직접 넣어야 함

인코딩(encoding) 이슈

- 한글이 깨지면 encoding="utf-8"을 먼저 확인
- Windows 메모장 기본 인코딩 차이 가능
- 텍스트 파일 저장 시 UTF-8로 저장 권장

예외 처리: FileNotFoundError

```
try:
    with open("missing.txt", "r",
              encoding="utf-8") as f:
        print(f.read())
except FileNotFoundError:
    print("파일이 없습니다")
```

핵심 포인트

- 현실의 데이터는 "항상 존재"하지 않는다
- 예외 처리로 프로그램이 죽지 않게
- 에러 메시지를 사용자 친화적으로

CSV란?

- Comma-Separated Values
- 표 형태(행/열)를 텍스트로 저장
- 다음 주 pandas에서 본격적으로 사용

csv 모듈로 읽기(기초)

```python
import csv
with open("data.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

핵심 포인트

- row는 리스트 형태
- 헤더(첫 줄) 처리 규칙 정하기
- 구분자(delimiter) 변경 가능

csv 모듈로 쓰기(기초)

```
1 import csv
2 rows = [["name", "score"], ["kim", 90]]
3 with open("out.csv", "w", newline="",
            encoding="utf-8") as f:
4     w = csv.writer(f)
5     w.writerows(rows)
```

핵심 포인트

- newline="" 권장(빈 줄 방지)
- 데이터 저장/교환에 유용
- 엑셀에서도 열 수 있음

JSON이란?

- 딕셔너리/리스트 구조를 텍스트로 저장하는 포맷
- 설정(config), API 응답, 데이터 저장에 매우 흔함

json 저장/불러오기

```python
import json
data = {"name": "Alex", "scores": [80, 90]}
with open("data.json", "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=2)
with open("data.json", "r", encoding="utf-8") as f:
    obj = json.load(f)
print(obj)
```

핵심 포인트

- 파이썬 객체를 JSON으로 변환
- ensure_ascii=False로 한글 유지
- indent로 보기 좋게 저장
- dict/list와 자연스럽게 연결

With AI 실습 1: 파일 처리 요구사항 명확히

- 프롬프트에 반드시 포함: 파일명/형식/예시 데이터
- 출력 형식(줄바꿈, 구분자)을 구체적으로 지정
- 예외 상황(파일 없음, 빈 파일) 처리도 요구

With AI 실습 2: 테스트케이스 생성

- LLM에게 “정상/빈 파일/깨진 형식/인코딩” 케이스를 요구
- 테스트 파일을 직접 만들어 실행해본다.
- 결과가 다르면: 요구사항을 다시 조정하거나 코드 수정

With AI 실습 3: 리팩터링(함수화)

```
1 def read_numbers(path):
2     with open(path, "r", encoding="utf-8") as f:
3         return [int(x) for x in f.read().split()]
4
5 nums = read_numbers("nums.txt")
6 print(sum(nums))
```

AI에게 요청

- 예외 처리 추가(파일 없음/빈 파일)
- 주석/Docstring 추가
- 테스트 코드 예시

연습문제 1: 파일에서 숫자 읽고 통계

- nums.txt에 숫자가 공백/줄바꿈으로 저장되어 있다고 가정
- 읽어서 리스트로 만든 뒤 합/평균/최대/최소 출력
- 빈 파일이면 “데이터 없음” 출력

연습문제 2: 결과 보고서(out.txt) 생성

- 입력 파일을 읽고 처리한 결과를 out.txt로 저장
- 형식: 제목 1줄 + 결과 여러 줄
- 요구: 덮어쓰기(w) vs 이어쓰기(a) 비교

연습문제 3: CSV 성적표

- grade.csv를 읽어 학생 평균을 계산
- 출력: 이름, 평균(소수 1자리)
- 심화: 평균이 60 미만인 학생 목록 출력

연습문제 4: JSON 설정 파일

- 설정: {"sound": true, "level": 2} 같은 구조
- 프로그램 시작 시 json 로드 → 설정 적용
- 종료 시 변경된 설정을 다시 저장

미니 퀴즈(5문항)

문항
1) with 문을 쓰는 이유는?
2) w 모드의 특징은?
3) readlines()의 반환 형태는?
4) CSV는 어떤 데이터를 저장하기 좋은가?
5) json.load와 json.dump의 차이는?

정답: 1-자동닫기, 2-덮어쓰기, 3-리스트, 4-표형태, 5-읽기/쓰기

요약 & 다음 주 예고

- 파일 I/O는 데이터 영속성과 분석의 출발점이다.
- with + encoding="utf-8" + 예외 처리가 기본 세트
- CSV/JSON을 다루면 실무형 프로그램에 가까워진다.
- 다음 주: NumPy/Pandas로 CSV 분석