AI활용프로그래밍

Week 10. NumPy, Pandas 활용 + LLM 사용 연습

CSV 데이터를 불러와 배열·데이터프레임으로 기초 분석하기

학습 목표

- pip로 NumPy/Pandas를 설치하고 환경을 점검할 수 있다.
- NumPy 배열(ndarray)의 생성/인덱싱/기초 연산을 수행할 수 있다.
- Pandas Series/DataFrame 구조를 이해하고 데이터를 다룰 수 있다.
- CSV를 불러와 행/열 선택과 기초 통계를 계산할 수 있다.
- LLM을 활용해 분석 코드 작성·해석·디버깅을 수행할 수 있다.

오늘의 구성

- 데이터 분석 워크플로: 불러오기 → 확인 → 계산 → 결과 정리
- NumPy 핵심: 배열, 인덱싱, 벡터화 연산, 기초 통계
- Pandas 핵심: Series/DataFrame, read_csv, 선택/필터
- 실습: 간단한 CSV 분석 3개
- With AI 프롬프트 + 미니 퀴즈 + 과제

데이터 분석, 왜 필요할까?

- 의사결정: 숫자/기록을 근거로 판단하기
- 현업 데이터 예시: 설문, 매출, 센서, 로그, 출석/과제 기록
- 리스트/반복문만으로는 '표 형태 데이터' 처리에 한계가 있다
- NumPy/Pandas는 표준적인 데이터 처리 도구(속도+표현력)
- 다음 주(11주): 그래프로 결과를 시각화(Matplotlib)

NumPy 배열 vs 파이썬 리스트

```
1 nums = [1, 2, 3]
2 print(nums * 2)   # [1,2,3,1,2,3] (반복)
3
4 import numpy as np
5 arr = np.array([1, 2, 3])
6 print(arr * 2)    # [2 4 6] (원소별 연산)
```

핵심 차이

- list: 여러 타입 가능, 연산은 주로 반복문으로
- ndarray: 같은 타입 중심, 벡터화(원소별) 연산이 기본
- 대량 데이터 처리에서 NumPy가 빠르고 간결하다
- 주의: list의 * 는 '반복', array의 * 는 '원소별 곱'

설치·임포트(리마인드)

```
# (가상환경 활성화 후)
pip install numpy pandas

import numpy as np
import pandas as pd

print(np.__version__)
print(pd.__version__)
```

자주 생기는 문제

- ModuleNotFoundError → venv 활성화 여부 확인
- pip가 다른 파이썬에 설치될 수 있음(경로/환경 확인)
- 가능하면 프로젝트마다 venv 1개를 유지
- LLM에게 에러 메시지 + 실행한 명령어를 함께 전달

NumPy 배열 생성

```
1 import numpy as np
2 a = np.array([1, 2, 3])
3 b = np.arange(0, 10, 2)   # 0~8, step=2
4 c = np.zeros((2, 3))
5 d = np.ones((2, 3))
6 e = np.random.randint(1, 6, size=5)  # 주사위
```

핵심 포인트

- np.array: 리스트 → 배열
- arange: 규칙적인 수열 만들기
- zeros/ones: 초기화에 유용
- random: 샘플 데이터 만들기(실습용)
- shape(행,열)은 튜플로 전달한다

배열 속성(shape/dtype)

```python
import numpy as np
arr = np.arange(1, 7)       # [1 2 3 4 5 6]
print(arr.shape)            # (6,)
print(arr.dtype)            # int64 등

m = arr.reshape(2, 3)
print(m)
# [[1 2 3]
#  [4 5 6]]
```

핵심 포인트

- shape: 데이터 '모양'을 알려준다(1차원/2차원...)
- dtype: 원소의 자료형
- reshape: 같은 원소 개수(size)를 유지해야 한다
- 데이터 분석에서 'shape 확인'은 필수 습관

인덱싱·슬라이싱·마스크

```
1 import numpy as np
2 m = np.array([[10, 20, 30],
3               [40, 50, 60]])
4 print(m[0, 1])     # 20
5 print(m[:, 0])     # [10 40]
6
7 mask = m >= 40
8 print(m[mask])     # [40 50 60]
```

기억할 것

- m[row, col] 형태로 2D 접근
- : 는 전체 범위
- Boolean mask로 조건에 맞는 값만 추출 가능
- 실습에서 '조건 필터링'의 기초가 된다

벡터화 연산 & 브로드캐스팅

```
1 import numpy as np
2 a = np.array([1, 2, 3])
3 print(a + 10)      # [11 12 13]
4
5 x = np.ones((2, 3))
6 y = np.array([10, 20, 30])
7 print(x + y)       # 행마다 y가 적용
```

핵심 포인트

- 반복문 없이도 원소별 연산이 가능(간결+빠름)
- broadcasting: shape이 다르면 규칙에 따라 확장
- 디버깅 팁: shape을 출력해보고 연산하자
- LLM에게 'shape까지 설명'해달라고 요청하면 이해가 빨라진다

NumPy 기초 통계

```
1 import numpy as np
2 m = np.array([[1, 2, 3],
3               [4, 5, 6]])
4 print(m.sum())             # 21
5 print(m.mean())            # 3.5
6 print(m.max(axis=0))       # 열별 최대
7 print(m.max(axis=1))       # 행별 최대
```

포인트

- sum/mean/max/min/std 등 기본 통계 함수 제공
- axis=0: 열 기준 / axis=1: 행 기준
- CSV 데이터를 행/열로 생각하는 습관이 중요
- Pandas의 describe()도 결국 비슷한 통계를 준다

Pandas: Series vs DataFrame

Series (1차원)

- 값 + index(라벨)
- 한 열(column)처럼 생각
- 예: 학생 점수 목록

DataFrame (2차원)

- 행(row) + 열(column) 표 구조
- 엑셀/CSV와 가장 가까움
- 실습의 중심 객체

오늘은 'CSV → DataFrame → 행/열 선택 → 기초 통계' 흐름을 익힙니다.

CSV 읽기/쓰기

```python
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head())

# 저장
df.to_csv('out.csv', index=False)

# 한글 깨짐이 있으면
# pd.read_csv('data.csv', encoding='utf-8-sig')
```

핵심 포인트

- read_csv는 '표 데이터'를 DataFrame으로 만든다
- head()로 상위 5행 확인(습관)
- 저장 시 index=False로 불필요한 인덱스 열 방지
- encoding 문제는 흔하다(utf-8/utf-8-sig 등)

데이터 빠르게 살펴보기

```
df.shape          # (행, 열)
df.columns        # 열 이름
df.info()         # 타입/결측치
df.describe()     # 기초 통계
df.head(3)        # 상위 3행
```

순서(추천)

- 1) head()로 대충 생김새 확인
- 2) columns로 열 이름 확인
- 3) info()로 타입/결측치 확인
- 4) describe()로 통계 확인
- 이 순서를 LLM에게도 그대로 요구해볼 것

행/열 선택(Selection)

```
# 열 선택
df['score']
df[['name', 'score']]

# 행 선택
df.iloc[0]      # 0번째 행
df.iloc[0:3]    # 0~2행

# 조건 선택(맛보기)
df[df['score'] >= 80]
```

핵심 포인트

- 열 선택: 대괄호 [] 로 컬럼을 고른다
- iloc: '번호' 기반 / loc: '라벨' 기반(다음 주에 더)
- 조건 필터는 df[조건] 형태(중요)
- KeyError가 나면 열 이름 오타/공백을 의심

필터링·정렬(맛보기)

```python
# 필터링
high = df[df['score'] >= 80]

# 정렬
df.sort_values('score', ascending=False)

# 새 컬럼 만들기
df['pass'] = df['score'] >= 60
```

메모

- 필터링/정렬은 12주차에 더 깊게 다룬다
- 오늘은 '가능하다' 정도만 익히면 OK
- 새 컬럼 만들기는 실습에서 자주 쓴다
- SettingWithCopyWarning이 뜨면 복사본 여부 확인

실습 데이터(예시 CSV)

name	study_hours	score
A	1	55
B	2	60
C	3	72
D	4	78
E	5	88
F	6	95

실습 목표: 이 CSV를 읽고(score 평균/최대), 조건 필터링, 새 컬럼 생성까지 해봅니다.

실습 1: 불러오기 & 기초 통계

```python
import pandas as pd
df = pd.read_csv('study_score.csv')

# 1) 상위 5행 보기
print(df.head())

# 2) score 평균/최대
print(df['score'].mean())
print(df['score'].max())
```

완료 기준

- CSV를 정상적으로 읽고 head()가 출력된다
- score 평균(mean)과 최대(max)를 계산한다
- 열 이름이 다르면 columns로 확인 후 수정한다
- LLM에게 '내 파일 컬럼명'까지 알려주고 질문한다

실습 2: 새로운 컬럼 만들기

```
# pass 컬럼 만들기(60점 이상)
df['pass'] = df['score'] >= 60

# 효율(점수/시간) 컬럼
df['eff'] = df['score'] / df['study_hours']

print(df[['name','eff','pass']])
```

완료 기준

- pass, eff 컬럼이 생성된다
- dtype이 숫자가 아니면 변환(to_numeric)을 고려
- 0으로 나누는 상황이 없는지 확인
- 컬럼 추가 후 df.head()로 결과를 확인

실습 3: 질문 기반 분석(Q&A)

- Q1) 가장 효율(eff)이 높은 학생은 누구인가?
- Q2) score 80점 이상인 학생만 뽑아보자
- Q3) study_hours가 늘수록 score가 오르는 경향이 있는가? (수치로만 간단히)
- 결과를 3~5줄로 요약해서 README에 작성
- LLM에게: '결과 해석'을 요청하되, 실제 출력값을 붙여서 질문

With AI: 분석 코드 생성 프롬프트

```
[역할] 너는 파이썬 데이터분석 튜터야.
[목표] 아래 CSV를 읽고 통계를 계산하는 코드를 작성해줘.
[CSV 정보]
- 파일명: study_score.csv
- 컬럼: name, study_hours, score
[할 일]
- score 평균/최대
- score 80 이상 필터링
- eff=score/study_hours 컬럼 추가
[출력] 코드 + 각 줄 설명
```

팁

- 파일명/컬럼명을 반드시 알려주기(중요)
- 원하는 출력(예: 표 형태, 특정 열만)까지 명시
- 코드만 받지 말고 '설명'도 같이 요구
- 실행 후 나온 에러/출력을 다시 붙여서 재질문

With AI: 결과 해석 & 검증

```
df.describe() 결과를 붙여넣고
- 평균/최대/최소를 5줄로 요약해줘
- 이상치(너무 큰/작은 값) 의심되는 행을 알려줘
- 다음에 그려볼 그래프 2개를 추천해줘

제약: 추측하지 말고, 내가 준 출력만 기반으로 말해줘.
```

주의

- LLM은 '추측'하기 쉽다 → 출력값 기반으로 묻기
- '추측 금지' 제약을 넣으면 헛소리가 줄어든다
- 통계 해석은 정답이 하나가 아닐 수 있음
- 다음 주 그래프(11주)로 해석을 더 강화할 것

자주 발생하는 오류(데이터)

```
1 FileNotFoundError: 'study_score.csv' (경로 확인)
2 UnicodeDecodeError (encoding 지정)
3 KeyError: 'score' (열 이름 오타/공백)
4 TypeError: 문자열/숫자 연산 (dtype 확인)
5 SettingWithCopyWarning (복사본 수정 경고)
```

해결 루틴

1) 현재 폴더 확인: print(os.getcwd())
2) df.columns 출력해서 열 이름 정확히 보기
3) df.info()로 dtype 확인
4) 에러 메시지 전체 + 내 코드 10줄을 LLM에 전달
5) 해결 후 head()로 결과 확인

미니 퀴즈(5문항)

문항

1) list의 * 연산과 NumPy array의 * 연산은 어떻게 다른가?
2) shape이 (2,3)인 배열은 몇 행 몇 열인가?
3) DataFrame은 무엇을 표현하는 객체인가?
4) CSV를 읽는 Pandas 함수 이름은?
5) df[df['score']>=80] 는 무엇을 하는 코드인가?

정답 예시: 1) 반복 vs 원소별, 2) 2행 3열, 3) 2차원 표, 4) read_csv, 5) 점수 80이상 필터

과제 & 다음 주 예고

- 필수: CSV 1개를 선택(제공/직접제작)하여 DataFrame으로 분석
- 요구: (1) head/info/describe 중 2개 이상 (2) 평균/최대 등 통계 3개 이상
- 요구: 조건 필터 1개 이상 + 새 컬럼 1개 이상
- 제출: .py + README(결과 요약 5줄, AI 활용 내역)
- 다음 주(11주): Matplotlib로 그래프 2종 이상 그리기