AI활용프로그래밍

Week 6. 리스트 자료구조

list로 여러 데이터를 다루는 방법 + With AI 실습

학습 목표

- 리스트(list)의 특징과 사용 목적을 설명할 수 있다.
- 인덱싱/슬라이싱으로 데이터를 읽고 수정할 수 있다.
- 리스트 메서드(append, pop, sort 등)를 활용할 수 있다.
- LLM을 활용해 리스트 기반 문제를 설계·구현·개선할 수 있다.

오늘의 구성

1) 리스트 개념과 문법
2) 리스트 수정/삭제/정렬
3) 리스트 + 반복문 패턴
4) With AI 실습(미니 프로그램)
5) 퀴즈 & 과제

리스트(list)란?

- 여러 값을 “순서대로” 저장하는 자료구조
- 원소(Element)는 서로 다른 자료형도 가능
- 인덱스(index)는 0부터 시작
- 대부분의 “여러 개 입력/여러 개 결과”는 리스트로 해결

리스트 생성 & 인덱싱

```
nums = [10, 20, 30]
print(nums[0])  # 10
print(nums[-1]) # 30

nums[1] = 99
print(nums)     # [10, 99, 30]
```

핵심 포인트

- 음수 인덱스: 뒤에서부터(-1 마지막)
- 리스트는 "수정 가능(mutable)"
- 인덱스 범위를 벗어나면 IndexError

슬라이싱(Slicing) 기초

```
a = [0, 1, 2, 3, 4, 5]
print(a[1:4])   # [1,2,3]
print(a[:3])    # [0,1,2]
print(a[3:])    # [3,4,5]
print(a[::2])   # [0,2,4]
```

핵심 포인트

- 끝 인덱스는 포함되지 않음
- step(간격)도 지정 가능
- 슬라이싱 결과는 "새 리스트"

리스트 추가/확장

- append(x): 맨 뒤에 1개 추가
- extend(iterable): 여러 개를 한 번에 추가
- insert(i, x): i 위치에 삽입
- 주의: append([1,2])는 “리스트 1개”가 들어감

append vs extend

```
a = [1, 2]
a.append(3)
print(a)  # [1,2,3]

b = [1, 2]
b.extend([3, 4])
print(b)  # [1,2,3,4]
```

핵심 포인트

- extend는 반복 가능한(iterable)을 펼쳐서 추가
- append는 "그대로 1개"로 추가
- 헷갈리면 출력으로 즉시 확인

리스트 삭제

- pop(): 마지막 원소 꺼내기(반환) / pop(i): i번째 꺼내기
- remove(x): 값이 x인 첫 원소 삭제
- del a[i]: 인덱스로 삭제
- clear(): 전체 삭제

pop/remove 예시

```
a = ["a", "b", "c", "b"]
x = a.pop()     # "b"
print(x, a)

a.remove("b") # 첫 번째 "b" 삭제
print(a)
```

핵심 포인트

- remove는 값이 없으면 ValueError
- pop은 꺼낸 값을 변수에 저장 가능
- 삭제 로직은 버그가 나기 쉬움 → 테스트 중요

정렬(sort/sorted)

- sort(): 리스트 자체를 정렬(원본 변경)
- sorted(): 정렬된 “새 리스트” 반환(원본 유지)
- reverse=True로 내림차순
- key=로 정렬 기준 지정 가능(다음 주차에서 확장)

정렬 예시

```
nums = [3, 1, 10, 2]
nums.sort()
print(nums)  # [1,2,3,10]

names = ["Kim", "Lee", "Park"]
print(sorted(names))
```

핵심 포인트

- 원본 유지가 필요하면 sorted 사용
- 정렬 전/후를 print로 확인
- 정렬은 데이터 분석(Week10~)에서 필수

리스트 평균 구하기

```
scores = [80, 90, 75]
total = 0
for s in scores:
    total += s

avg = total / len(scores)
print("avg:", avg)
```

핵심 포인트

- len(list): 원소 개수
- 누적 변수는 0부터 시작
- 함수로 만들면 재사용↑ (Week5 복습)

참조와 복사(중요)

- b = a 를 하면 “같은 리스트”를 가리킴(참조)
- 원본을 유지하려면 복사 필요: a.copy() 또는 a[:]
- 버그 패턴: b를 바꿨는데 a도 같이 바뀜
- 리스트가 커질수록 이 문제가 치명적

복사 예시

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)  # [1,2,3,4]

c = a.copy()
c.append(5)
print(a)  # [1,2,3,4]
print(c)  # [1,2,3,4,5]
```

핵심 포인트

- b=a 는 복사가 아님!
- copy()/슬라이싱으로 독립 리스트 생성
- 2차원 리스트는 "얕은 복사" 주의(심화)

With AI: 리스트 문제 접근법

1) 데이터가 “여러 개”면 리스트를 먼저 떠올리기
2) 입력/출력 예시를 만들어 AI에게 공유
3) 메서드 사용을 제한하거나 지정하기(append/pop 등)
4) 완성 후: 예외 케이스(빈 리스트) 테스트하기

프롬프트 템플릿(리스트 프로그램)

```
역할: 파이썬 튜터
목표: TODO 리스트 프로그램을 만들어줘.
기능: add <할일>, list, done <번호>, quit
자료구조: list만 사용
요청: 코드 + 동작 예시 + 개선 아이디어 3개
```

프롬프트 포인트

- 기능 목록을 먼저 적기
- "자료구조 제한"으로 학습 범위 통제
- 동작 예시(샘플 실행)를 요구

실습 1: TODO 리스트(기본)

- 리스트에 할 일을 저장
- add: 입력받아 append
- list: 현재 목록 출력(번호 포함)
- done: 번호를 입력받아 pop/remove로 삭제

실습 2: 점수 분석기

- 점수 여러 개를 입력받아 리스트로 저장
- 출력: 평균/최대/최소/합계
- 도전: 90점 이상 개수 카운트
- LLM 활용: “테스트 입력”을 함께 만들기

실습 3: 문자열 → 리스트 처리

- 문제: "10 20 30" 형태로 입력받아 리스트로 변환
- 힌트: split() 사용
- 변환 후: 합계/평균 출력
- LLM 활용: 단계별(문자열→리스트→숫자변환) 설명 요청

AI 답변 검증 체크리스트

- 빈 리스트일 때도 동작하는가?
- 번호 입력(인덱스)이 범위를 벗어나면?
- 출력 형식(번호, 콤마 등)이 읽기 쉬운가?
- 리스트 복사/참조 문제는 없는가?
- 테스트 5개 이상 실행했는가?

미니 퀴즈(5문항)

- 1) append와 extend의 차이는?
- 2) pop()이 반환하는 것은?
- 3) sorted와 sort의 차이는?
- 4) b=a가 왜 위험한가?
- 5) 슬라이싱 a[1:4]에서 4는 포함되는가?

과제/연습문제

- 기본: TODO 리스트 프로그램 제출(기능 3개 이상)
- 도전: 중복 없는 리스트 만들기(이미 있으면 추가 X)
- 제출: .py 1개 + 실행 화면 캡처 1장
- AI 사용 시: 사용한 프롬프트 2개 + 개선한 부분 기록

요약 & 다음 주 예고

- 오늘: 리스트 생성/수정/삭제/정렬 + 반복 패턴
- 핵심: “여러 값”을 다루는 순간 리스트가 기본 선택
- 다음 주(Week 7): 튜플/딕셔너리/집합으로 확장
- 예고: 단어 빈도수 세기(딕셔너리) / 중복 제거(set)