AI활용프로그래밍

Week 4. 조건문·반복문

흐름 제어 + With AI 디버깅/연습문제

학습 목표

- 조건문(if/elif/else)로 프로그램 흐름을 제어할 수 있다.
- 반복문(for/while)로 반복 작업을 자동화할 수 있다.
- break/continue를 활용해 반복을 제어할 수 있다.
- LLM에게 테스트 케이스 기반으로 코드 수정·디버깅을 요청할 수 있다.

오늘의 구성

1) 조건문(if) 기초
2) 반복문(for/while) 기초
3) 반복문 패턴(누적/카운트/최대·최소)
4) With AI 디버깅 실습
5) 미니 퀴즈 & 과제

조건문이 필요한 이유

- 프로그램은 기본적으로 위에서 아래로 한 줄씩 실행
- 하지만 현실 문제는 “상황에 따라 다른 행동”이 필요
- if: 조건이 True일 때만 실행
- else/elif: 조건이 False일 때의 대안 경로

if / elif / else 기본

```
score = int(input("score: "))

if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"

print("grade:", grade)
```

핵심 포인트

- 들여쓰기(indent)가 블록을 결정
- 조건 순서가 중요(큰 범위→작은 범위)
- elif는 여러 번 가능

조건식 작성 팁

- 조건식은 bool이 되도록 만들기
- 복잡하면 중간 변수로 쪼개기
- 범위 체크: 0 <= x <= 100 (파이썬은 가능!)
- 주의: = (대입) vs == (비교)

중첩 if(필요할 때만)

```
age = int(input("age: "))
has_ticket = True

if age >= 18:
    if has_ticket:
        print("ENTER")
    else:
        print("NO TICKET")
else:
    print("TOO YOUNG")
```

핵심 포인트

- 중첩이 깊어지면 가독성이 떨어짐
- 가능하면 조건을 합치거나(AND) 함수로 분리(Week5)
- 출력은 "결론"만 간단히

반복문이 필요한 이유

- 같은 일을 여러 번 해야 할 때 코드 복사/붙여넣기는 위험
- 반복문은 "반복 규칙"만 쓰면 자동으로 반복 수행
- for: 정해진 횟수/시퀀스를 순회
- while: 조건이 True인 동안 반복(조건 기반)

for + range 기초

```
# 0부터 4까지 출력
for i in range(5):
    print(i)

# 1부터 5까지
for i in range(1, 6):
    print(i)
```

핵심 포인트

range(5) → 0,1,2,3,4
range(start, end)에서 end는 포함 X
반복 변수 i는 관례(원하면 의미 있는 이름)

while 기초

```
n = 3
while n > 0:
    print("count:", n)
    n = n - 1

print("done")
```

핵심 포인트

- while은 조건이 바뀌지 않으면 무한루프
- 루프 안에서 조건을 변화시키는 코드가 필요
- 중간 print로 흐름 확인

break / continue

- break: 반복문을 즉시 종료
- continue: 이번 반복만 건너뛰고 다음 반복으로
- 사용 팁: 너무 많이 쓰면 흐름이 복잡해짐
- 가능하면 조건문으로 먼저 걸러내기

조건+반복 조합: 메뉴 루프

```
while True:
    cmd = input("명령(add/list/quit): ")
    if cmd == "quit":
        break
    elif cmd == "list":
        print("TODO 출력")
    elif cmd == "add":
        print("TODO 추가")
    else:
        print("지원하지 않는 명령")
```

핵심 포인트

- while True + break는 "메뉴형 프로그램"의 기본
- elif로 명령 분기(조건문) 처리
- Week 6(리스트)와 결합하면 실제 앱처럼 동작

누적합(반복문 대표 패턴)

```
total = 0
for i in range(1, 6):
    total = total + i

print("sum(1..5):", total)
```

핵심 포인트

- 누적 변수(total)는 보통 0부터 시작
- "초기값 → 반복 갱신" 패턴 익히기
- 최대/최소/카운트도 동일한 구조

최댓값 찾기(간단)

```
nums = [3, 9, 2, 7]
max_v = nums[0]

for x in nums:
    if x > max_v:
        max_v = x

print("max:", max_v)
```

핵심 포인트

- 초기값을 첫 원소로 두는 방식
- if와 for 조합은 매우 중요
- Week 6(리스트)에서 더 많이 연습

흔한 버그 TOP 5

- 1) 들여쓰기 오류(IndentationError)
- 2) 조건 순서 실수(elif 순서)
- 3) range 끝값 포함 여부 착각
- 4) while 조건 갱신 누락(무한루프)
- 5) 비교 연산자 실수(==, >= 등)

With AI: 디버깅 프롬프트 핵심

- AI에게 “에러 메시지 + 코드 + 기대 출력”을 같이 주기
- 어떤 입력에서 문제가 발생하는지 재현 가능한 형태로 제공
- 수정 후: 테스트 케이스를 다시 실행해 검증
- 주의: AI가 고친 코드도 “내가 이해”해야 제출 가능

프롬프트 템플릿(디버깅)

```
역할: 파이썬 디버깅 도우미
문제: 아래 코드가 원하는 대로 동작하지 않음
기대: 입력 90 → A, 70 → C
실제: 입력 90 → C
코드: (아래 붙여넣기)
요청: 버그 원인 1개 + 수정 코드 + 테스트 3개
```

프롬프트 포인트

- "기대 vs 실제"를 적으면 원인 찾기 쉬움
- 테스트를 요구하면 회귀 버그를 줄임
- 수정 이유를 1~2문장으로 설명 요청

실습 1: 성적 등급 계산기

- 입력: 0~100 점수(score)
- 출력: A(90~), B(80~), C(70~), D(60~), F(그 외)
- 추가: 범위 밖 입력이면 “입력 오류” 출력
- LLM 활용: 테스트 케이스 5개를 먼저 만들고 코드 생성 요청

실습 2: 구구단 출력

- 입력: n(2~9)
- 출력: n×1부터 n×9까지 한 줄씩 출력
- 팁: f-string 사용
- LLM 활용: 출력 형식(예: 3 x 2 = 6) 지정

실습 3: 숫자 맞히기(while)

- 목표: 정답 숫자(예: 7)를 맞힐 때까지 반복 입력
- 힌트: 작으면 “UP”, 크면 “DOWN”
- 추가: 시도 횟수 카운트
- LLM 활용: “무한루프 방지 조건”을 함께 설계해달라고 요청

AI 답변 검증(테스트 중심)

- 테스트 케이스 최소 5개 만들기(정상/경계/이상)
- 경계값: 59/60/69/70/79/80/89/90
- 반복문은 "종료 조건"을 반드시 점검
- 출력 형식이 문제 요구와 정확히 일치하는지 확인

미니 퀴즈(5문항)

- 1) if 블록을 결정하는 것은?
- 2) range(1, 6)이 만드는 수는?
- 3) while에서 무한루프가 되는 가장 흔한 이유는?
- 4) break와 continue의 차이는?
- 5) 조건식에서 == 와 = 의 차이는?

과제/연습문제

- 기본: “성적 등급 계산기” 완성 + 테스트 10개
- 도전: 1~100 사이에서 소수(prime)만 출력하기
- 제출: .py 1개 + 실행 결과 캡처 1장
- AI 사용 시: 사용 프롬프트와 수정한 부분을 주석으로 기록

요약 & 다음 주 예고

- 오늘: if/for/while + 반복 패턴(누적/최대/카운트)
- 핵심: “조건(논리) + 반복(규칙)” 조합으로 문제 해결
- 다음 주(Week 5): 함수와 라이브러리로 코드를 재사용 가능하게
- 예고: 함수 설계 → AI로 개선/리팩터링