Python
CH03: 파이썬 시작하기 III

전종준 교수

서울시립대학교 통계학과/통계데이터사이언스대학원

Contents

Overview

Overview

조건문: 상황에 따라 다르게 행동하기

- 프로그램이 ”선택”할 수 있게 만듭니다.
- 예: 날씨에 따라 행동이 달라지는 스마트 알림 앱

반복문: 작업을 반복하기

- 비슷한 작업을 반복하고 싶을 때 사용해요.
- 예: 친구 3명에게 같은 메시지를 보낼 때

조건문 if: 상황에 따라 행동 바꾸기

- `if`, `else`, `elif`로 조건을 만듭니다.
- 어떤 조건이 참(True)일 때만 실행됩니다.

예시:

```
score = 85

if score >= 90:
    print("A등급입니다!")
elif score >= 80:
    print("B등급입니다!")
else:
    print("더 열심히 해봐요!")
```

반복문 for: 리스트의 요소를 하나씩 처리하기

- 리스트나 문자열 같은 것의 항목을 하나씩 꺼내서 작업합니다.

예시:

```
shopping_list = ["우유", "계란", "쌀"]

for item in shopping_list:
    print(item + "을(를) 장바구니에 담았어요.")
```

조건과 반복을 함께 쓰기

- 반복 중에서도 특정 조건일 때만 특별한 일을 할 수 있어요!

예시:

```
friends = ["민준", "철수", "지우", "서연"]

for name in friends:
    if name == "철수":
        print(name + "는 이미 메시지를 받았어요.")
    else:
        print(name + "에게 메시지 보냅니다.")
```

조건문

불리언(Boolean): 참(True)과 거짓(False)

- 파이썬에서 참/거짓을 표현하는 데이터 타입
- `True`, `False`는 첫 글자가 대문자

예시 코드:

```
1 is_hungry = True
2 is_raining = False
3
4 print(type(is_hungry))  # <class 'bool'>
```

비교 연산자: 값끼리 비교해보기

- 두 값이 같은지, 큰지, 작은지를 비교할 수 있음.
- 결과는 항상 `True` 또는 `False`입니다.

종류:

- ==, !=, >, <, >=, <=

예시:

```
1 print(3 == 3)   # True
2 print(5 != 2)   # True
3 print(10 < 1)   # False
```

논리 연산자: 여러 조건을 합성하기

- `and` 또는 &: 둘 다 참이면 참
- `or` 또는 |: 하나라도 참이면 참
- `not` : 참을 거짓으로, 거짓을 참으로

예시 코드:

```
age = 20
has_ticket = True

print(age >= 18 and has_ticket)  # True
print(age < 18 or has_ticket)    # True
print(not has_ticket)            # False
```

실생활과 연결하기: 입장 조건

- 영화관 입장 조건: 18세 이상이면서 표가 있어야 한다.

예시 코드:

```
age = 17
has_ticket = True

can_enter = age >= 18 and has_ticket
print("입장 가능?", can_enter)  # False
```

불과 조건이 만나면? → 조건문

- 불(True/False)은 컴퓨터에게 ”이 조건이 맞니?”라고 묻는 방법.
- 조건에 따라 실제로 다른 코드를 실행하게 만들수 있음

예시:

```
1 if age >= 18:
2     print("입장하세요!")
3 else:
4     print("입장 불가입니다.")
```

조건문이란?

- 특정 조건이 `True`일 때만 실행되는 코드 블록.
- 컴퓨터에게 "이럴 땐 이렇게 해!"라고 알려주는 방법.

형식:

```
if 조건:
    실행할_코드
```

if - else: 둘 중 하나 선택

- `if` 조건이 참이면 실행, 그렇지 않으면 `else`

예시 코드:

```
1 hungry = True
2
3 if hungry:
4     print("밥 먹자!")
5 else:
6     print("그럼 그냥 쉰다.")
```

elif: 여러 조건 중 하나 선택

- `if` → `elif` → `else` 순서로 검사합니다.
- 가장 먼저 참이 되는 조건만 실행됩니다.

예시 코드:

```
score = 75

if score >= 90:
    print("A학점")
elif score >= 80:
    print("B학점")
elif score >= 70:
    print("C학점")
else:
    print("재도전!")
```

들여쓰기(indent)

- 들여쓰기(보통 스페이스 4칸)는 조건문 안에 포함된 코드를 의미
- 들여쓰기를 잘못하면 오류 발생

틀린 예:

```
1 if True:
2 print("안녕하세요!")  # 들여쓰기 없음 → 오류!
```

예제: 영화관 입장 가능 여부

- 18세 이상이면서 표가 있어야 입장 가능!

코드 예시:

```
age = 20
has_ticket = True

if age >= 18 and has_ticket:
    print("입장하세요!")
else:
    print("입장할 수 없습니다.")
```

실습 문제: 조건 판단 로직 만들기

- 아래 조건을 코드로 만들어 보아라
- 나이에 따라 버스 요금 정하기
- 입력: 나이 (정수)

요금 규칙:

- 0~7세 → 무료
- 8~19세 → 청소년 요금
- 20세 이상 → 일반 요금

정답 예시: 버스 요금 계산

```
age = 15

if age <= 7:
    print("무료입니다.")
elif age <= 19:
    print("청소년 요금입니다.")
else:
    print("일반 요금입니다.")
```

조건문 요약 정리

- `if`, `elif`, `else`로 조건을 나눌 수 있음.
- 조건식은 항상 `True` 또는 `False`로 평가.
- **비교 연산자**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **논리 연산자**: `and`, `or`, `not`
- 들여쓰기(indent) 주의!

반복문

반복문이란?

- 같은 동작을 여러 번 반복하고 싶을 때 사용.
- 파이썬에서는 주로 `for`문을 사용.

예: 친구에게 메시지 3번 보내기

```
print("안녕 민준!")
print("안녕 민준!")
print("안녕 민준!")
```

반복문 사용:

```
for i in range(3):
    print("안녕 민준!")
```

for문 기본 구조

- 리스트, 문자열 등을 하나씩 꺼내서 작업할 수 있음.
- `for 변수 in 시퀀스:`
- 변수에 시퀀스 내 원소를 하나씩 저장한 후 아래 문장을 실행
- 더 이상 꺼내올 시퀀스 내 원소가 없는 경우 반복을 중지

예시:

```
1 shopping_list = ["우유", "계란", "쌀"]
2
3 for item in shopping_list:
4     print(item + "을(를) 장바구니에 담았어요.")
```

문자열도 반복할 수 있어요

- 문자열은 문자들의 리스트처럼 반복할 수 있음!

예시:

```
1 for ch in "파이썬":
2     print(ch)
```

range(): 숫자 반복을 도와주는 함수

- `range(n)`은 0부터 n-1까지의 숫자를 만들어줌.

예시:

```
1 for i in range(5):
2     print(i)
```

반복 + 조건: 특정 조건에만 반응하기

- 반복하면서 조건을 검사할 수도 있어요!

예시: 짝수만 출력하기

```
1 for num in range(1, 6):
2     if num % 2 == 0:
3         print(num)
```

(참고) % 기호는 나머지를 계산해줌.

실생활 예시: 친구에게 생일 메시지 보내기

- 친구 리스트를 반복하면서 같은 메시지를 보낼 수 있음.

예시 코드:

```
friends = ["민준", "서연", "지우"]

for name in friends:
    print(name + "아, 생일 축하해!")
```

반복문 요약 정리

- `for 변수 in 리스트:` 구조로 반복
- 문자열, 리스트, range 등 다양한 대상 반복 가능
- 조건문과 함께 쓰면 강력한 제어 가능!

예시:

```
1 for i in range(3):
2     print("안녕 민준!")
```

제어

흐름 제어문이란?

- 반복 중간에 멈추거나 건너뛰고 싶을 때 사용
- 대표적인 키워드: `break`, `continue`, `pass`

break 문

- 반복문을 완전히 종료
- 조건에 따라 반복을 멈추고 싶을 때 사용

예시 코드:

```
i = 0
while True:
    if i == 3:
        break
    print(i)
    i += 1
```

continue 문

- 반복문의 현재 순서를 건너뛰고 다음 순서로 넘어가요.
- 특정 조건에서만 출력하고 싶을 때 사용해요.

예시 코드:

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

pass 문

- 아무 동작도 하지 않아요. 문법적으로 채워야 할 자리에 사용해요.

예시 코드:

```
1 for i in range(3):
2     if i == 1:
3         pass
4     print(i)
```

예제 1: 첫 3개의 짝수 출력

- 1부터 100 사이의 숫자 중 첫 3개의 짝수를 출력합니다.
- `continue`로 홀수를 건너뛰고, `break`로 3개 출력 후 종료합니다.

```
1  count = 0
2  for i in range(1, 101):
3      if i % 2 != 0:
4          continue
5      print(i)
6      count = count + 1
7      if count == 3:
8          break
```

예제 2: 비밀번호 맞을 때까지 입력 받기

- 비밀번호가 맞을 때까지 계속 묻습니다.
- 최대 5번까지 시도할 수 있습니다.

```
correct_pw = "python123"
for attempt in range(5):
    pw = input("비밀번호 입력: ")
    if pw == correct_pw:
        print("접속 성공")
        break
    else:
        print("틀렸습니다.")
else:
    print("접속 실패 - 시도 초과")
```

초급 넘어서기: while 문과 이중 loop

반복문: `while` 문

- `while` 문은 조건이 참인 동안 계속 반복함
- 반복 횟수가 정해져 있지 않을 때 자주 사용함
- 조건이 거짓이 되면 반복을 멈춤

기본 구조:

```
while 조건:
    실행할 코드
```

while 예제: 숫자 세기

- 1부터 5까지 숫자를 출력.

예시 코드:

```
n = 1
while n <= 5:
    print(n)
    n += 1
```

while 문 주의사항

- 조건이 계속 참이면 무한 반복이 발생할 수 있음
- 반복을 종료하려면 조건을 적절히 갱신하거나 break 문을 사용할 수 있음

무한 루프 예시:

```
1 while True:
2     command = input("종료하려면 'q' 입력: ")
3     if command == 'q':
4         break
```

이중 루프란?

- 반복문 안에 또 다른 반복문이 들어 있는 구조
- 바깥쪽 루프가 한 번 실행될 때마다 안쪽 루프는 전체를 반복
- 주로 표 형태 출력, 모든 쌍(pair) 처리 등에 사용

기본 구조:

```
for i in 바깥범위:
    for j in 안쪽범위:
        실행할 코드
```

예제: 구구단 출력

- 2단부터 4단까지 구구단을 출력

```
1  for dan in range(2, 5):        # 2단부터 4단까지
2      for i in range(1, 10):     # 1부터 9까지 곱함
3          print(dan*i)
4      print("----------")
```

예제: 별로 된 사각형 출력

- 이중 루프를 사용해 3행 5열의 별 사각형을 출력₩

```
for row in range(3):       # 3행
    for col in range(5):   # 5열
        print("*", end="") #줄바꿈 없이 출력
    print() # 줄바꿈
```

이중 루프에서 break 사용

- break는 현재 속한 반복문 하나만 종료
- 안쪽 루프에서 break를 사용하면 바깥 루프는 계속 진행

예제:

```
for x in range(3):              # 바깥 루프: 0, 1, 2
    for y in range(5):          # 안쪽 루프: 0 ~ 4
        if y == 2:
            break               # 안쪽 루프만 종료
        print(x, y)
```

이중 루프에서 continue 사용

- continue는 현재 속한 루프의 다음 반복으로 넘어감
- 안쪽 루프에서 continue는 안쪽 루프의 다음 반복으로 넘어감

예제:

```
for i in range(3):
    for j in range(5):
        if j == 2:
            continue        # j가 2일 땐 건너뜀
        print(i, j)
```

모든 루프 탈출하기

- 일반적인 `break`는 안쪽 루프만 종료
- 모든 루프를 완전히 종료하려면 변수로 제어하거나 함수+`return` 사용이 필요

예제: 플래그 변수 사용

```
found = False
for x in range(3):
    for y in range(5):
        if x + y == 5:
            found = True
            break
    if found:
        break
print("종료")
```