03장 제어문: 프로그램의 흐름을 제어한다

왜 제어문을 사용해야 할까?

- 프로그램은 기본적으로 위에서 아래로 순차적으로 실행된다
- 하지만, 실제 문제를 해결하려면 조건에 따라 다르게 동작하거나, 반복적으로 작업을 수행해야 할 때가 많다.
- 제어문을 사용하면 프로그램의 실행 흐름을 자유롭게 조절하여, 다양한 상황에 맞는 유연한 코드를 작성할 수 있음

제어문이란?

- 프로그램에서 명령문의 실행 순서(흐름)를 개발자가 원하는 대로 바꿀 수 있도록 해주는 구문입니다.
- 즉, 특정 조건에 따라 명령을 선택적으로 실행하거나, 반복적으로 실행할 수 있게 해줍니다.

종류

1. **조건문**: 특정 조건에 따라 명령을 선택적으로 실행 (예: `if`, `elif`, `else`)
2. **반복문**: 특정 명령을 여러 번 반복 실행 (예: `for`, `while`)
3. **분기문**: 반복 흐름을 중단하거나 건너뜀 (예: `break`, `continue`)

조건문을 위한 비교 연산자

연산자	의미	예시 (a=3, b=5)	결과
==	같다	a == b	False
!=	같지 않다	a != b	True
>	크다	a > b	False
<	작다	a < b	True
>=	크거나 같다	a >= b	False
<=	작거나 같다	a <= b	True

비교 연산자는 조건문에서 참/거짓을 판단할 때 사용합니다.

```
# 비교 연산자 예시
x = 3
y = 2
print(x == y)    # False
print(x == 3)    # True
print(x < 4)     # True
```

```
False
True
```

```
True
```

논리 연산자: and, or, not

- **and**: 두 조건이 모두 참일 때만 참
- **or**: 둘 중 하나라도 참이면 참
- **not**: 조건의 참/거짓을 반대로

복잡한 조건을 만들 때 사용합니다.

```
# 논리 연산자 예시
x = 3
y = 4
z = 3
print(x != 3 and x == z)   # False
print(x != 3 or x == z)    # True
print(not(x == 3))         # False
```

```
False
True
False
```

포함 연산자: in, not in

- **in**: 값이 시퀀스(리스트, 문자열 등)에 포함되어 있으면 참
- **not in**: 값이 포함되어 있지 않으면 참

포함 여부를 조건문에서 자주 사용합니다.

```
# in, not in 예시
print('a' in 'apple')         # True
print(3 in [1,2,4])           # False
print("-"*30)
print('a' not in 'apple')     # False
print(3 not in [1,2,4])       # True
```

```
True
False
------------------------------
False
True
```

조건문 (if문)

조건이 참(True)일 때만 특정 코드가 실행됩니다.
대표적으로 `if`, `elif`, `else` 구문이 있습니다.

```
# if, elif, else 예시
score = 85
if score >= 90:
    print("A학점")
elif score >= 80:
    print("B학점")
else:
    print("C학점")
```

```
B학점
```

- `if`: 첫 번째 조건을 검사
- `elif`: 위 조건이 거짓이면 다음 조건 검사
- `else`: 모든 조건이 거짓이면 실행

점수에 따라 "A학점", "B학점", "C학점" 중 하나가 출력됩니다.

반복문 (for, while문)

특정 명령을 여러 번 반복해서 실행할 때 사용합니다.

while문

조건이 참인 동안 반복 실행합니다.

```
while 조건식:
    반복할 코드
```

```
# 1부터 5까지 출력
i = 1
while i <= 5:
    print(i)
    i += 1
```

```
1
2
3
4
5
```

for문

시퀀스(리스트, 문자열 등)의 요소를 하나씩 꺼내며 반복합니다.

```
for 변수 in 시퀀스:
    반복할 코드
```

```
# 리스트의 모든 요소 출력
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

```
1
2
3
4
5
```

분기문 (break, continue)

반복문 안에서 반복의 흐름을 제어할 때 사용합니다.

break문

- 반복문을 즉시 종료하고 반복문 밖으로 빠져나갑니다.

break문

- 반복문을 즉시 종료하고 반복문 밖으로 빠져나갑니다.

```
# break 예시
for i in range(1, 10):
    if i == 5:
        break
```

```
    print(i)
# 1, 2, 3, 4 출력 (i가 5가 되면 반복 종료)
```

```
1
2
3
4
```

continue문

- 반복문의 나머지 부분을 건너뛰고, 다음 반복을 진행합니다.

```
# continue 예시
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
# 1, 2, 4, 5 출력 (i가 3일 때는 print를 건너뜀)
```

```
1
2
4
5
```