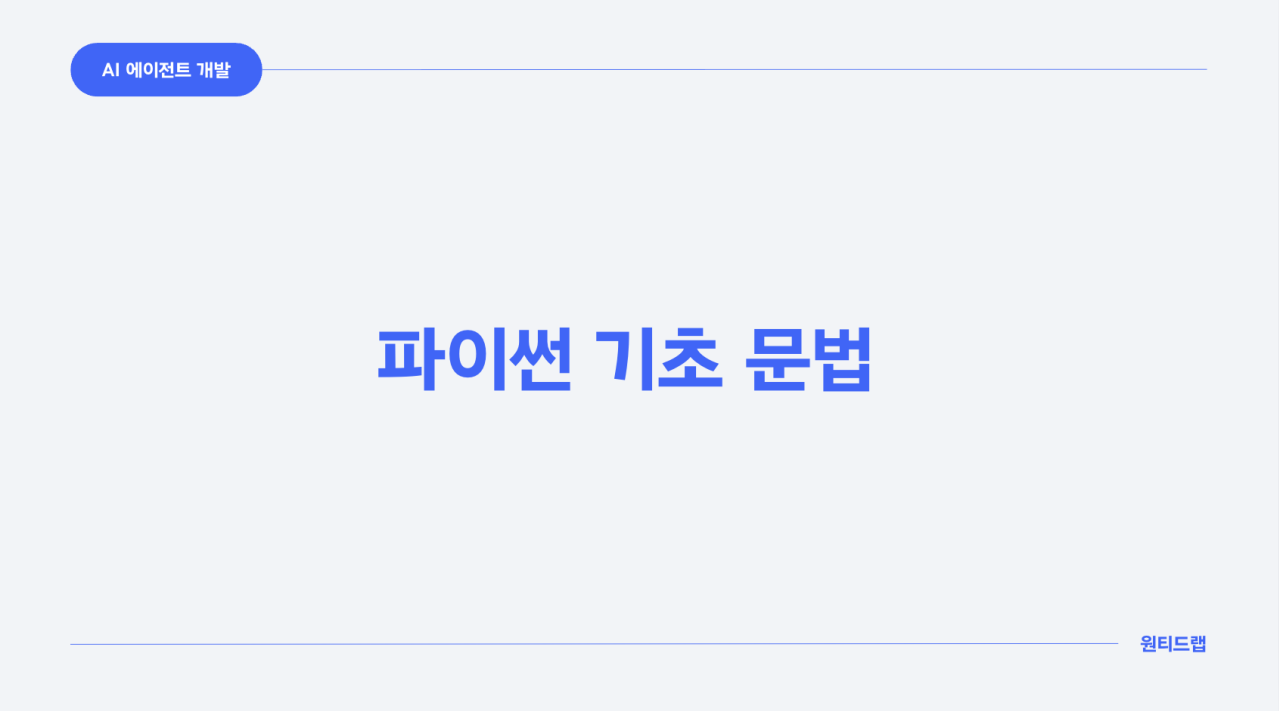
01 파이썬 기초 문법 II (제어문)



1. 조건문

1) 표현

- 조건식을 만족하는 경우와 그렇지 못한 경우를 구분하여 각각 다른 프로그래민 코드를 실행
- 콜론(:)과 들여쓰기로 구분

```
# CASE 1
if 조건식:
    조건식이 True인 경우
```

```
# CASE 2
if 조건식:
    조건식이 True인 경우
else:
    조건식이 False인 경우
```

```
# CASE 3
if 조건식1:
    조건식1이 True인 경우
elif 조건식2:
    조건식1이 False인 것 중
    조건식2가 True인 경우
else:
    그 외
```

2) 연산자

비교 연산자

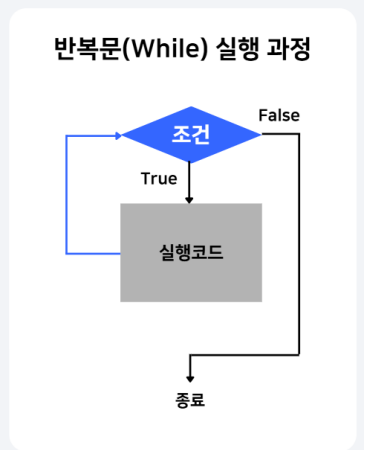
연산	의미	연산	의미
`a > b`	a가 b보다 크다	`a < b`	a가 b보다 작다
`a >= b`	a가 b보다 크거나 같다	`a <= b`	a가 b보다 작거나 같다
`a == b`	a와 b는 같다	`a != b`	a는 b와 같지 않다.

조건 연산자

연산	의미	연산	의미
`a or b`	a와 b 둘 중 하나만 참이어도 참	`a in 문자열/리스트/튜플`	a가 문자열/리스트/튜플 안에 있으면 참
`a and b`	a와 b 모두 참이어야 참	`a not in 문자열/리스트/튜플`	a가 문자열/리스트/튜플 안에 없으면 참
`not a`	a가 거짓이면 참	`a in 딕셔너리`	a가 딕셔너리 키에 있으면 참
		`a not in 딕셔너리`	a가 딕셔너리 키에 없으면

2. 반복문

1) While문



- 조건문이 True인 동안 `while` 문 안의 명령문이 반복 실행됨
- 특정 상황에서 반복을 중지하게 while문 안에서 조건을 제어함
- 콜론(:)과 들여쓰기로 구분

표현

```
count = 0
while count < 3:
    print(count)
    count += 1

# 실행 결과
# 0
# 1
# 2
```

```
# 작동 방식
count = 0 | 조건식 count < 3 True | print 실행 0 | count += 1 (count = 1)
count = 1 | 조건식 count < 3 True | print 실행 1 | count += 1 (count = 2)
count = 2 | 조건식 count < 3 True | print 실행 2 | count += 1 (count = 3)
count = 3 | 조건식 count < 3 False
```

⚠ **`while True:` 는 무한 반복**

- `while` 문은 조건이 만족하는 한 무한 반복 됨
- 주피터노트북에서 끝내는 방법은 '정지' 버튼 클릭

조건 제어

break

반복문 안에서 강제로 반복문을 빠져 나올 때 사용

```
count = 0
while count < 3:
    print(count)
    count += 1

    if count == 2:
        break

# 실행 결과
# 0
# 1
```

```
# 작동 방식
count = 0 | 조건식 count < 3 True | print 실행 0 | count += 1 (count = 1) | 조건식 count == 1 False
count = 1 | 조건식 count < 3 True | print 실행 1 | count += 1 (count = 2) | 조건식 count == 2 True
```

continue

반복문을 중단시키지 않고 다음 반복으로 넘어갈 때 사용

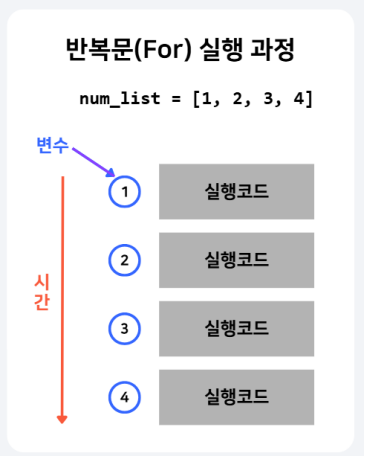
```
count = 0
while count < 5:
```

```
    count += 1
    if count % 2 == 0:
        continue
    print(count)

# 실행 결과
# 1
# 3
# 5
```

```
# 작동 방식
count = 0 | 조건식 count < 5 True | count += 1 (count = 1) | 조건식 count % 2 == 0 False | print 실행 1
count = 1 | 조건식 count < 5 True | count += 1 (count = 2) | 조건식 count % 2 == 0 True 반복문 다시 시작
count = 2 | 조건식 count < 5 True | count += 1 (count = 3) | 조건식 count % 2 == 0 False | print 실행 3
count = 3 | 조건식 count < 5 True | count += 1 (count = 4) | 조건식 count % 2 == 0 True 반복문 다시 시작
count = 4 | 조건식 count < 5 True | count += 1 (count = 5) | 조건식 count % 2 == 0 False | print 실행 5
count = 5 | 조건식 count < 5 False
```

2) For문



- **정해진 시퀀스** 길이만큼 반복 → **반복문 시작 전 시퀀스가 미리 변수로 정의되어야 함**
- 시퀀스: 문자열, 리스트, 튜플과 같이 데이터의 순서와 위치가 저장되는 데이터
- 콜론(:)과 들여쓰기로 구분

표현

```
num_list = ["a", "b", "c"]
for num in num_list:
    print(num)

# 실행 결과
# a
# b
# c
```

```
# 작동 방식
num_list의 길이가 3이기 때문에 3번 반복
반복 1: num_list의 0번째 위치의 값을 num 변수에 대입 | print 실행 "a"
반복 2: num_list의 1번째 위치의 값을 num 변수에 대입 | print 실행 "b"
반복 3: num_list의 2번째 위치의 값을 num 변수에 대입 | print 실행 "C"
```

주요 함수

`range()`

- 반복 횟수를 지정하고 싶을 때 사용
- `range(n)`은 정수 0부터 n-1까지의 범위
- `range(m, n)`은 정수 m부터 n-1까지의 범위
- `range(m, n, x)`은 정수 m부터 n-1까지 x 간격

```
for num in range(4):
    print(num)

# 실행 결과
# 0
# 1
# 2
# 3
```

```
# 작동 방식
range(4): 0부터 3까지의 범위 → 4번 반복
반복 1: num 변수에 0 대입 | print 실행 0
반복 2: num 변수에 1 대입 | print 실행 1
반복 3: num 변수에 2 대입 | print 실행 2
반복 4: num 변수에 3 대입 | print 실행 3
```

`enumerate()`

- index와 값을 함께 변수로 가져오고 싶을 때 사용
- 주로 특정 위치의 값을 제어하고 싶을 때 사용됨

```
fruits = ["apple", "banana", "melon"]
for idx, fruit in enumerate(fruits):
    print(idx, fruit)

# 실행 결과
# 0 apple
# 1 banana
# 2 melon
```

```
# 작동 방식
fruits의 길이가 3이기 때문에 3번 반복
반복 1: idx = 0, fruit = "apple"  | print 실행 0 apple
반복 2: idx = 1, fruit = "banana" | print 실행 1 banana
반복 3: idx = 2, fruit = "melon"  | print 실행 2 melon
```

3) List Comprehension

- 한줄 코드 작성
- 간단한 반복문의 경우에는 한 줄로 간단히 표현 가능

CASE 1 (조건문 if 포함)

```
nums = []
for num in range(1, 11):
    if num % 2 == 0:
        nums.append(num)
```

```
nums = [num for num in range(11) if num % 2 == 0]
```

CASE 2(조건문 if-else 포함)

```
nums = []
for num in range(1, 11):
    if num % 2 == 0:
        nums.append(num)
    else:
        nums.append("홀수")
```

```
nums = [num if num % 2 == 0 else "홀수" for num in range(11)]
```