PYTHON PROGRAMMING

LECTURE 4: CONDITION & LOOP

goorm

KAIST AI
Graduate School of AI

DAVIAN

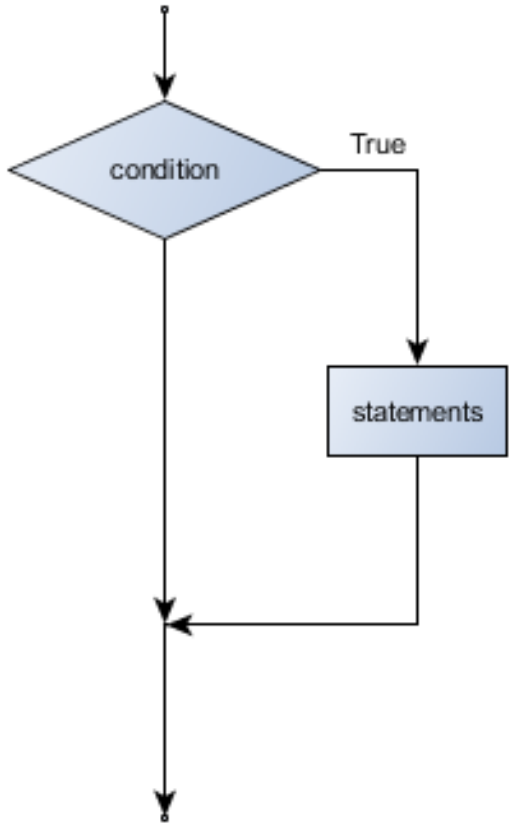
Conditional Statements

- 특정 조건이 만족될 경우 실행할 문항을 설정

```
명령 1
명령 2

if 〈조건〉:
        if-명령 1
        if-명령 2

명령 3
명령 4
```



- 들여쓰기와 :으로 구문을 구분
- 들여쓰기의 Convention은 스페이스 4칸
  - Tab 키를 눌러 삽입

Conditional Statements

- if [조건] – 조건을 검사하여 block을 실행
- elif [조건] – 이전 조건과 맞지 않을 경우 조건을 다시 검사 및 실행
- else – 이전 모든 조건이 맞지 않을 경우 실행

```
score = 80

if score > 60:
    print ("Over 60")

if score > 70:
    print ("Over 70")

if score > 80:
    print ("Over 80")
```

```
score = 80

if score > 80:
    print ("Grade A")

elif score > 70:
    print ("Grade B")

else:
    print ("Grade F")
```

Conditional Statements

```
if False:
    print("This sentence does not show")

if "":
    print("Empty is False")

if 0:
    print("0 is False")

if None:
    print("None is also False")
```

Conditional Statements

```
a, b = 5, 8

if a == 5 and b == 6:
    print ("This is False")

if not a < b < 9:
    print ("This is False")

if a + 3 == b:
    print ("This is True")
```

비교 연산자와
논리 연산자를
사용

Ternary operators

삼항 연산자

- [Value1] if [Condition] else [Value2]
- Condition이 참이면 Value1을 거짓이면 Value2를 반환
- **연산자이다**
  - 블록으로 구분되는 문법요소가 아님

```
>>> value = 32

>>> "odd" if value % 2 else "even"
'even'

>>> ("odd" if value % 2 else "even") + "_number"
'even_number'
```

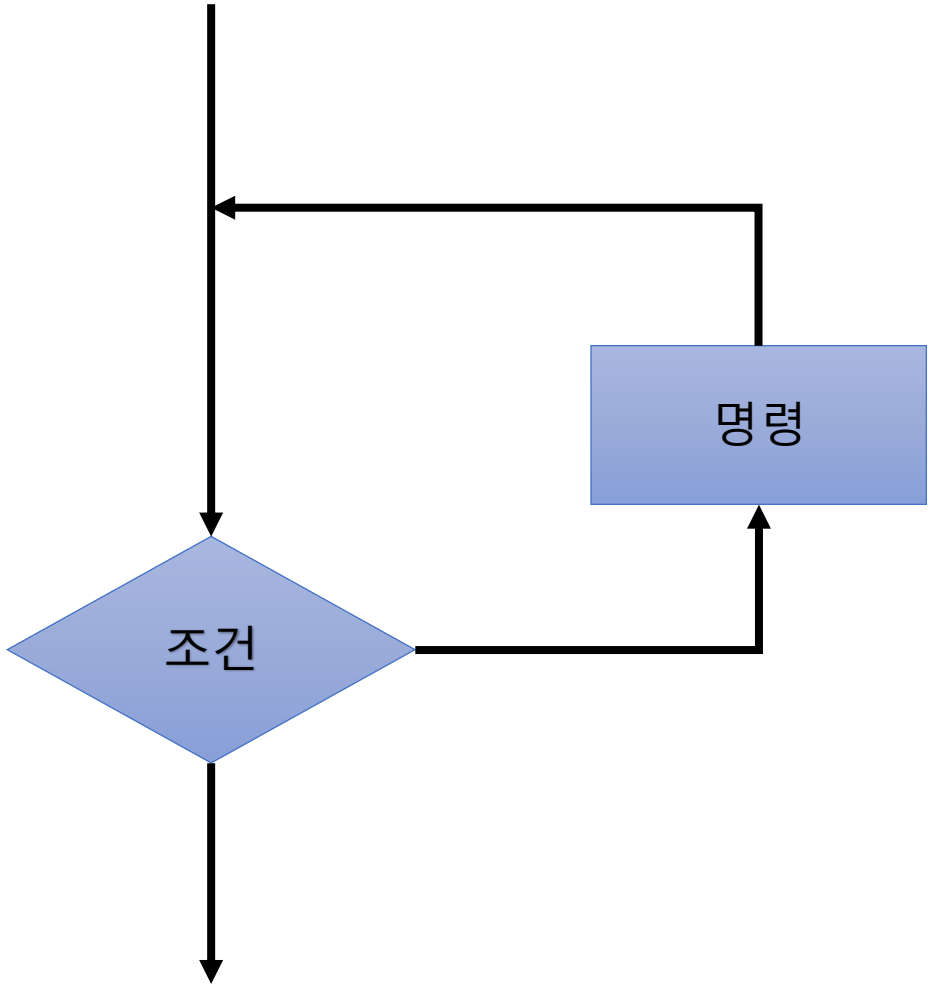
Loop

- 반복해서 구문을 수행

```
명령 1
명령 2

while 〈조건〉:
        while-명령 1
        while-명령 2

명령 3
명령 4
```



- 들여쓰기와 :으로 구문을 구분

While Statement

조건을 만족하는 동안 출력

```
i = 1                       # 초기 값

while i < 10:               # i가 10보다 작은 동안
    print (i)               # i 출력
    i += 1                  # i에 1씩 더함
```

For Statement

- Python의 For문은 주어진 객체를 순환하는 개념
- for [Element] in [Iterable] 의 형태로 사용

```
for i in [0, 1, 2, 3, 4]:          # i가 리스트를 순환
    print (i)

for i in [0, 1, 2, 3, 4]:
    if i % 2:
        print(i, "is odd")
```

For Statement

```
for i in [1, 2, 3, 4, 5]:   # List를 순환
    print ("Test1", i)

for i in range (5):         # 0부터 5미만까지 순환
    print ("Test2", i)

for i in range (1, 6):      # 1부터 6미만까지 순환
    print ("Test3", i)

for i in range (1, 10, 2): # 1부터 10미만까지 2칸씩 뛰며 순환
    print ("Test4", i)
```

- **range** 내장 함수로 숫자 반복 생성 가능 (Generator 반환)
  - range (start, end, step) 형태로 사용, **End는 미포함**
- Generator: 리스트와는 다르게 숫자를 하나씩 생성 반환 (메모리 효율적)

Notions for Loop

- 반복문의 변수명은 일반적으로 i, j, k로 지정
- 반복문을 포함해서 프로그래밍에서 숫자의 시작은 대부분 0부터
- 반복문이 끝나지 않는 무한 loop에 주의
- 모든 순환이 가능한 객체는 for문을 적용하는 것이 가능

```
for c in "This is text":
    print (c)

for word in ["한국어", "문장", "처리"]
    print (word)

for key in {"text": 1, "word": 2}:
    print (key)
```

Controlling Loop – break & continue

Break문으로 가장 바깥의 반복문을 나가는 것이 가능

```
for i in range(1, 100):
    if i % 17 == 0:
        break           # i가 17이상이 되면 반복 나가기
    print (i)
```

Continue문으로 가장 바깥의 반복문의 처음으로 되돌아가기 가능

```
for i in range(100):
    if i % 17:
        continue        # i가 17의 배수가 아니면 for 처음으로
    print (i)
```

Controlling Loop – else

Else문으로 반복을 완전이 돌았을 경우 실행되는 block 지정가능

```
for i in range(10):
    print (i)
else:
    print ("Loop complete with break")
```

Break로 중간에 나오게 되면 Else문이 실행되지 않음

```
for i in range(10):
    print (i)
    if i > 5:
        break
else:
    print ("Loop complete without break")
```