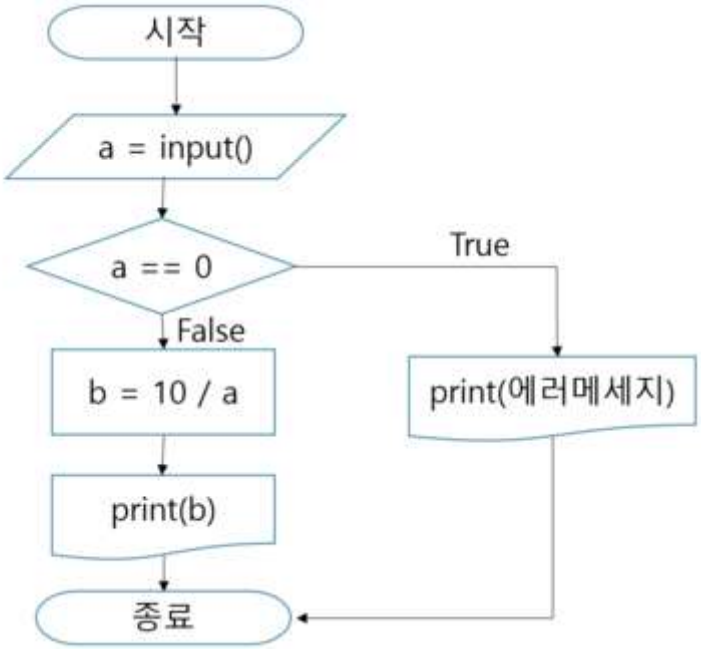
제어문(Control flow statement)

기본적으로 프로그램은 순차구조를 가진다. 즉 작성한 순서대로 실행이 된다.
이런 실행흐름을 다른 순서로 제어하기 위한 구문을 만드는 문법이 제어문이다.
제어문은 **조건문** 과 **반복문** 두가지 문법이 있다.

- 조건문
  - if 문
- 반복문
  - while 문
  - for in 문

조건문/분기문 (conditional statement)

- 프로그램이 명령문들을 실행하는 도중 특정 순서에서 **조건에 따라 흐름의 나눠져야 하는 경우 사용한다**
- 파이썬은 조건문으로 **if문**이 있다.



입력 받은 a 의 값이 0인지 여부에 따라 두가지 흐름으로 분기된다.

구문

- 조건이 True일 경우만 특정 구문들을 실행 하는 조건문.

```
if 조건:    # 조건은 bool 표현식을 기술한다. 조건선언 다음에 : 으로 선언해서 코드블록을 구분한다.
    명령문1  # 조건이 True이면 실행할 구문들을 코드블럭에 기술한다.
    명령문2  # 코드 블록은 들여쓰기를 이용해 묶어준다. 보통 공백 4칸으로 들여쓰기를 한다.
    ...
```

> **파이썬의 코드블록(code block)**
> 코드블록이란 여러개의 실행명령문들을 묶어놓은 것을 말한다. 파이썬에서는 코드블록을 작성할 때 들여쓰기를 이용해 묶어준다. 같은 칸만큼 들여쓰기를 한 명령문들이 같은 블록으로

In [ ]:

- 조건이 True일때 실행할 구문과 False일때 실행하는 조건문.

```
if 조건:
    명령문1_1 # 조건이 True일 경우 실행할 구문들
    명령문1_2
    ...
else:
    명령문2_1 # 조건이 False일 경우 실행할 구문들
    명령문2_2
    ...
```

In [ ]:

- 조건이 여러개인 조건문.

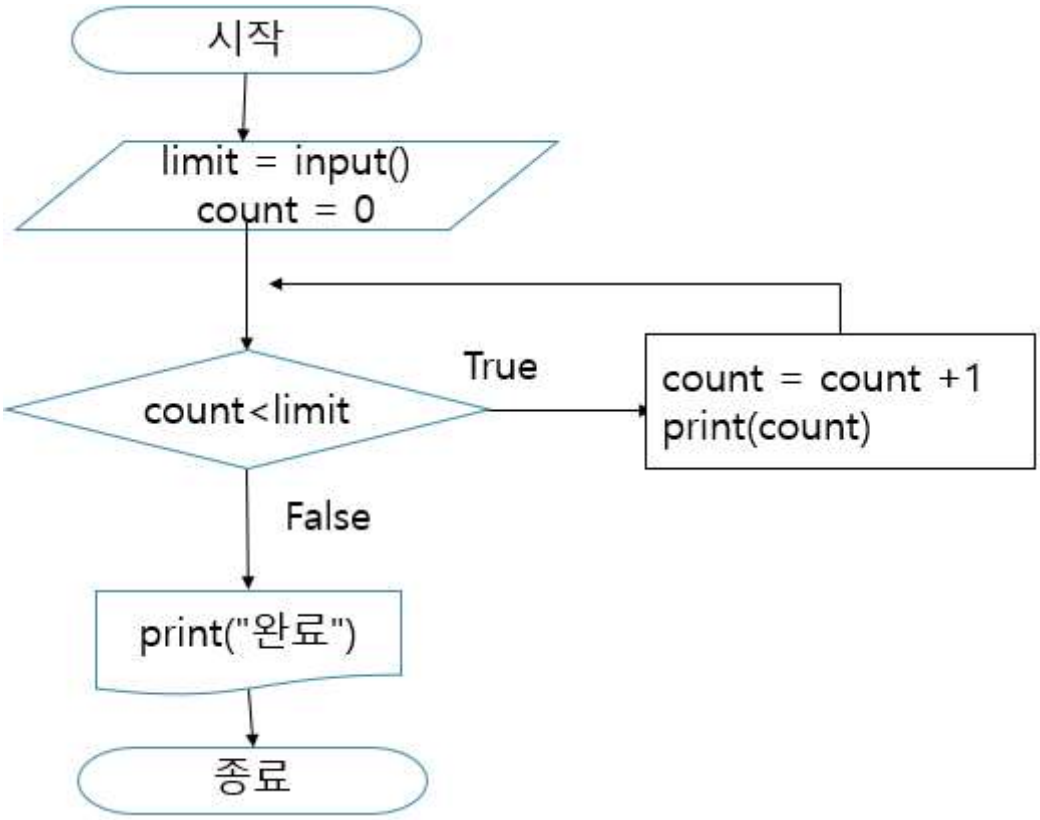
```
if 조건1:
    명령문1_1  # 조건1이 True일 경우 실행할 코드블록.
    명령문1_2
    ...
elif 조건2:    # 다음 조건으로 앞의 조건들이 모드 False일 경우 비교한다.
    명령문2_1  # 조건2가 True일 경우 실행할 코드블록.
    명령문2_2
    ...
elif 조건3 :
    명령문3_1
    명령문3_2
    ...
else:        # 위의 모든 조건이 False일 경우 실행하는 코드블록. 생략 가능하다.
    명령문4
```

In [ ]:

반복문 (Loop statement)

특정 구문들을 반복해서 실행할 때 사용한다. 동일한 코드를 여러번 반복하거나 값이 일정하게 변하는 코드를 반복할 경우 사용한다.
단순 반복을 처리하는 **while문**과 iterable객체가 제공하는 값들을 반복 조회하는 **for in문** 두가지 문법이 있

다.



count의 값이 limit의 값보다 크거나 같을때 까지 count의 값을 1증가 후 출력하는 구문을 반복한다

while문

- 조건이 True인 동안 구문을 반복해서 실행한다.

구문

```
while 조건:       # 조건은 bool 표현식을 기술한다. 조건선언 다음에 : 으로 선언해서 코드블록을 구분한다
    반복할 구문1  # 반복할 구문을 코드블록으로 작성한다.
    반복할 구문2
    ...
```

In [ ]:

for in 문

- Iterable 객체를 순환조회하는 구문
  - for in문은 Iterable 타입의 객체가 가지고 있는 값들을 각각 조회하여 처리하는 구문을 작성할 때 사용한다.

- Iterable
  - 반복가능한 객체. 반복문(for in)을 이용해 일련의 값들을 반복적으로 각각 제공하는 객체를 말한다.
  - 대표적으로 List, Tuple, Dictionary, Set, 문자열 등이 있다.

구문

In [ ]:

```
```

continue와 break를 이용한 반복문 제어

- **continue**
  - 현재 반복을 중단하고 다음 반복을 진행한다.
  - 특정 조건에서 처리를 멈추고 다음 처리를 반복할 때 사용한다.
- **break**
  - 반복을 중단한다.
  - 특정 조건에서 반복문을 중간에 중지할때 사용한다.
- continue와 break는 특정 조건에서 실행되야 하는 경우가 대부분이므로 if문 안에 작성한다.

In [ ]:

```
```

for in 문 연관 내장 함수

range()

- 일정한 간격의 연속된 정수를 제공하는 반복가능 객체 생성한다.
- 구문
  - `range([시작값], 멈춤값, [증감값])`
    - 시작값, 멈춤값, 증감값 모두 정수만 가능하다.
    - 시작값 > 멈춤값 이고 증감값이 음수이면 내림차순으로 값을 제공한다.

    1. 전달값이 **1개: 멈춤값**.
       - 0 ~ (멈춤값-1)까지 1씩 증가하는 정수를 제공
    2. 전달값이 **2개: 시작값, 멈춤값**.
       - 시작값 ~ (멈춤값-1) 까지 1씩 증가하는 정수 제공
    3. 전달값이 **3개: 시작값, 멈춤값, 증감값(간격)**.
       - 시작값 ~ (멈춤값-1)까지 증감값만큼 증가하는 정수를 제공.

In [ ]:

```
```

enumerate()

- 구문
  - enumerate(Iterable, [, start=정수])
    - 현재 몇번째 값을 제공하는 지(현재 몇번째 반복인지)를 나타내는 **index**와 제공하는 **원소**를 tuple로 묶어서 반환
    - Iterable
      - 값을 제공할 Iterable객체
    - start: 정수
      - index 시작 값. 생략하면 0부터 시작한다.

In [ ]:

zip()

- 여러 개의 Iterable 객체를 받아 반복시 같은 index의 값끼리 튜플로 묶어 반환한다.
- 구문
  - zip(Iterable1, Iterable2, Iterable3 [, ...])
    - Iterable 2개이상.전달한다.
- 각 자료구조 객체의 크기가 다를 경우 크기가 작은 것의 개수에 맞춰 반복한다.

In [ ]:

컴프리헨션(Comprehension)

- 기존 Iterable의 원소들을 이용해서 새로운 자료구조(List, Dictionary, Set)를 생성하는 구문.
  - 기존 Iterable의 **원소들을 처리한 결과**나 **특정 조건이 True인 값들을** 새로운 자료구조에 넣을때 사용.
  - 결과를 넣을 새로운 자료구조에 따라 다음 세가지 구문이 있다.
    - 리스트 컴프리헨션
    - 딕셔너리 컴프리헨션
    - 셋 컴프리헨션
- 튜플 컴프리헨션은 없다.
- 딕셔너리 컴프리헨션과 셋 컴프리헨션은 파이썬 3 에 새로 추가되었다.

In [ ]:

TODO

In [ ]:

```
#(1) 다음 점수 구간에 맞게 학점을 출력하세요.
# 91 ~ 100 : A학점
# 81 ~ 90 :  B학점
# 71 ~ 80 :  C학점
# 61 ~ 70 :  D학점
# 60이하   :  F학점


#(2) 사용자로 부터 ID를 입력 받은 뒤 입력받은 ID가 5글자 이상이면 "사용할 수 있습니다."를 5글자 [

#(3) 사용자로부터 우리나라 도시명을 입력 받은 뒤 입력받은 도시명이 서울이면 "특별시"를 인천,부산,

#(4) 아래 리스트의 평균을 구하시오.
jumsu = [100, 90, 100, 80, 70, 100, 80, 90, 95, 85]

#(5) 위 jumsu리스트에서 평균점수이상은 pass, 미만은 fail을 index번호와 함께 출력하시오. (ex: 0-pa

#(6) 아래 리스트 값들 중 최대값을 조회해 출력
jumsu = [60, 90, 80, 80, 70, 55, 80, 90, 95, 85]

#(7) 다음 리스트 중에서 "쥐"와 "토끼" 제외한 나머지를 출력하세요.
str_list = ["쥐", "소", "호랑이", "토끼", "용", "뱀"]

#(8) 사용자로부터 정수를 입력받아 그 단의 구구단을 출력하시오.
# ex) 단을 입력하시오 : 2
# 2 x 1 = 2
# 2 x 2 = 4
#..
# 2 x 9 = 18

#컴프리헨션

#(9) 다음 리스트가 가진 값에 두배(* 2)를 가지는 새로운 리스트를 만드시오. (리스트 컴프리헨션 이용
lst = [10, 30, 70, 5, 120, 700, 1, 35]

#(10) 다음 리스트가 가진 값에 10배의 값을 가지는 값을 (원래값, 10배값) 의 튜플 묶음으로 가지는 리
# Ex) [(10,100), (30,300), .., (35, 350)]
lst = [10, 30, 70, 5, 5, 120, 700, 1, 35, 35]



#(11) 다음 리스트가 가진 값들 중 3의 배수만 가지는 리스트를 만드시오. (리스트 컴프리헨션 이용)
lst2 = [ 3, 20, 33, 21, 33, 8, 11, 10, 7, 17, 60, 120, 2]

#(12) 다음 파일이름들을 담은 리스트에서 확장자가 exe인 파일만 골라서 새로운 리스트에 담으시오.(st
file_name=["test.txt", "a.exe", "jupyter.bat", "function.exe", "b.exe", "cat.jpg", "dog.png", "ru

#(13) 다음 중 10글자 이상인 파일명(확장자포함)만 가지는 리스트를 만드시오.
file_name=["mystroy.txt", "a.exe", "jupyter.bat", "function.exe", "b.exe", "cat.jpg", "dog.png",
```

```
#(14) 다음 리스트에서 소문자만 가지는 새로운 리스트를 만드시오.
str_list = ["A", "B", "c", "D", "E", "F", "g", "h", "I", "J", "k"]
```