제어문(Control Statement)

파이썬 문서

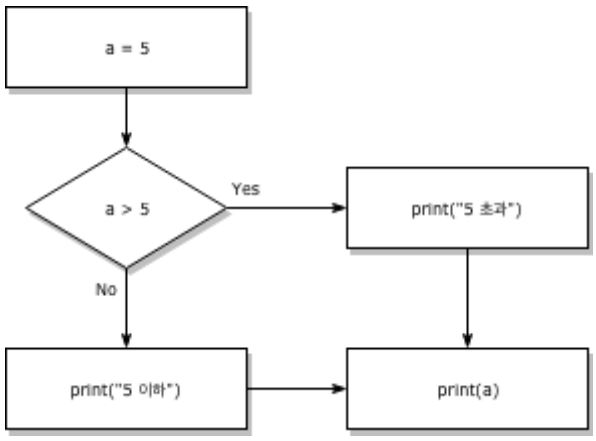
지금까지 우리는 위에서부터 아래로 순차적으로 명령을 수행하는 프로그램을 작성하였습니다.

특정 상황에 따라 코드를 선택적으로 실행(분기)하거나 동일한 코드를 계속해서 실행해야하려면 어떻게 해야할까요?

이 경우, **코드 실행의 순차적인 흐름을 제어**(Control Flow)할 필요가 있습니다.

이러한 순차적인 코드의 흐름을 제어하는 것을 제어문이라고 하고, 제어문은 크게 **조건문**과 **반복문**으로 나눌 수 있습니다.

제어문을 통해 다음과 순서도(Flow Chart)를 코드로 표현할 수 있습니다.



In [ ]:
```
# 위의 Flow Chart를 조건문으로 작성해봅시다.
```

In [ ]:
```
a = 5
if a > 5:
    print('5 초과')
else:
    print('5 이하')
print(a)
```

조건문(Conditional Statement)

`if` 문은 반드시 **참/거짓을 판단할 수 있는 조건**과 함께 사용이 되어야한다.

`if` 조건문의 구성

활용법

- **문법**

```
if <expression>:
    <코드 블럭>
else:
    <코드 블럭>
```

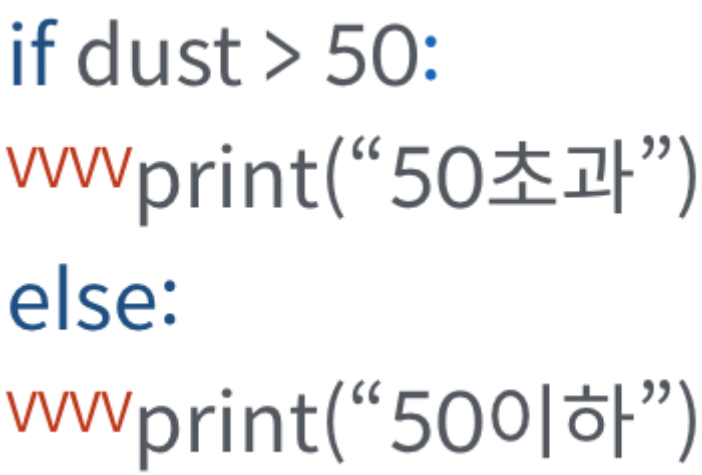
- **예시**

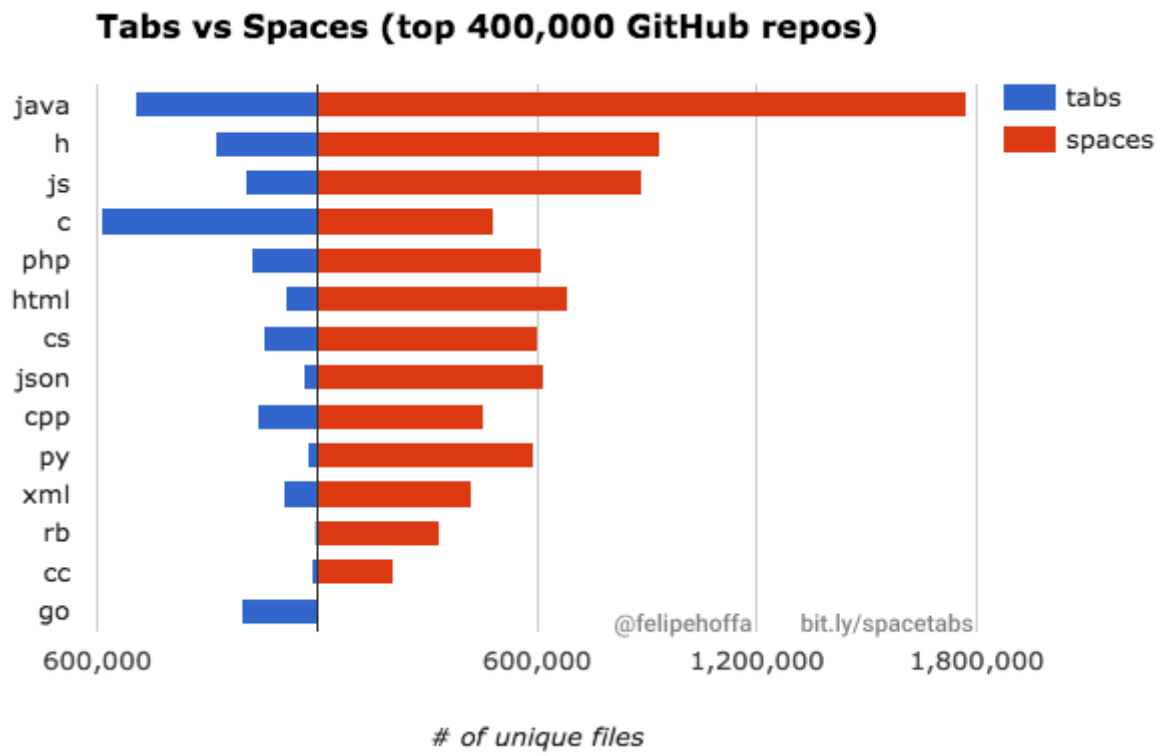
```
if a > 0:
    print('양수입니다.')
else:
    print('음수입니다.')
```

- `expression` 에는 일반적으로 참/거짓에 대한 조건식이 들어간다.
- **조건**이 **참**인 경우 `:` 이후의 문장을 수행한다.
- **조건**이 **거짓**인 경우 `else:` 이후의 문장을 수행한다.
- 여러 개의 `elif` 부가 있을 수 있고(없거나), `else` 는 선택적이다.

주의사항

- 이때 반드시 **들여쓰기**를 유의해야한다.
  - 파이썬에서는 코드 블록을 자바나 C언어의 `{}` 와 달리 **들여쓰기**로 판단하기 때문이다.
- 앞으로 우리는 PEP 8에서 권장하는 **4spaces**를 사용할 것이다.





출처 : 400,000 GitHub repositories, 1 billion files, 14 terabytes of code: Spaces or Tabs?

[연습] 크리스마스 판독기

> 조건문을 통해 사용자가 입력한 날짜가 크리스마스인지 확인하세요.

[입력 예시]

12/25

[출력 예시]

크리스마스입니다.

In [ ]:
```
is_christmas = input('날짜를 입력해주세요 ex)12/24 : ')
print(is_christmas)

# 아래에 코드를 작성하세요
```

In [ ]:
```
print(is_christmas)
if is_christmas == '12/25':
    print('크리스마스입니다.')
else:
    print('크리스마스가 아닙니다.')
```

[실습] 홀/짝 판독기

> 조건문을 통해 변수 num의 값과 홀수/짝수 여부를 출력하세요.

[입력 예시]

3

[출력 예시]

홀수입니다.

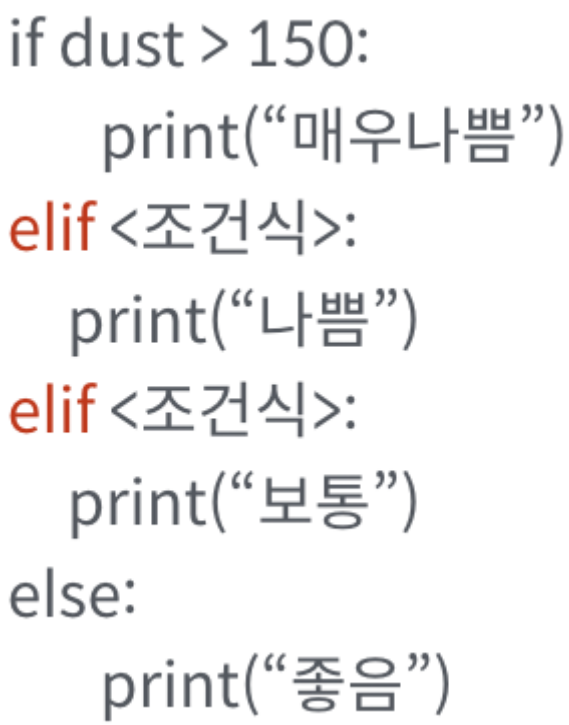
In [ ]:
```
num = int(input('숫자를 입력하세요 : '))
if num % 2 == 1:
    print('홀수입니다.')
else:
    print('짝수입니다.')

# 아래에 코드를 작성하세요
```

In [ ]:
```
# 0 -> False
# 나머지 모든 숫자는 -> True 대표적 1
if num % 2:
    print('홀수')
else:
    print('짝수')
```

elif 복수 조건

2개 이상의 조건을 활용할 경우 `elif <조건>:` 을 활용한다.



elif는 사용할 때 조건이 붙어야한다!

[연습] 복수 조건문 연습

> 조건문을 통해 변수 score에 따른 평점을 출력하세요.

점수	등급
90점 이상	A
80점 이상	B

점수	등급
70점 이상	C
60점 이상	D
60점 미만	F

[입력 예시]

85

[출력 예시]

B

In [ ]:
```
score = int(input('점수를 입력하세요 : '))
if score >= 90:
    print('A')
elif score >= 80:
    print('B')
elif score >= 70:
    print('C')
elif score >= 60:
    print('D')
# else는 나머지 모든 케이스를 뜻하기 때문에, 조건식이 들어갈 수 없음!
else:
    print('F')

# 아래에 코드를 작성하세요
# 위에 기준범위들의 순서가 바뀌면 안된다!
```

In [ ]:
```
if score >= 90:
    print('A')
# 90이상이었으면, 위의 코드가 실행되고 끝났을 것.
# 그게 아니니까, 90미만인 숫자일 수 밖에 없음!
elif score >= 70:
    print('C')
elif score >= 80:
    print('B')
else:
    print('F')
```

In [ ]:
```
if 90 <= score <= 100:
    print('A')
elif 70 <= score <= 80:
    print('C')
elif 80 <= score <= 90:
    print('B')
else:
    print('F')
```

중첩 조건문(Nested Conditional Statement)

조건문은 다른 조건문에 중첩될 수도 있습니다.

[연습] 중첩 조건문 활용

> 위 실습문제 2개 코드를 활용하여 95점 이상이면, "참 잘했어요"도 함께 출력해주세요.

[출력 예시]

A

참잘했어요.

In [ ]:
```
score = 96

# 아래에 코드를 작성하세요
```

In [ ]:
```
if score >= 90:
    print('A')
    if score >= 95:
        print('참 잘했어요')
```

조건 표현식(Conditional Expression) 지금은 SKIP!!

- 조건 표현식은 일반적으로 조건에 따라 값을 정할 때 활용된다.
- **삼항 연산자(Ternary Operator)**라고 부르기도 한다.

활용법

```
true_value if <조건식> else false_value
```

In [ ]:
```
num = int(input('숫자를 입력하세요 : '))

print('0 보다 큼') if num > 0 else print('0 보다 크지않음')
```

In [ ]:
```
# 아래의 코드는 무엇을 위한 코드일까요?
num = int(input('숫자를 입력하세요 : '))
value = num if num >= 0 else -num
print(value)
```

In [ ]:
```
# 아래에 답변을 작성하세요.
```

In [ ]:
```

```

[연습] 조건 표현식 작성하기

> 다음의 코드와 동일한 조건 표현식을 작성해보세요.

```
num = 2
if num % 2:
    result = '홀수입니다.'
else:
```

```
    result = '짝수입니다.'
print(result)
```

[출력 예시]

짝수입니다.

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```
num = int(input())
```

[실습] 조건 표현식과 동일한 if 문 작성하기

> 다음의 코드와 동일한 `if` 문을 작성해보세요.

```
num = -5
value = num if num >= 0 else 0
print(value)
```

[출력 예시]

0

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```

```

반복문(Loop Statement)

- while
- for

`while` 반복문

`while` 문은 조건식이 참(`True`)인 경우 반복적으로 코드를 실행한다.

활용법

- **문법**

```
while <조건식>:
    <코드 블럭>
```

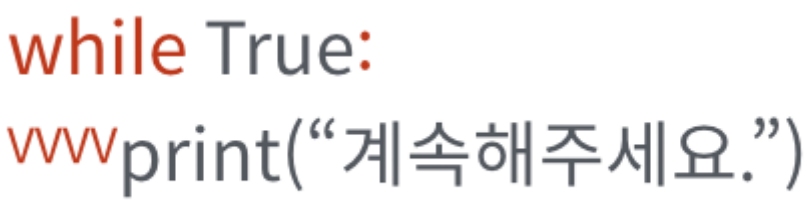
- 예시

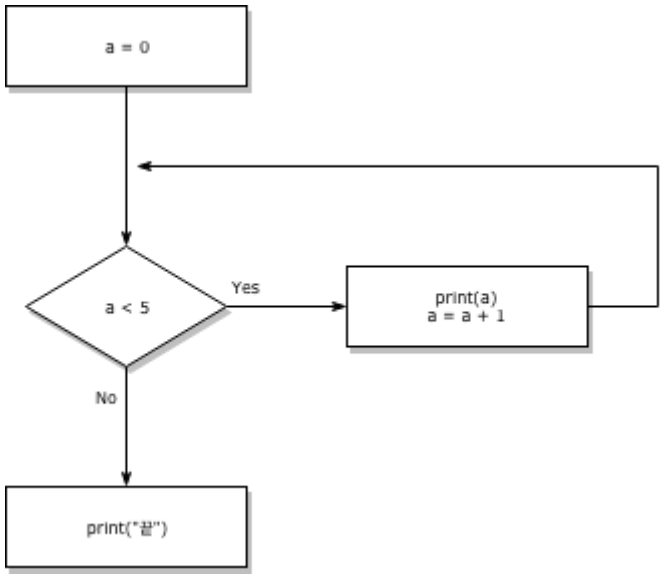
```
while True:
    print('조건식이 참일 때까지')
```

```
    print('계속 반복')
```

주의사항

- `while` 문 역시 조건식 뒤에 콜론(:)이 반드시 필요하며, 이후 실행될 코드 블럭은 **4spaces**로 **들여쓰기**를 한다.
- **반드시 종료조건을 설정해야 한다.**





In [ ]:
```
# 위의 flow chart를 조건문을 통해 만들어봅시다.
# 아래에 코드를 작성하세요.
```

In [ ]:
```
a = 0
while a < 5:
    print(a)
    a += 1
print('끝')
```

[연습] while 문 작성하기

> 사용자가 "안녕"이라고 입력할 때까지 인사하는 코드를 작성해보세요.

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```

```

[실습] 합(Summation)

> 1부터 사용자가 입력한 양의 정수까지의 총합을 구하는 코드를 작성해보세요.

[입력 예시]

10

[출력 예시]

55

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```

```

[응용] 한자리 씩 출력하기

> 사용자로부터 숫자 입력 받은 양의 정수의 각 자리 수를 1의 자리부터 차례대로 출력하는 코드를 작성해보세요.

[입력 예시]

185

[출력 예시]

5

8

1

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```
10으로 나눈거 나머지를 출력, 몫은 다시 넣고
```

for 문

for 문은 시퀀스(string, tuple, list, range)나 다른 순회가능한 객체(iterable)의 요소들을 순회한다.

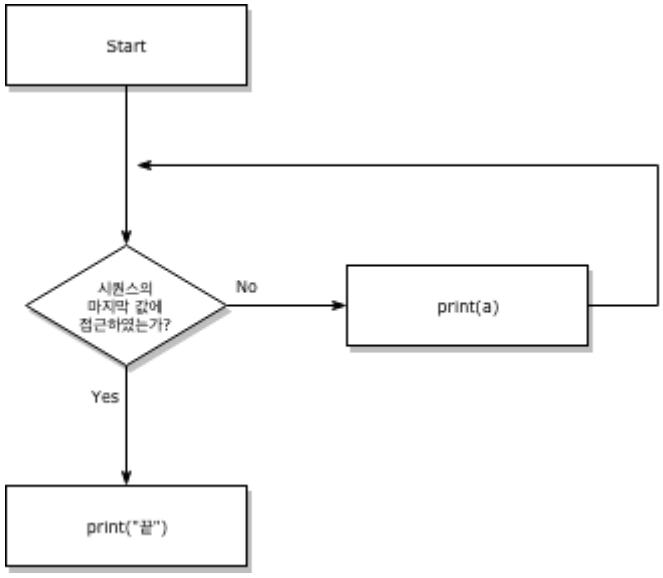
활용법

- **문법**

```
for <임시변수> in <순회가능한데이터(iterable)>:
    <코드 블럭>
```

- **예시**

```
for menu in ['김밥', '햄버거', '피자', '라면']:
    print(menu)
```



In [ ]:
```
# flowchart를 for문을 통해 코드로 작성해봅시다.
# 아래에 코드를 작성하세요.
```

In [ ]:
```
# range(10) => 0 ~ 9
for i in range(10):
    # i = 하나씩 꺼내서 i에 저장!
    print(i)
```

In [ ]:
```
for char in '김혜인':
    # char = 하나씩 꺼내서 char에 저장
    print(char)
```

반복 for (리스트)

정해진 시퀀스 내에서의 반복 시 사용

'가지고 있는 모든 것을 꺼낸다'

```
dust = [59, 24, 102]
for i in dust:      i = 102
    print(i)        print(102)
```

```
59 24 102
```

for 문에서 요소 값에 다른 값을 할당해도 다음 반복구문에 영향을 주지 않는다.

다음 요소 값에 의해 덮어 씌워지기 때문이다.

In [ ]:
```
# for 문 안에서 임시 변수에 다른 값을 할당해도 반복구문에 영향을 주지 않습니다.
```

In [ ]:
```
for i in range(3):
    # i = 0, i = 1, i = 2
    print(i)
    i = 5
```

[연습] for 문 작성하기

> for 문을 활용하여 사용자가 입력한 문자를 한글자씩 출력해보세요.

[입력 예시]

문자를 입력하세요 : 안녕!

[출력 예시]

안
녕
!

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```
chars = input('문자를 입력하세요 : ')
```

[실습] for 문과 if 문 작성하기

> 반복문과 조건문만 활용하여 1~30까지 숫자 중에 홀수만 출력해보세요.

[출력 예시]

1
3
5
...
27
29

In [ ]:
```
# 아래에 코드를 작성하세요.
```

In [ ]:
```

```

리스트(list) 순회에서 index의 활용하기

range(리스트의 길이)

range() 와 순회할 list의 길이를 활용하여 index를 조작 가능

In [ ]:
```
# range 함수로 0~3 범위를 반복하는 for 문을 작성해봅시다.
```

In [ ]:
```

```

In [ ]:
```
# range(리스트의 길이)를 활용해서 출력해봅시다.
lunch = ['짜장면', '초밥', '피자', '햄버거']
```

In [ ]:
```
for idx in  range(len(lunch)):
    print(lunch[idx])   # 전체 길이는 4이니까 range하면 0,1,2,3
```

In [ ]:
```
# 순서를 함께 출력해봅시다.
for menu in lunch:
    print(menu)
```

enumerate()

인덱스(index)와 값(value)을 함께 활용 가능

> enumerate() 를 활용하면, 추가적인 변수를 활용할 수 있습니다.
>
> - enumerate() 는 내장 함수 중 하나이며, 다음과 같이 구성되어 있습니다.
>
> **enumerate**(*iterable*, *start=0*)
> 열거 객체를 돌려줍니다. *iterable* 은 시퀀스, 이터레이터 또는 이터레이션을 지원하는 다른 객체여야 합니다. enumerate() 에 의해 반환된 이터레이터의 __next__() 메서드는 카운트 (기본값 0을 갖는 *start* 부터)와 *iterable* 을 이터레이션 해서 얻어지는 값을 포함하는 튜플을 돌려줍니다.

In [ ]:
```
# enumerate()를 활용해서 출력해봅시다.
lunch = ['짜장면', '초밥', '피자', '햄버거']
```

In [ ]:
```

```

In [ ]:
```
# enumerate()를 사용하였을 때 어떻게 표현이 되는지 확인해봅시다.
```

In [ ]:
```
classroom = ['Kim', 'Hong', 'Kang']ss
```

In [ ]:
```
# 숫자를 1부터 카운트 할 수도 있습니다.
```

In [ ]:
```

```

In [ ]:
```

```

반복제어(break, continue, for-else)

break

반복문을 종료한다.

- `for` 나 `while` 문에서 빠져나간다.

In [ ]:
```
# while 문에서 break를 활용해봅시다.
```

In [ ]:
```
n = 0
while n < 3:
    print(n)
    n += 1
```

In [ ]:
```
#
n = 0
while True:
    if n == 3:
        break
    print(n)
    n += 1
```

In [ ]:
```
# for 문에서도 break를 사용할 수 있습니다.
```

In [ ]:
```
for i in range(10):
    if i > 1:
        print('0과 1만 필요해!')
        break
    print(i)
```

[연습] break 활용하기

> 조건문과 반복문, break 를 활용하여 리스트에서 쌀이 나왔을때 for 문을 멈추는 코드를 작성하세요.

[출력 예시]

보리

보리

쌀

잡았다!

In [ ]:
```
rice = ['보리', '보리', '보리', '쌀', '보리']

# 아래에 코드를 작성하세요.
```

In [ ]:
```
for i in rice:
    print(i)
    if i == '쌀':
        print('잡았다!')
        break
```

continue

`continue` 문은 continue 이후의 코드를 수행하지 않고 다음 요소부터 계속*(continue)*하여 반복을 수행한다.

In [ ]:
```
# continue 문을 활용해봅시다.
```

In [ ]:
```
for i in range(6):
    if i % 2 == 0:
        continue
        # continue 이후의 코드는 실행되지 않습니다.
    print(f'{i}는 홀수다.')
```

[연습] continue 문 작성하기

> 나이가 입력된 리스트가 있을때, 조건문과 반복문, continue를 활용하여 20살 이상일때만 "성인입니다"라는 출력을 하는 코드를 작성하세요.

[출력 예시]

23 살은 성인입니다.

30 살은 성인입니다.

25 살은 성인입니다.

31 살은 성인입니다.

In [ ]:
```
ages = [10, 23, 8, 30, 25, 31]

# 아래에 코드를 작성하세요.
```

In [ ]:
```
for age in ages:
    if age < 20:
        continue    # 반복문 순회를 돌린다.
    print(f'{age}살은 성인입니다.')
```

else

끝까지 반복문을 실행한 이후에 실행된다.

- 반복에서 리스트의 소진이나 (`for` 의 경우) 조건이 거짓이 돼서 (`while` 의 경우) 종료할 때 실행된다.
- 하지만 반복문이 **`break` 문으로 종료될 때는 실행되지 않는다.** (즉, `break` 를 통해 중간에 종료되지 않은 경우만 실행)

In [ ]:
```
# break가 안되는 상황을 만들어봅시다.
```

In [ ]:
```
for i in range(3):
```

```
    print(i)
    if i == 100:
        print(f'{i}에서 break 실행됨.')
        break
else:
    print("break 실행안됨.")
```

In [ ]:
```
# break가 되는 상황을 만들어봅시다.
```

In [ ]:
```
for i in range(3):
    print(i)
    if i == 1:
        print(f'{i}에서 break 시행됨.')
        break
else:
    print('break 시행안됨.')
```

[연습] for-else 활용하기

> 조건문과 반복문, break, else 를 통해서 아래의 코드와 동일한 코드를 작성하세요.
>
> - numbers 리스트에 4가 있을 경우 `True` 를 출력하고, 없을 경우 `False` 를 출력한다.

[출력 예시]

```
False
```

In [ ]:
```
numbers = [1, 3, 7, 9]

# 아래에 코드를 작성하세요.
```

In [ ]:
```
for num in numbers:
    if num == 5:
        print("True")    # 5가 들어가면 else분기는 시행안됨.
        break
else:
    print('False')
```

pass

아무것도 하지 않는다.

- 문법적으로 문장이 필요하지만, 프로그램이 특별히 할 일이 없을 때 자리를 채우는 용도로 사용할 수 있다.

pass 와 continue 차이

In [ ]:
```
# pass
for i in range(5):
    if i == 3:
        pass
    print(i)
```

In [ ]:

```
# continue
for i in range(5):
    if i == 3:
        continue
    print(i)
```