2-2장 반복문

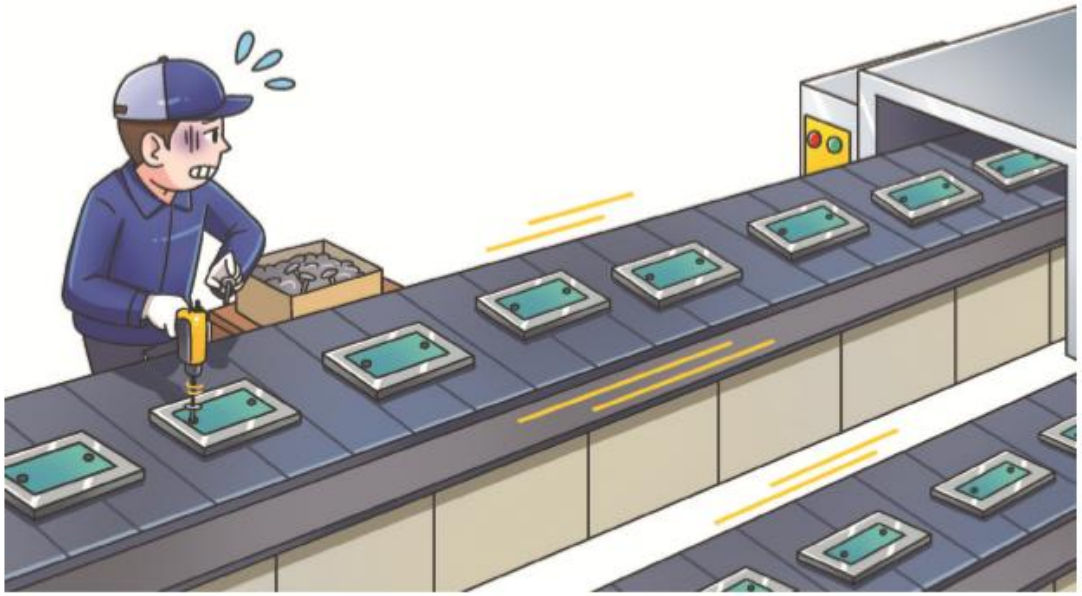
반복의 필요성

- 회사에 중요한 손님이 오셔서 화면에 "환영합니다."을 5번 출력해야 한다고 가정하자. 이제까지 학습한 방법만을 사용하면 다음과 같이 print() 함수를 호출하는 문장을 5번 되풀이해야 한다.

```
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
```

컴퓨터와 반복

- **반복적이고 지루한 작업은 컴퓨터를 이용하여 자동화하여야 한다. 동일한 작업을 오류 없이 반복하는 것은 컴퓨터가 아주 잘 할 수 있는 일이다.**



반복 구조를 사용하면

```
for x in range(5) :
        print("환영합니다.")
```

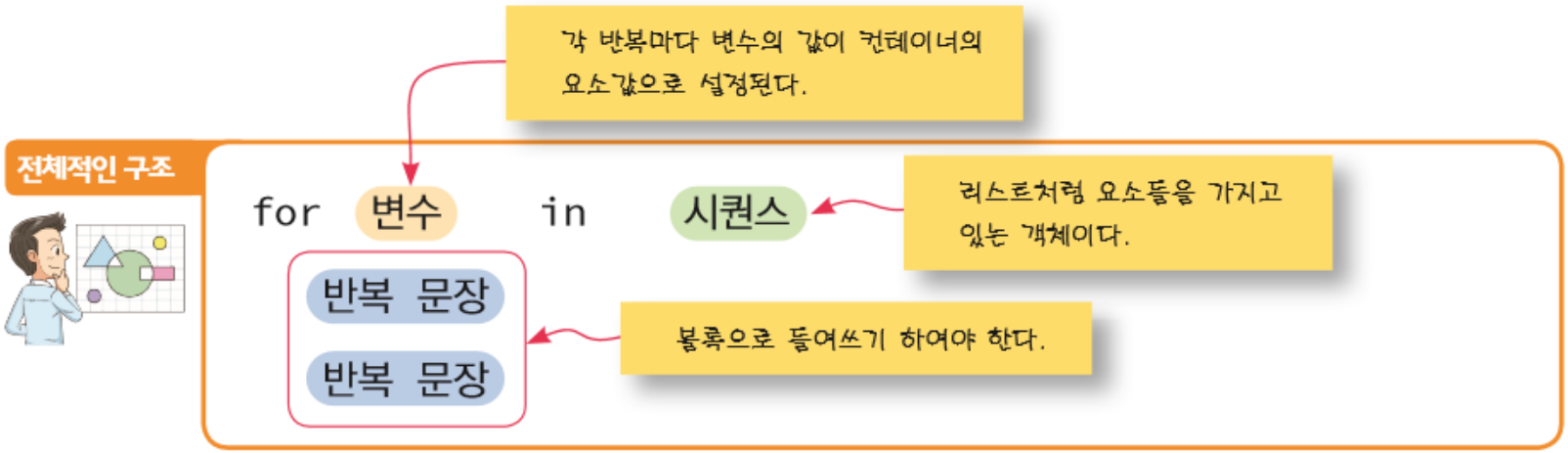
```
환영합니다.
환영합니다.
환영합니다.
환영합니다.
환영합니다.
```

2가지의 반복 구조

for 문 - 정해진 횟수만큼 반복하는 구조이다.

while 문 - 어떤 조건이 만족되는 동안, 반복을 계속하는 구조이다.

FOR 문

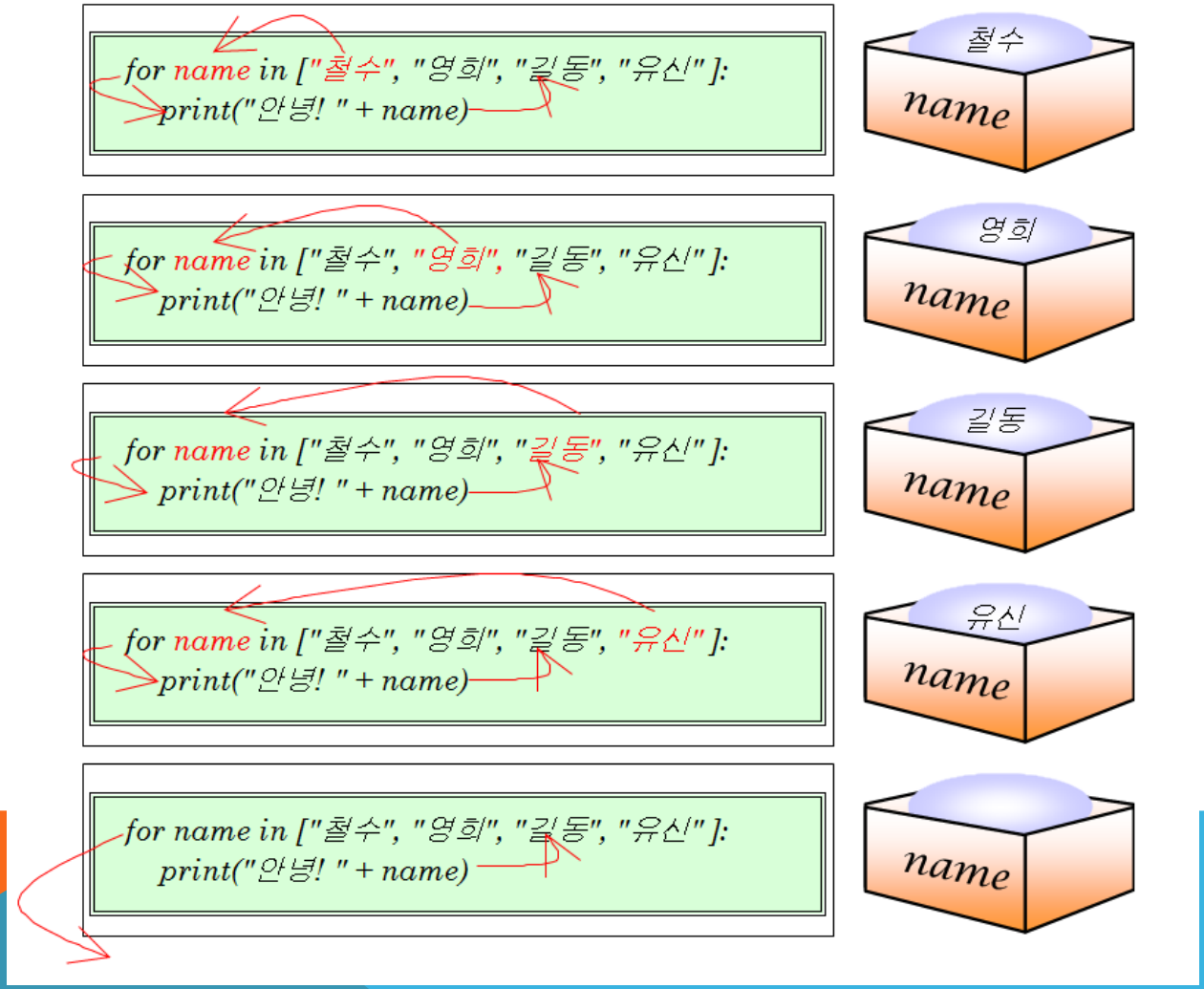

각 반복마다 변수의 값이 컨테이너의 요소값으로 설정된다.
전체적인 구조
for
변수
in
시퀀스
리스트처럼 요소들을 가지고 있는 객체이다.
반복 문장
반복 문장
블록으로 들여쓰기 하여야 한다.

리스트에 대한 반복

```
for name in ["철수", "영희", "길동", "유신" ]:
        print("안녕! " + name)
```

```
안녕! 철수
안녕! 영희
안녕! 길동
안녕! 유신
```

리스트 반복 이해하기


for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
철수
name
for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
영희
name
for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
길동
name
for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
유신
name
for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
name

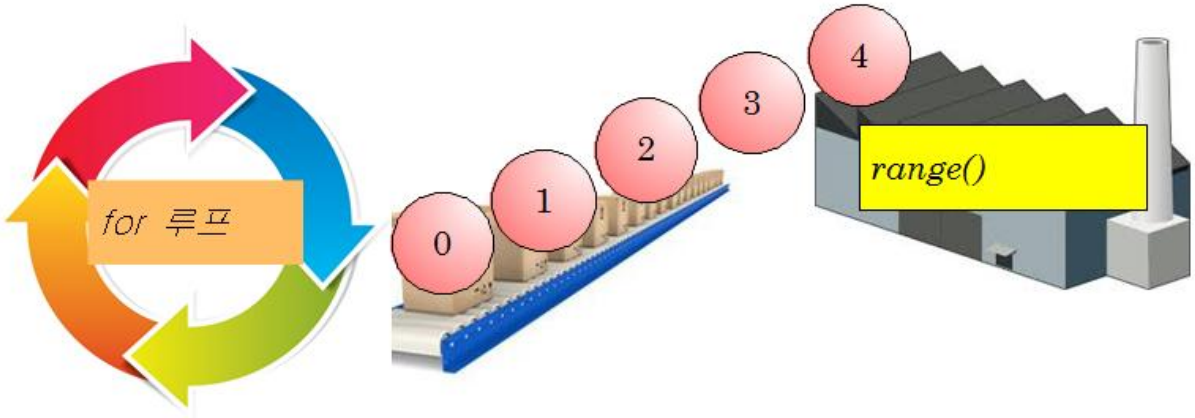
정수 리스트에 대한 반복

```
for x in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] :
    print(x, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

RANGE() 함수

- range() 함수를 이용하면 특정 구간의 정수들을 생성할 수 있다. 예를 들어서 range(10)하면 0부터 9까지의 정수가 생성된다.



예제

```
sum = 0
for x in range(10) :
    sum = sum + x
print(sum)
```

```
45
```

range(start, stop)와 같이 호출하면 start부터 시작하여서 (stop-1)까지의 정수가 생성된다. 이때 stop은 포함되지 않는다.

```
sum = 0
for x in range(0, 10) :
    sum = sum + x
print(sum)
```

```
45
```

RANGE() 함수 정리

전체적인 구조

```
range( [start ,] stop  [, step ])
```

range() 함수는 start부터 stop-1까지 step의 간격으로 정수들을 생성한다. start와 step이 대괄호로 싸여져 있는데 이는 생략할 수 있다는 의미이다. start나 step이 생략되면 start는 0, step은 1로 간주된다.

문자열 반복

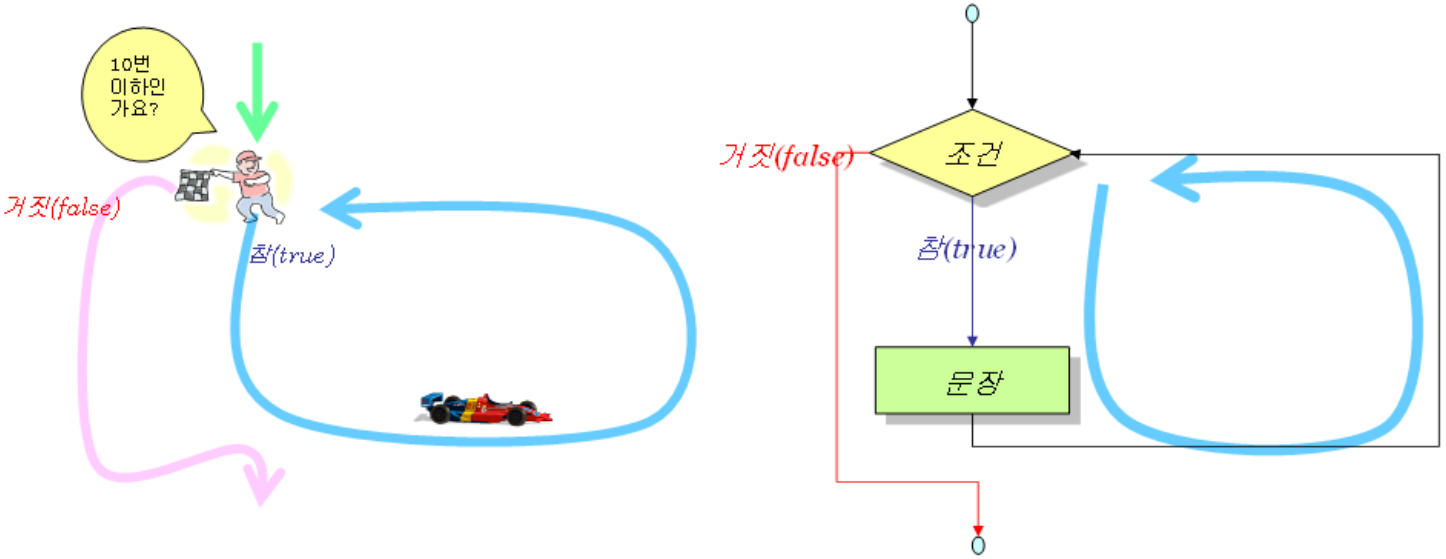
문자열도 시퀀스의 일부분이다. 따라서 문자열을 대상으로 반복문을 만들 수 있다.

```
for c in  "abcdef":
        print(c, end=" ")
```

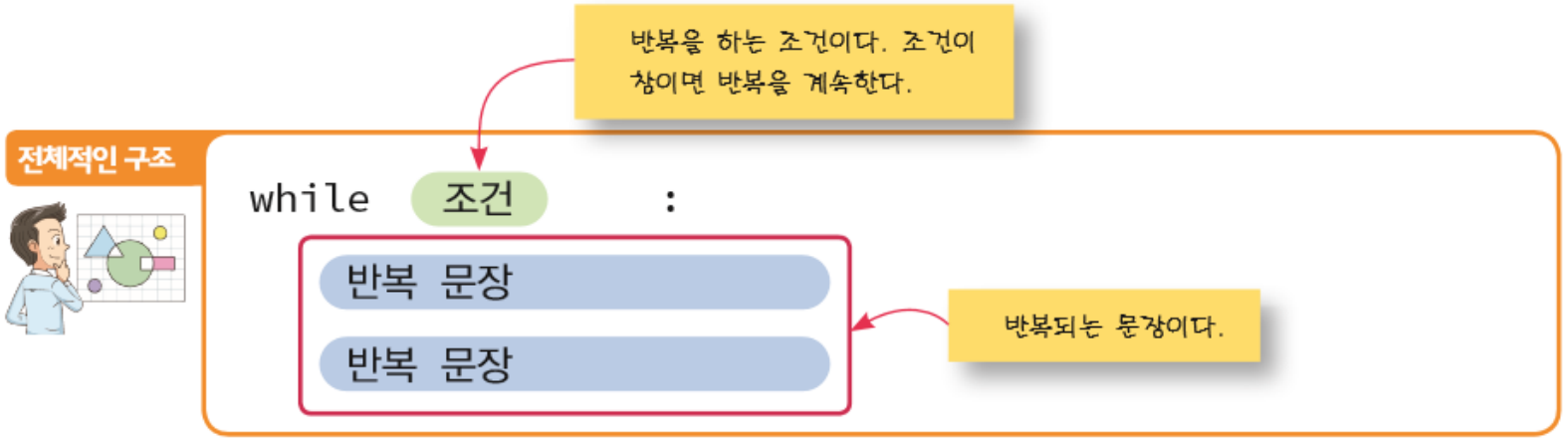
```
a b c d e f
```

WHILE 문

while 문은 조건을 정해놓고 반복을 하는 구조이다.



WHILE 문의 구조


반복을 하는 조건이다. 조건이 참이면 반복을 계속한다.
전체적인 구조
while 조건 :
반복 문장
반복 문장
반복되는 문장이다.

예제

```
i = 0;
while i < 5 :
    print("환영합니다.")
    i = i + 1
print("반복이 종료되었습니다.")
```

```
환영합니다.
환영합니다.
환영합니다.
환영합니다.
환영합니다.
반복이 종료되었습니다.
```

보초값(SENTINEL) 사용하기

- 만약 입력될 데이터의 정확한 개수가 미리 알려지지 않거나 데이터가 너무 많아서 개수를 알기가 어려운 경우에는 어떻게 하는 것이 좋을까? 이런 경우에는 데이터의 끝에다 끝을 알리는 특수한 데이터를 놓으면 된다.

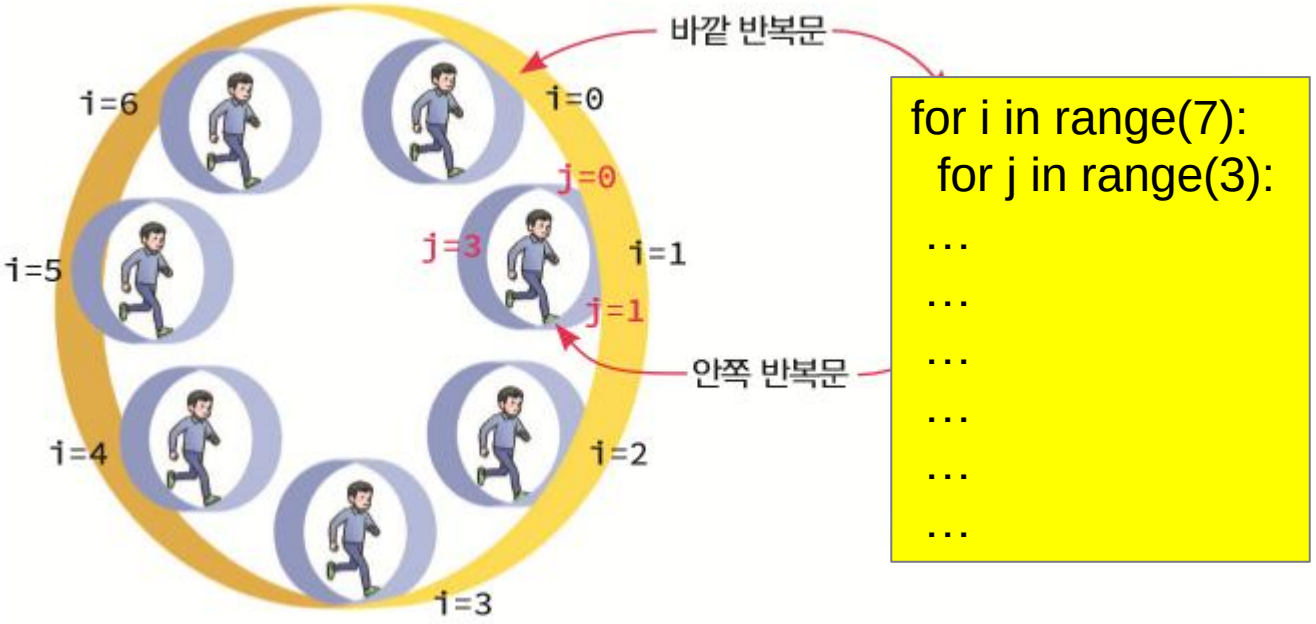
예제

- **사용자로부터 임의의 개수의 성적을 받아서 평균을 계산한 후에 출력하는 프로그램 을 작성하여 보자. 센티널로는 음수의 값을 사용하자.**

```
종료하려면 음수를 입력하시오
성적을 입력하시오: 10
성적을 입력하시오: 20
성적을 입력하시오: 30
성적을 입력하시오: -1
성적의 평균은 20.000000 입니다.
```

중첩 루프

반복문은 중첩하여 사용될 수 있다. 즉 반복문 안에 다른 반복문이 포함될 수 있다.



예제

```
# 중첩 for 문을 이용하여 *기호를 사각형 모양으로 출력하는 프로그램

for y in range(5):
        for x in range(10):
                print("*", end="")
        print("")               # 내부 반복문이 종료될 때마다 실행
```

```
**********
**********
**********
**********
**********
```

문자열 처리하기

문자열도 시퀀스의 일종

```
fruit = "apple"
for letter in fruit:
   print(letter, end=" ")
```

```
a p p l e
```

예제

문자열을 받아서 모음을 전부 없애는 코드는 다음과 같이 작성할 수 있다.

```
s = input('문자열을 입력하시오: ')
vowels = "aeiouAEIOU"
result  = ""
for letter in s:
    if letter not in vowels:
        result += letter
print(result)
```

```
문자열을 입력하시오: kkkoommm
kkkmmm
```

예제

문자열 중에서 자음과 모음의 개수를 집계하는 프로그램을 작성하여 보자.

```
original = input('문자열을 입력하시오: ')
word = original.lower()
vowels = 0
consonants = 0

if len(original) > 0 and original.isalpha():
    for char in  word:
        if  char in 'aeiou':
            vowels = vowels + 1
        else:
            consonants = consonants + 1

print("모음의 개수", vowels)
print("자음의 개수", consonants)
```

```
문자열을 입력하시오: iokkk
모음의 개수 2
자음의 개수 3
```