



4장. 반 복



2가지의 반복 구조

- **for** 문 – 정해진 횟수만큼 반복하는 구조이다.
- **while** 문 – 어떤 조건이 만족되는 동안, 반복을 계속하는 구조이다.

for 문

전체적인 구조



for 변수

in

시퀀스

반복 문장

반복 문장

각 반복마다 변수의 값이 컨테이너의 요소값으로 설정된다.

리스트처럼 요소들을 가지고 있는 객체이다.

블록으로 들여쓰기 하여야 한다.

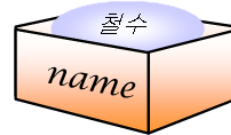
리스트에 대한 반복

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

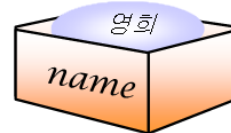
안녕! 철수
안녕! 영희
안녕! 길동
안녕! 유신

리스트 반복 이해하기

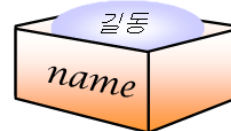
```
for name in ["철수", "영희", "길동", "유신"]:  
    print("안녕! " + name)
```



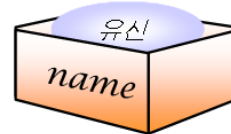
```
for name in ["철수", "영희", "길동", "유신"]:  
    print("안녕! " + name)
```



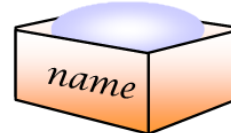
```
for name in ["철수", "영희", "길동", "유신"]:  
    print("안녕! " + name)
```



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



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

예제

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

45

- 
- `range(start, stop)`와 같이 호출하면 `start`부터 시작하여 `(stop-1)`까지의 정수가 생성된다. 이때 `stop`은 포함되지 않는다.

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

45

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

Lab: 정수들의 합

- 1부터 사용자가 입력한 수 n 까지 더해서 ($1+2+3+\dots+n$)을 계산하는 프로그램을 작성하여 보자. for 문을 사용하면 간명하게 합계를 구할 수 있다.

어디까지 계산할까요: 10

1부터 10까지의 정수의 합= 55

Solution

```
# 반복을 이용한 정수합 프로그램
sum = 0

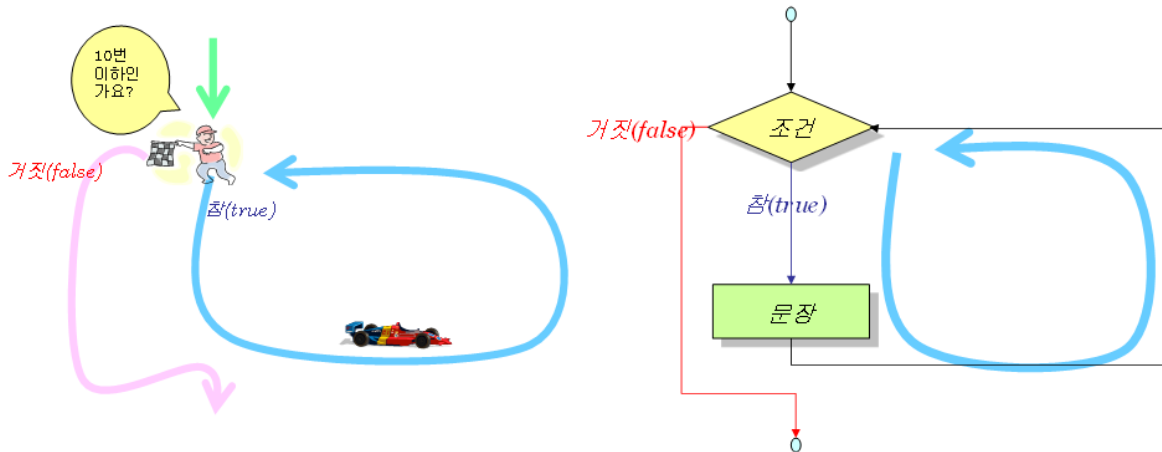
limit = int(input("어디까지 계산할까요: "))
for i in range(1, limit+1):
    sum += i

print("1부터 ", limit, "까지의 정수의 합= ", sum)
```

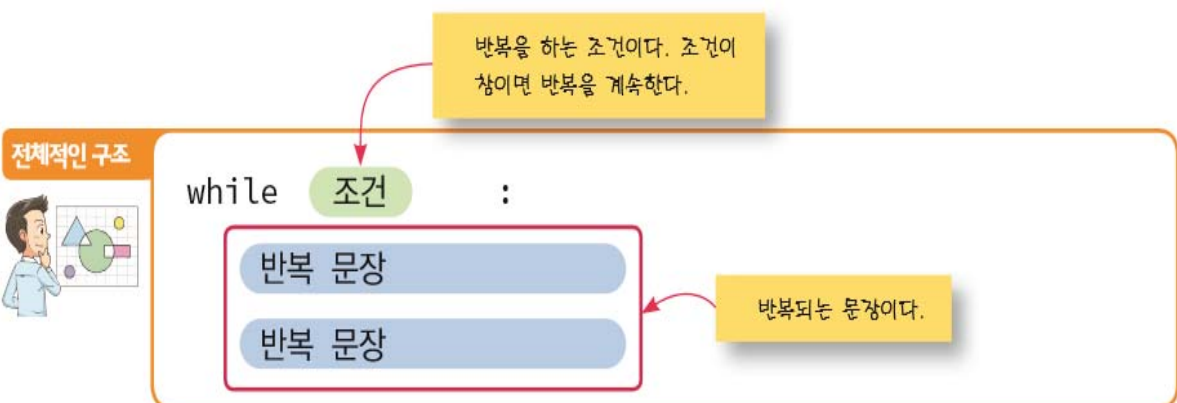
	i의 값	sum의 값
1번째 반복	1	0+1
2번째 반복	2	0+1+2
3번째 반복	3	0+1+2+3
...
10번째 반복	10	0+1+2+3+...+10

while 문

- while 문은 조건을 정해놓고 반복을 하는 구조이다.



while 문의 구조



예제

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

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

Lab: 구구단 출력

- 구구단 중에서 3단을 반복문을 이용하여 출력하여 보자.
3*1, 3*2, 3*3, ..., 3*9까지 9번 반복시키면 출력하면 될 것이다.

3*1 = 3
3*2 = 6
3*3 = 9
3*4 = 12
3*5 = 15
3*6 = 18
3*7 = 21
3*8 = 24
3*9 = 27

Solution

```
i = 1
```

```
while i <= 9:
```

```
    print("3*%d = %d" % (i, 3*i))
```

```
    i = i + 1
```

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 프로그램을 작성하여 보자. 센티널로 는 음수의 값을 사용하자.

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

Solution

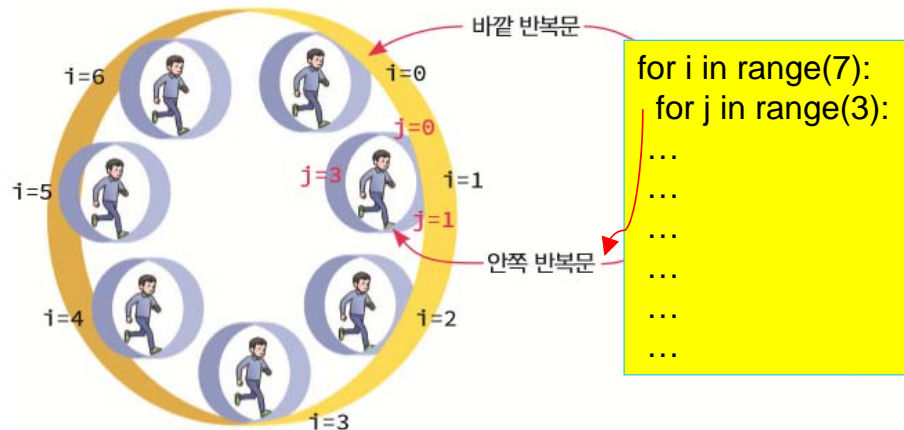
```
# while 문을 이용한 성적의 평균 구하기 프로그램
# 필요한 변수들을 초기화한다.
n = 0
sum = 0
score = 0

print("종료하려면 음수를 입력하시오")
# grade가 0 이상이면 반복
# 성적을 입력받아서 합계를 구하고 학생 수를 센다.
while score >= 0:
    score = int(input("성적을 입력하시오: "))
    if score > 0:
        sum = sum + score
        n = n + 1

# 평균을 계산하고 화면에 출력한다.
if n > 0:
    average = sum / n
print("성적의 평균은 %f입니다." % average)
```

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

Lab: 문자열 조사

- 문자열을 조사하여서 알파벳 문자의 개수, 숫자의 개수, 스페이스의 개수를 출력하는 프로그램을 작성하라.

문자열을 입력하시오: Meav-01-I Dreamt I Dwelt In Marble Halls-192k.mp3
알파벳 문자의 개수= 33
숫자 문자의 개수= 6
스페이스 문자의 개수= 6

Solution

```
statement = input("문자열을 입력하시오: ")
```

```
alphas = 0
```

```
digits = 0
```

```
spaces = 0
```

```
for c in statement:
```

```
    if c.isalpha():
```

```
        alphas = alphas + 1
```

```
    if c.isdigit():
```

```
        digits = digits + 1
```

```
    if c.isspace():
```

```
        spaces = spaces + 1
```

```
print ("알파벳 문자의 개수=", alphas)
```

```
print ("숫자 문자의 개수=", digits)
```

```
print ("스페이스 문자의 개수=", spaces)
```

핵심 정리

- 반복문에는 for 문과 while 문이 있다.
- for 문은 리스트에서 한 항목씩 가져와서 처리한다.
range() 함수를 이용하면 정수들의 리스트를 생성할 수 있다.
- while 문은 조건이 만족되는 동안, 반복을 계속한다.