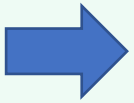
제5장
반복문

1. 반복문(iteration)의 필요성

1) 반복문의 필요성

- 인간은 항상 새롭고 흥미로운 것을 좋아하는 경향이 많아서, 똑같은 작업을 반복하는 것을 지루해한다. 어찌 보면 당연한 일이다. 예를 들어 공장에서 똑같은 공정에서 매일 똑같은 일만 반복한다면 그건 정말 지루하기 짝이 없을 것이다. 하여, 이런 경우에 프로그래밍을 통하여 자동화를 시키면 컴퓨터를 통하여 인간은 그 반복적인 작업에서 벗어날 수 있을 것이다.
- 이처럼 동일한 명령어를 여러 번 실행하는 것을 반복이라고 한다. 반복은 아주 자주 나타내기 때문에 파이썬은 반복을 쉽게 할 수 있는 구조를 가지고 있다. for문과 while문이 바로 반복문인 것이다.

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



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

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

좌측 for문에서 range()함수는 정수들의 리스트를 생성한다.
예를 들면 range(5)는 0,1,2,3,
4까지의 정수를 생성하여서 반환한다.
하여, for문은 리스트의 첫 번째 숫자인 0부터 4까지 반복하고 종료가 된다.

- 위와 같이 “환영합니다 ” 라는 문자열을 출력하고 싶다면 지금까지 배운 것으로는 좌측과 같이 작성을 해야 하지만 반복문을 이용하면 우측과 같이 단 2줄이면 가능한 것이다.

1. 반복문(iteration)의 필요성

1) 반복문의 필요성

- 파이썬에서 지원하는 반복문에는 2가지 종류가 있다. 개발자가 현재 상황에 맞는 구조를 선택하여 사용하면 될 것이다.

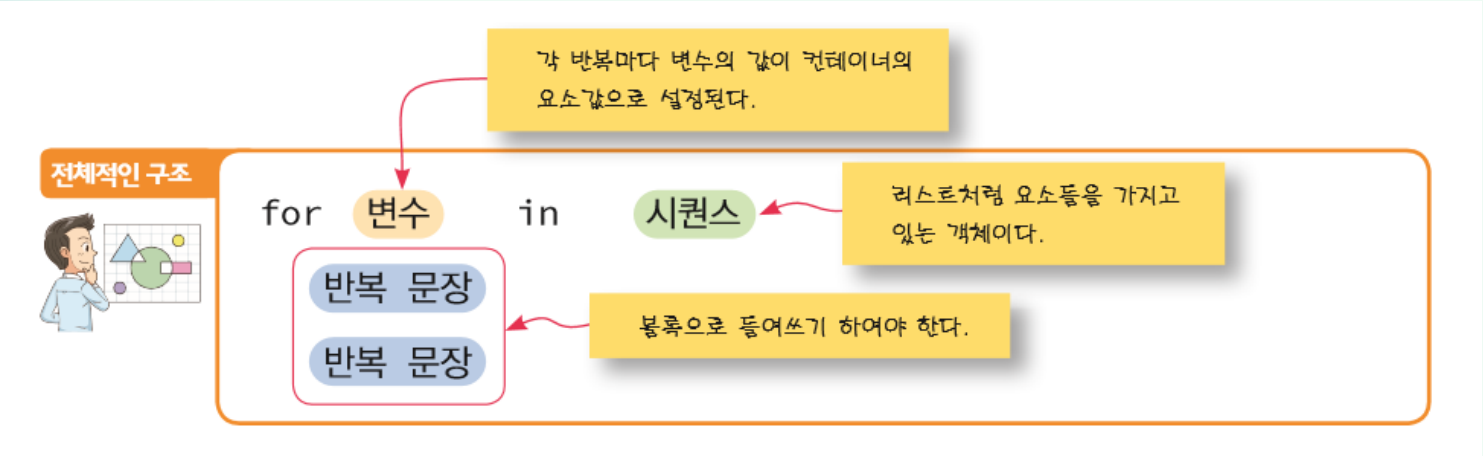
① for문 : 정해진 횟수만큼 반복하는 구조이다.

② while문 : 어떤 조건이 만족되는 동안, 반복을 계속하는 구조이다.(통상, 무한루프용으로 많이 사용된다.)

2. for문

1) for문

- for 문은 정해진 횟수만큼 반복할 때 사용하는 반복 구조이다. for 루프(loop)라고도 한다. for 문은 반복 구조 중에서 가장 많이 사용하는데 그것은 당연히 장점이 많기 때문이다.



2) 리스트에 대한 반복

- 시퀀스는 어떻게 만드는지 확인해보자. 문자열이나 리스트가 바로 시퀀스이다. 리스트에 대하여 반복하는 아래 예제를 한번 보도록 한다.

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

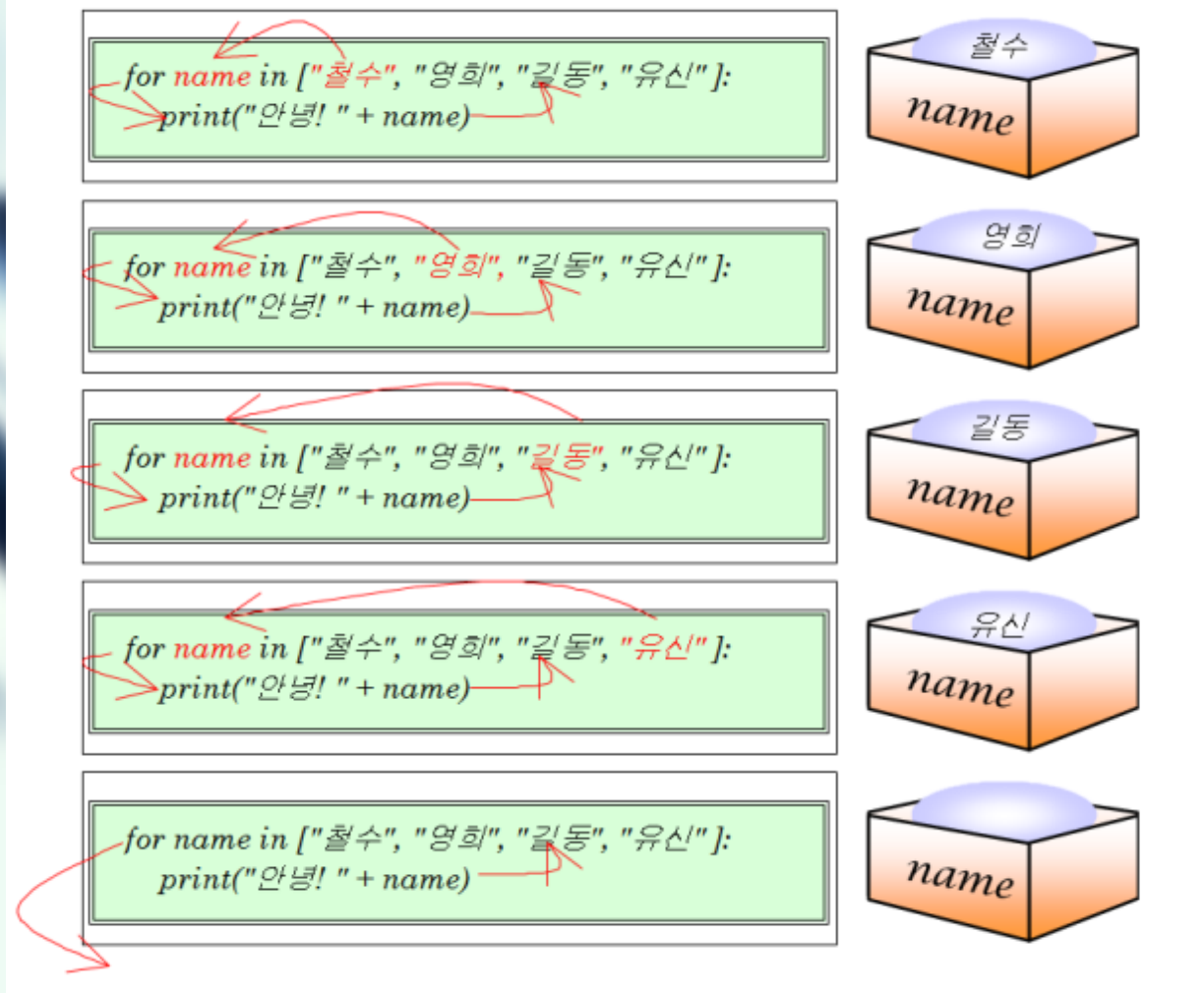
```
안녕! 철수
안녕! 영희
안녕! 길동
안녕! 유신
```

for 문장에 있는 변수 name은 루프 변수라고 칭한다. 변수 이름은 얼마든지 변경 가능하다.역시 루프 코드도 들여쓰기 규칙이 적용된다.
한 번 반복이 끝나면 처리할 항목이 더 있는지 검사한 후 처리할 게 없다면 루프는 종료된다.

2. for문

2) 리스트에 대한 반복

- 앞의 예제를 그림으로 나타내면 아래와 같다.



2. for문

3) 정수 리스트에 대한 반복

- 이번에는 정수 리스트에서 정수를 하나씩 꺼내서 출력하는 코드를 살펴보자.

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

출력 결과
0 1 2 3 4 5 6 7 8 9

- 위의 코드는 리스트에 있는 정수들을 for 문이 실행되면서 이들 리스트에서 정수를 하나씩 꺼내서 변수 x에 할당하며, 각 반복에서 변수 x의 값을 출력한다. 단, 여기서 print(x, end=“ “) 은 변수 x의 값을 출력한 후 줄을 바꾸지 말고 출력해라 라는 의미로 해석하면 된다.

4) range(x) 함수

- 0부터 9까지의 정수들의 리스트는 불편하지만 만들 수 있다. 숫자 10개면 손으로 쓸 수 있지만 숫자가 100만개면 어떻게 할까? 골치가 아플 것이다. 이 때 좀더 편리한 방법으로 유용한 함수가 바로 range() 함수이다.
- range(x) 함수를 이용하면 특정 구간의 정수들을 생성할 수 있다. 예를 들면 range(10)하면 0 ~ 9까지의 정수가 생성된다.

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

출력 결과
45

2. for문

5) range(start, stop) 함수

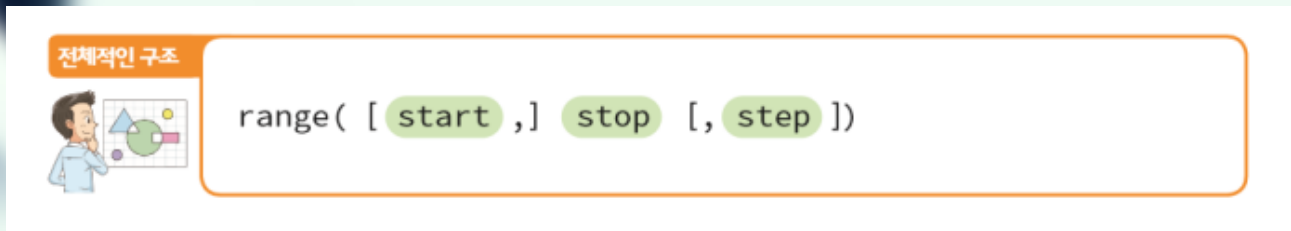
- 아울러 range(start, stop)의 매개변수 2개를 가지는 함수도 사용할 수가 있다. 이 함수를 호출하면 start부터 시작하여 (stop)-1까지의 정수가 생성된다.이 때 stop은 포함되지 않는다.

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

```
출력 결과
45
```

6) range(start, stop, step) 함수

- start 부터 (stop-1)까지 step의 간격으로 정수를 생성한다. 예를 들어서 range(0, 10, 2)와 같이 호출되면 0부터 2씩 건너뛰면서 10보다는 작은 정수 [0, 2, 4, 6, 8]가 생성된다.



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

  * 아직은 초보적인 수준이라 위의 내용을 이해하는 것이 중요하다. 하지만 추후 중급 수준으로 가게 된다면 range() 함수가 연속적인 값들을 생성할 때 메모리의 낭비를 막기 위해서 파이썬 버전 3부터는 제너레이터라는 객체를 리턴한다. 이 부분은 뒤에서 나올 것이니 한 번 들어봤다 정도로 일단 가볍게 넘어가도록 하자.

2. for문

7) 문자열 반복

- 문자열도 시퀀스의 일부분이다. 따라서 문자열을 대상으로 반복문은 얼마든지 만들 수 있다. 문자열 “abcdef”에 들어 있는 문자들을 처리하려면 아래와 같이 반복문을 만들어서 사용하면 될 것이다.

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

출력 결과
a b c d e f

8) 반복이 끝나면 어떻게 될까?

- 반복문이 끝나면 당연한 말이겠지만 반복문 아래에 있는 문장이 실행된다.

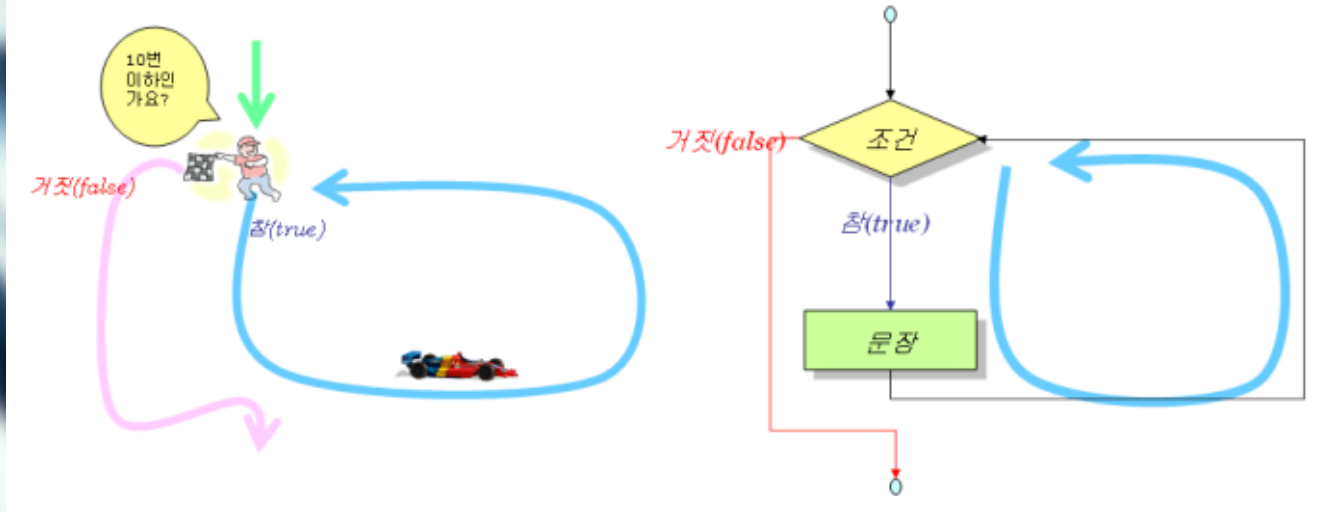
```
for x in range(0, 5):
        print(x, end=" ")

print(“반복 종료“)
```

출력 결과
0 1 2 3 4 반복 종료

3. while문

1) while문의 개요

- while문은 어떤 조건을 정해 놓고 반복을 하는 구조를 말한다. 예를 들어, 자동차 경주에서 반드시 트랙을 10번 돌아야 한다면 반복하는 조건은 “횟수가 10번 미만인가요?” 가 될 것이다. 반복을 결정하는 조건이 있고 조건이 참이면 반복을 하고 그렇치 않으면 반복 루프를 빠져나가게 된다.



2) while문의 구조

- while문은 구조는 while문의 옆에 반복의 조건을 기술한다. 조건이 만족되는 동안, 블록 안의 문장이 반복 실행된다.

3. while문

2) while문의 구조

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

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

- 하지만 위의 코드에서 i의 증가가 없다면 어떻게 될까? 그렇다. 그냥 계속 루프를 돌 것이다. 그 이유는 바로 조건이 계속 참이기 때문인 것이다. 이러한 것을 무한 루프(infinite loop)라고 하면 이런 형태가 나오지 않도록 프로그램을 작성해야 한다.
- 그럼에도 불구하고 우리 주변에는 무한 루프로 되어져 있는 프로그램들이 상당히 많이 존재한다. 여러분들이 사용하는 에어컨, PC, 모니터, TV, 자판기 등등 여러가지가 있다. 근데 여기서 중요한 건 분명히 이러한 물건들은 반드시 무한 루프를 빠져나가는 곳이 존재한다.

  하여, 무한 루프를 프로그램으로 작성할 때는 어떤 조건일 때 이 루프를 빠져나가는지를 반드시 기술해야 한다.

4. 보초값(sentinel) 사용하기

1) 보초값의 개념

- 이제는 반복문을 활용하여 사용자가 입력하는 5개의 정수의 합을 우리는 구할 수 있다. 하지만 만약 입력될 데이터의 정확한 개수가 미리 알려지지 않거나 데이터가 너무 많아서 개수를 알기가 힘든 경우에는 어떻게 할까?
  이런 경우에는 데이터의 끝에다 끝을 알리는 특수한 데이터를 놓으면 될 것이다. 프로그램에서 데이터를 입력하다가 이 특수한 데이터가 나타나면 데이터의 입력을 중단하면 되는 것이다. 데이터의 끝을 알리는데 사용되는 데이터 값을 통상 보초값(sentinel)이라고 칭한다. 보초값은 일반적인 데이터 값에서 절대 등장할 수 없는 값으로 선택하는 것이 좋다. 예를 들어 성적을 입력 받아 성적의 평균을 구하는 프로그램이면, 음수나 100을 초과하는 값을 센티널 즉, 보초값으로 선택하는 것이 좋을 것이다.
- 예를 들어서 사용자로부터 임의의 개수의 성적을 받아서 평균을 계산한 후 출력하는 프로그램을 작성한다면, 센티널로 음수의 값을 사용하면 될 것이며, 사용자가 음수를 입력하는 순간 반복을 중단하면 될 것이다.

4. 보초값(sentinel) 사용하기

2) 보초값의 예제

- 앞서 말한 예제 코드이다. 살펴보도록 하자.

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

print("종료하려면 음수를 입력하시오")
# grade가 0이상이면 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)
```

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

몇 개의 데이터가 입력될 지 모르니 음수 값을 보초값으로 세워 두고 음수가 입력되면 루프를 빠져나오게 한 예제이다.

5. 중첩 루프 사용하기

1) 중첩 루프의 개념

- 반복문은 중첩하여 사용될 수 있다. 즉 반복문 안에 다른 반복문이 포함될 수 있다. 즉, 이러한 형태를 중첩 반복문(nested loop)라고 한다. 외부에 위치하는 반복문을 외부 반복문(outer loop)라고 하며, 안쪽의 반복문을 내부 반복문(inner loop)라고 한다. 중요한 것은 내부 반복문은 외부 반복문이 한번 반복할 때마다 새롭게 실행된다는 점이다.

2) 중첩 반복문의 예제

- 중첩 반복문은 실제 프로그래밍에서 매우 자주 등장한다. 예를 들어서 테이블(표)과 비슷한 데이터를 처리하는데 유용하다. 아래 예제는 * 기호를 사각형 모양으로 출력한 것이다.

```
# 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
```

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

6. 문자열 처리하기

1) 반복문으로 문자열 처리하기

- 문자열을 처리하는 용도로도 반복문이 많이 사용된다. 아래 간단한 예제를 살펴보자.

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

출력 결과
a p p l e

- 문자열도 앞서 강의했듯이 시퀀스의 일종이라는 것이다.

2) 반복문으로 문자열 처리 예제

- 문자열을 받아서 모음을 전부 없애는 코드는 아래와 같이 작성할 수 있다.

```
s = input('문자열을 입력하시오: ')
vowels = "aeiouAEIOU“          # 영문의 모음 리스트를 문자열로 저장
result  = ""
for letter in s:
    if letter not in vowels:          # 문자가 모음리스트에 있지 않으면…
        result += letter
print(result)
```

출력 결과
문자열을 입력하시오: kkkoommm
kkkmmm

6. 문자열 처리하기

2) 반복문으로 문자열 처리 예제
- 문자열 중에서 자음과 모음의 개수를 집계하는 프로그램 예제

```
original = input('문자열을 입력하시오: ')
word = original.lower()                    # 입력 받은 문자열을 소문자로 변경한다.
vowels = 0
consonants = 0

if len(original) > 0 and original.isalpha():    # 문자열의 길이가 0초과이고, 알파벳이라면….
    for char in  word:                          # 각 문자에 대한 반복을 실행
        if  char in 'aeiou’:                    # 모음이라면…
            vowels = vowels + 1                  # vowels변수에 1을 증가한다.
        else:                                   # 모음이 아니라면…
            consonants = consonants + 1      # consonants변수에 1을 증가한다.

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

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

감사합니다.