4장 반복

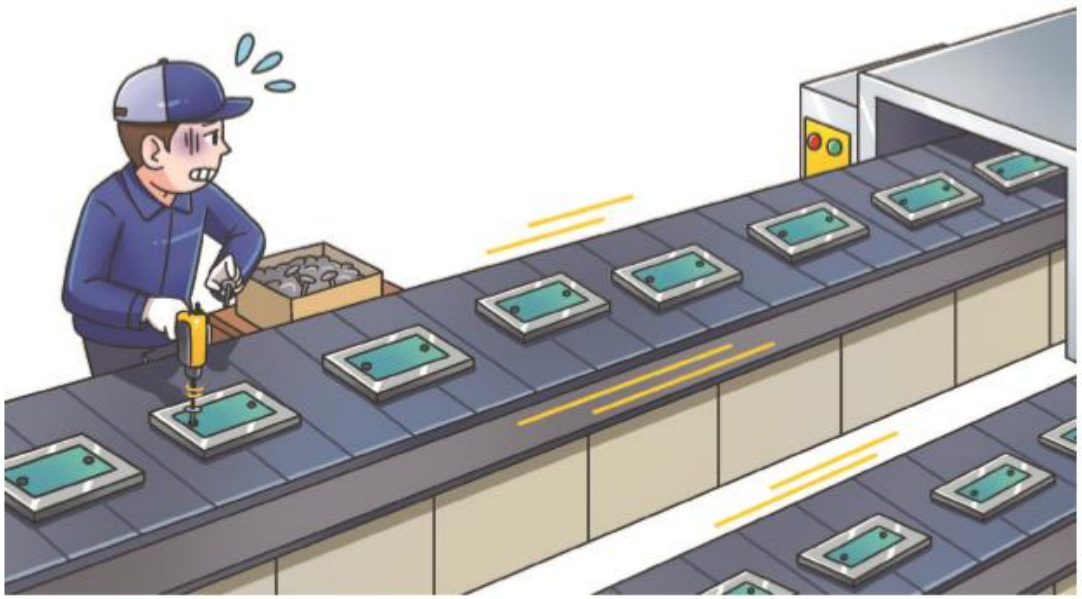
반복의 필요성

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

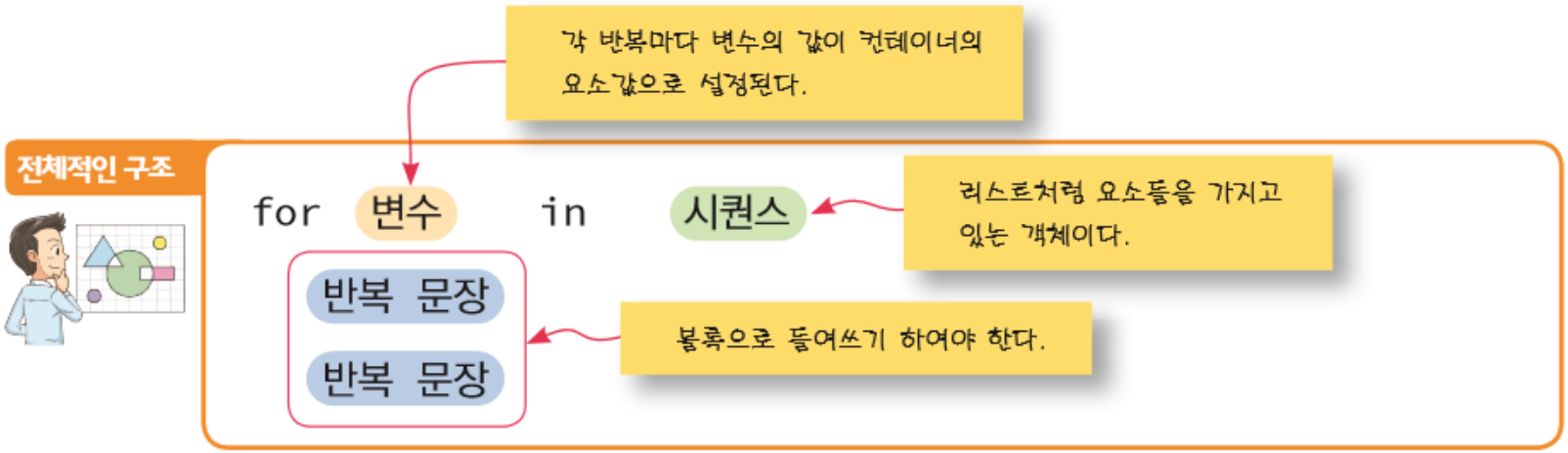
```
for x in range(5) :
        print("환영합니다.")
```

```
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

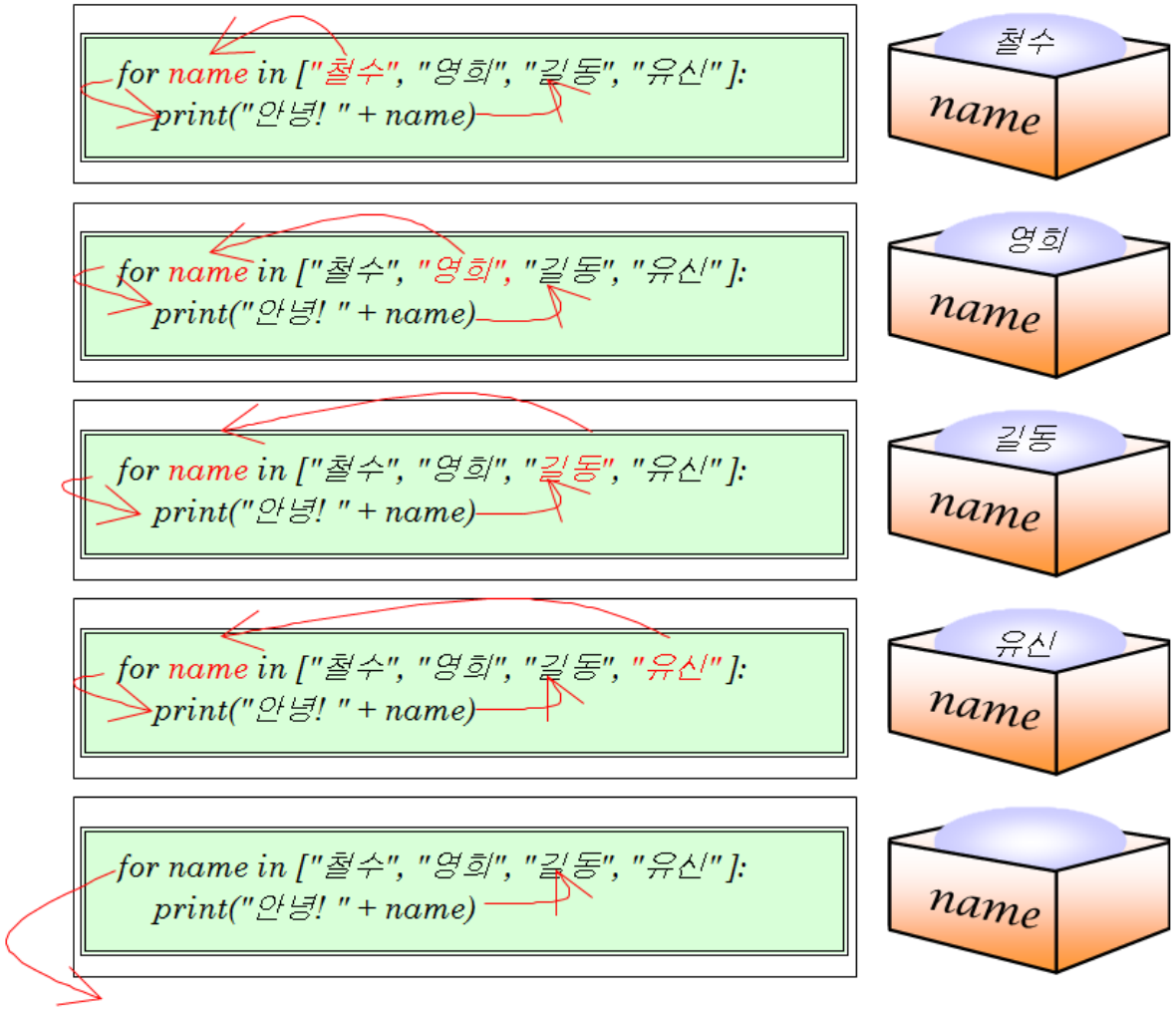

각 반복마다 변수의 값이 컨테이너의 요소값으로 설정된다.
전체적인 구조
for 변수 in 시퀀스
리스트처럼 요소들을 가지고 있는 객체이다.
반복 문장
반복 문장
블록으로 들여쓰기 하여야 한다.

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

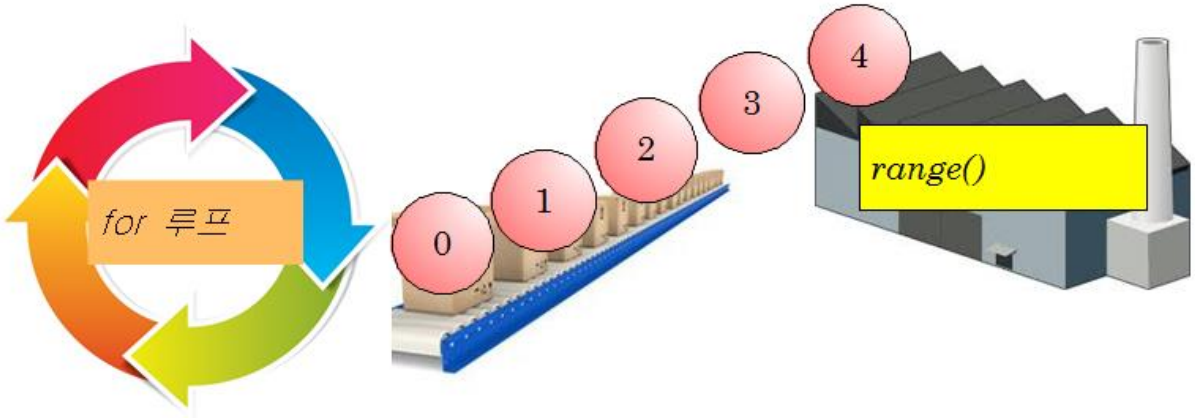
정수 리스트에 대한 반복

```
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 **0**부터 **9**까지의 정수가 생성된다.



예제

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

```
45
```

- **range(start, stop)**와 같이 호출하면 **start**부터 시작하여서 **(stop-1)**까지의 정수가 생성된다. 이때 **stop**은 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 **start**부터 **stop-1**까지 **step**의 간격으로 정수들을 생성한다. **start**와 **step**이 대괄호로 싸여져 있는데 이는 생략할 수 있다는 의미이다. **start**나 **step**이 생략되면 **start**는 **0, step**은 **1**로 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
```

Lab: 정수들의 합

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

```
어디까지 계산할까요: 10
1부터 10 까지의 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

Lab: 팩토리얼 계산

- **for**문을 이용하여서 팩토리얼을 계산해보자. 팩토리얼 **n!**은 **1**부터 **n**까지의 정수를 모두 곱한 것을 의미한다. 즉, **n! = 1×2×3×……×(n-1)×n**이다.

```
정수를 입력하시오: 10
10 !은 3628800.0 이다.
```

Solution

```
# 반복을 이용한 팩토리얼 구하기
fact=1.0
n = int(input("정수를 입력하시오:  "))

for i in range(1, n+1) :
        fact = fact * i;

print(n,"!은",  fact, "이다.")
```

Lab: 온도 변환 테이블 출력

- 화씨온도-섭씨온도 변환 테이블을 출력하는 프로그램을 작성하여 보자. 반복 구조를 사용하여야 하고 정수보다는 실수로 출력하는 편이 정확하다. 화씨 **0**도부터 **100**도까지, **10**도 단위로 증가시키면서 대응되는 섭씨온도를 옆에 출력한다. **C = (F-32)×5/9** 수식을 사용한다.

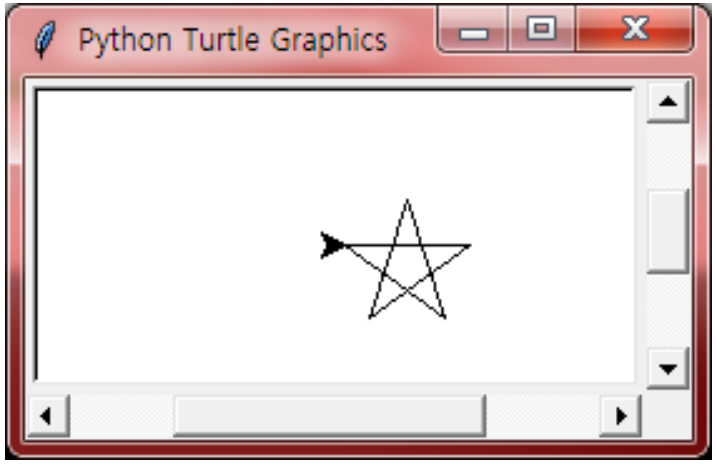
```
0  -> -17.78
10  -> -12.22
20  -> -6.67
30  -> -1.11
40  -> 4.44
50  -> 10.0
60  -> 15.56
70  -> 21.11
80  -> 26.67
90  -> 32.22
100  -> 37.78
```

Solution

```
for t in range(0, 100+1, 10):
    c = (t - 32.0) * 5.0 / 9.0
    print(t, " ->", round(c, 2));
```

Lab: 화면에 별 그리기

- 터틀 그래픽을 이용하여 별을 화면에 그려보자.

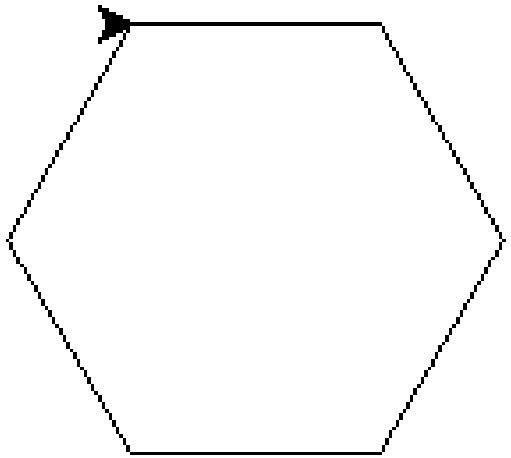


Solution

```
import turtle

star = turtle.Turtle()

for i in range(5):
    star.forward(50)
    star.right(144)
```

Lab: 화면에 다각형 그리기

- 터틀 그래픽을 이용하여 다각형을 화면에 그려보자.

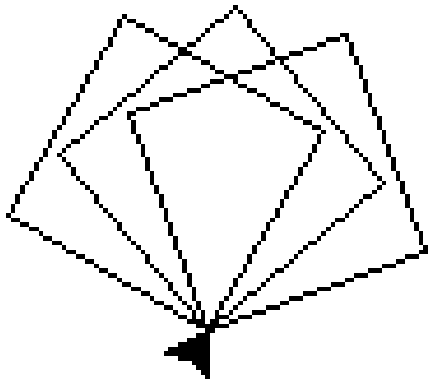


Solution

```
import turtle

polygon = turtle.Turtle()

num_sides = 6
side_length = 70
angle = 360.0 / num_sides

for i in range(num_sides):
    polygon.forward(side_length)
    polygon.right(angle)
```

Lab: 화면에 사각형 그리기

- 터틀 그래픽을 이용하여 사각형을 3개 그려보자. 각 사각형은 20도씩 기울어져 있다.

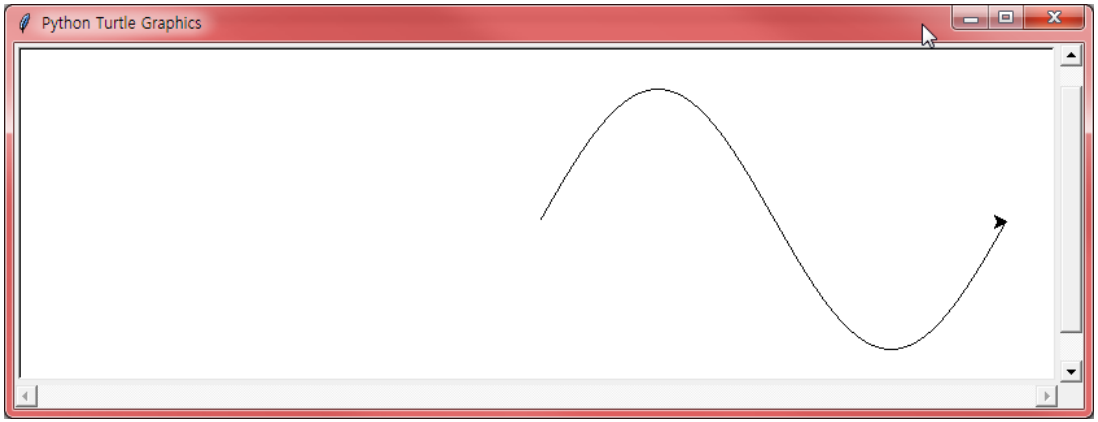


Solution

```
import turtle

for i in range(3):          # 3번 반복한다.
    turtle.left(20)         # 왼쪽으로 20도 회전한다.

    turtle.forward(50)      # 50픽셀 전진한다.
    turtle.left(90)
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(50)
    turtle.left(90)
```

Lab: 함수 그래프 그리기

- 싸인(sine) 그래프를 반복문을 이용하여서 그려보자. 싸인 함수는 수학, 물리학, 공학에서 아주 많이 나타나는 함수이다. 이번에도 터틀 그래픽의 기능을 사용하여 본다.

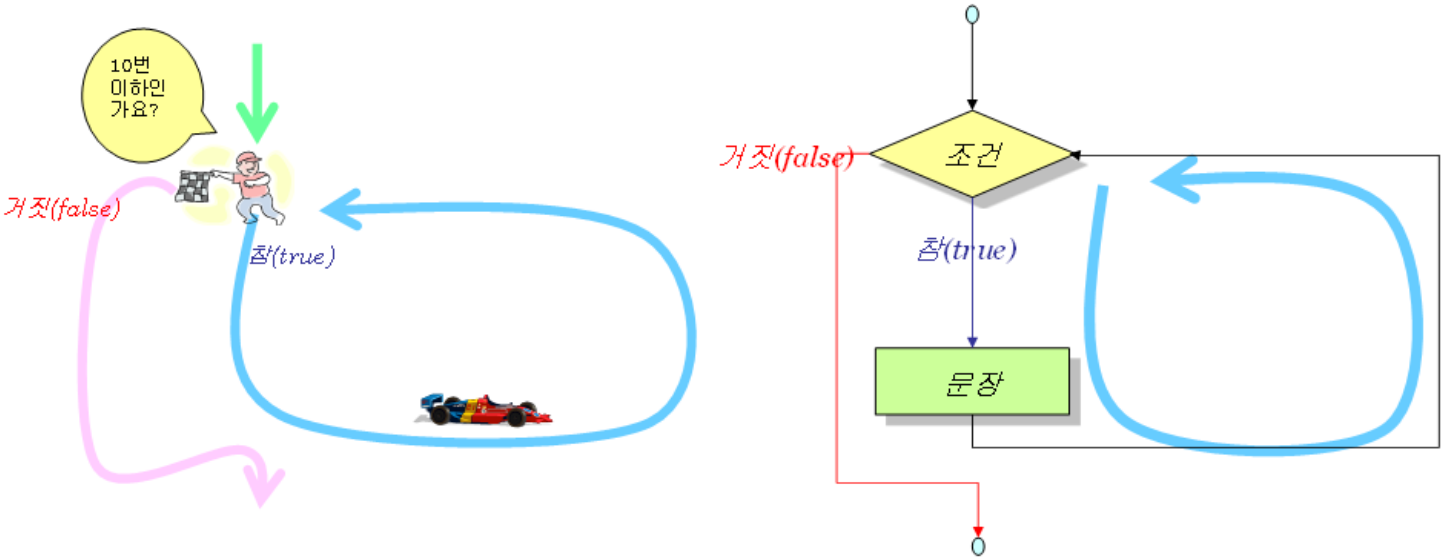


Solution

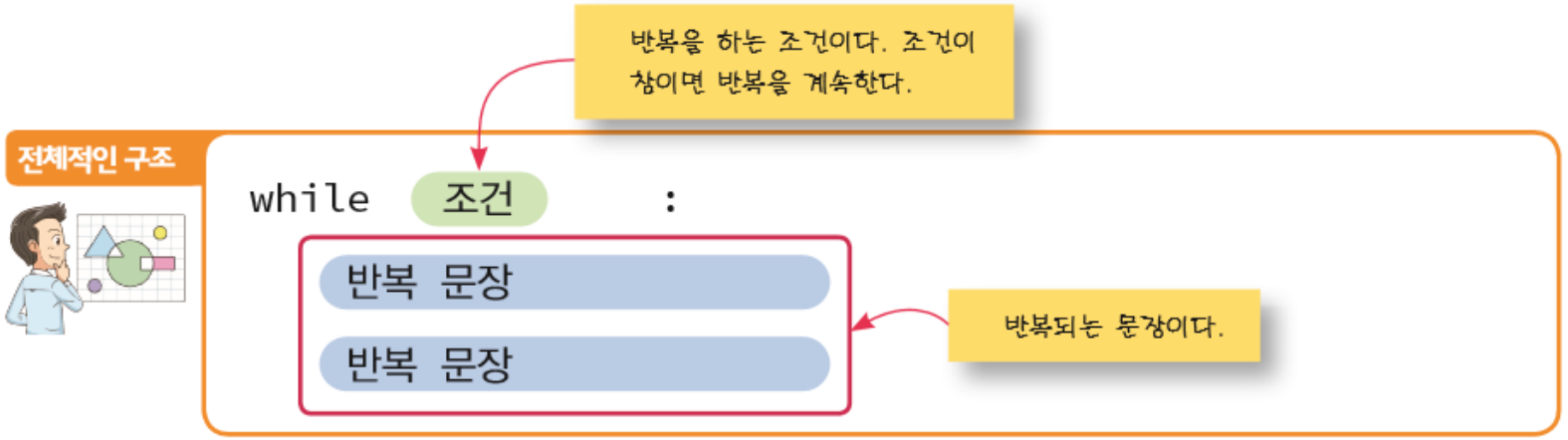
```
import math
import turtle

t = turtle.Turtle()

t.pendown()                          # 터틀 객체의 펜을 내린다.
for angle in range(360):             # 360번 반복한다.
    y = math.sin(math.radians(angle)) # angle 값에 대응되는 싸인값을 계산한다.
    scaledX = angle                  # x축의 좌표값을 각도로 한다.
    scaledY = y * 100                # y축의 좌표값을 싸인값으로 한다.
    t.goto (scaledX, scaledY) # 터틀 객체를 (scaledX, scaledY)로 이동시킨다.
t.penup()                            # 터틀 객체의 펜을 올린다.
```

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.
```

Lab: 함수 그래프 그리기

- 0, 1, 2, .,,, 9까지를 차례대로 화면에 출력하는 프로그램을 작성하여 보자. 변수 i의 값을 0으로 초기화하고 반복하면서 i를 출력하고 1씩 증가시키면 된다. i가 10보다 작을 때까지 반복시키면 된다.

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

Solution

```
i = 0
while i < 10:
    print (i, end=" ");
    i = i + 1
```

Lab: (1+2+3+...+9+10) 계산하기

- **(1+2+3+...+9+10)**의 값을 계산하는 프로그램을 작성하여 보자. 이것은 공식으로도 계산할 수 있으나 우리는 반복 구조를 사용해보자.

합계= 55

Solution

```
i = 1
sum = 0;

while i <= 10:
        sum = sum + i
        i = i + 1
print("합계=", sum)
```

Lab: 팩토리얼 계산

- 팩토리얼을 계산하는 프로그램을 작성하여 보자. 팩토리얼 **n!**은 **1**부터 **n**까지의 정수를 모두 곱한 것을 의미한다. 즉, **n! = 1×2×3×……×(n-1)×n**이다. 예를 들어서 **10!**을 계산하는 프로그램을 작성하여 보자.

10!은 3628800입니다.

Solution

```
i = 1
factorial = 1

while (i <= 10):
        factorial = factorial * i
        i = i + 1
print ("10!은 %d입니다." % factorial)
```

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
```

Lab: 배수의 합 계산 프로그램

- 1부터 100사이의 모든 3의 배수의 합을 계산하여 출력하는 프로그램을 반복 구조를 사용하여 작성하라.

1 부터 100 사이의 모든 3 의 배수의 합은 1683 입니다.

Solution

```
sum = 0
number = 1

while number <= 100  :
      if number %3 == 0 :
            sum = sum + number
      number = number + 1
print("1부터 100 사이의 모든 3의 배수의 합은 %d입니다." % sum)
```

Lab: 자리수의 합

- 정수 안의 각 자리수의 합을 계산하는 프로그램을 작성해 보자. 예를 들어서 **1234**라면 **(1+2+3+4)**를 계산하는 것이다.

자리수의 합은 10입니다.

Solution

```
number = 1234
sum = 0;
while number > 0 :
    digit = number % 10
    sum = sum + digit
    number = number // 10
print("자리수의 합은 %d입니다." % sum)
```

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 끝에 다 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.
```

Solution

```
# 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)
```

Lab: 숫자 맞추기 게임

- 앞에 등장하였던 숫자 맞추기 게임 업그레이드

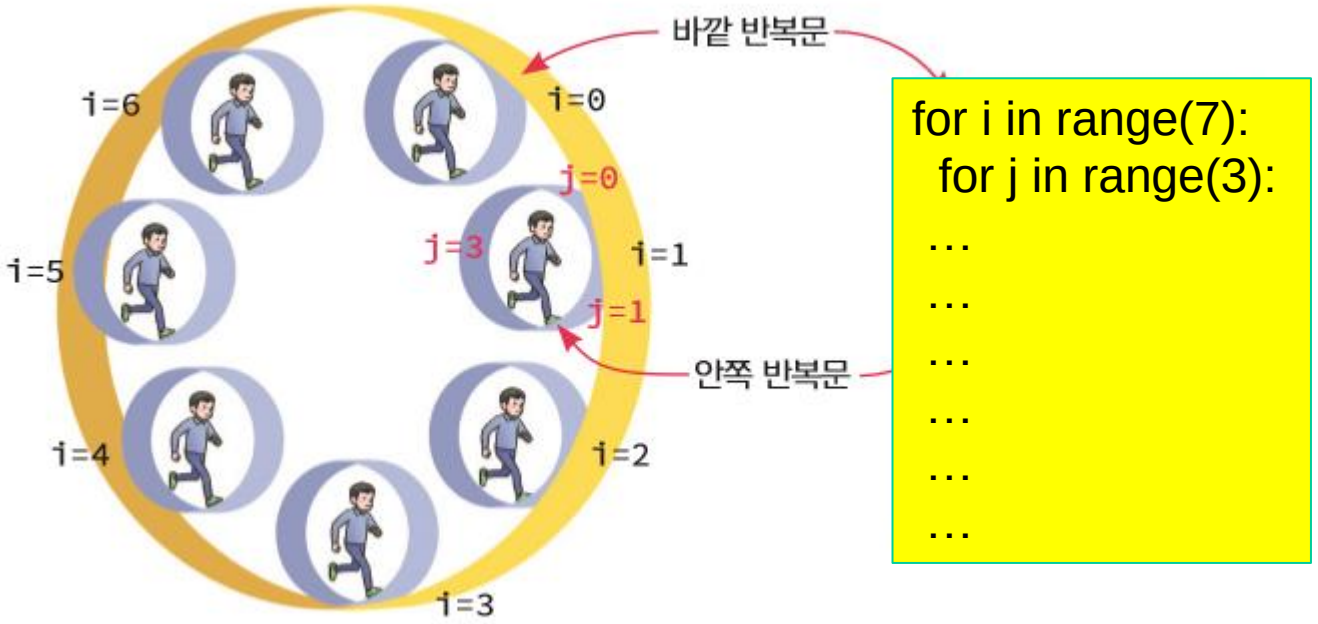
```
1부터 100 사이의 숫자를 맞추시오
숫자를 입력하시오: 50
낮음!
숫자를 입력하시오: 75
낮음!
숫자를 입력하시오: 82
낮음!
숫자를 입력하시오: 91
높음!
숫자를 입력하시오: 86
낮음!
숫자를 입력하시오: 87
축하합니다. 시도횟수= 6
```

Solution

```
import random

tries = 0
number = random.randint(1, 100)

print("1부터 100 사이의 숫자를 맞추시오")

while tries < 10:
    guess = int(input("숫자를 입력하시오: "))
    tries = tries + 1
    if guess < number:
        print("낮음!")
    elif guess > number:
        print("높음!")
    else:
        break

if guess == number:
    print("축하합니다. 시도횟수=", tries)
else:
    print("정답은 ", number)
```

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
```

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

Lab: 피타고라스 삼각형 찾기

- 피타고라스의 정리는 직각 삼각형에서 직각을 낀 두 변의 길이를 **a, b**라고 하고, 빗변의 길이를 **c**라고 하면 의 수식이 성립한다는 것이다. 각 변의 길이가 **100**보다 작은 삼각형 중에서 피타고라스의 정리가 성립하는 직각 삼각형은 몇 개나 있을까?

```
3 4 5
4 3 5
5 12 13
...
```

Solution

```
for a in range(1, 101, 1):
    for b in range(1, 101, 1):
        for c in range(1, 101, 1):
            if( (a*a+b*b)==c*c ):
                print(a, b, c)
```

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)
```

Lab: 계좌번호 처리

- 인터넷 뱅킹을 사용하다보면 계좌번호를 입력할 때, **"312-02-1234567"**과 같이 "-"을 사용하면 안 된다는 경고를 받는다. 사용자로부터 "-"가 포함된 계좌 번호를 받아서 "-"을 삭제한 문자열을 만들어보자.

계좌번호를 입력하시오: 312-02-1234567
312021234567

Solution

```
raw = input('계좌번호를  입력하시오: ')
processed = ""

for c in raw:
    if c != "-":
        processed = processed + c
print(processed)
```

헥심 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.

Q & A