Chapter 03
조건문 및 반복문

컴퓨팅 사고력을 키우는 SW 교육

파이썬

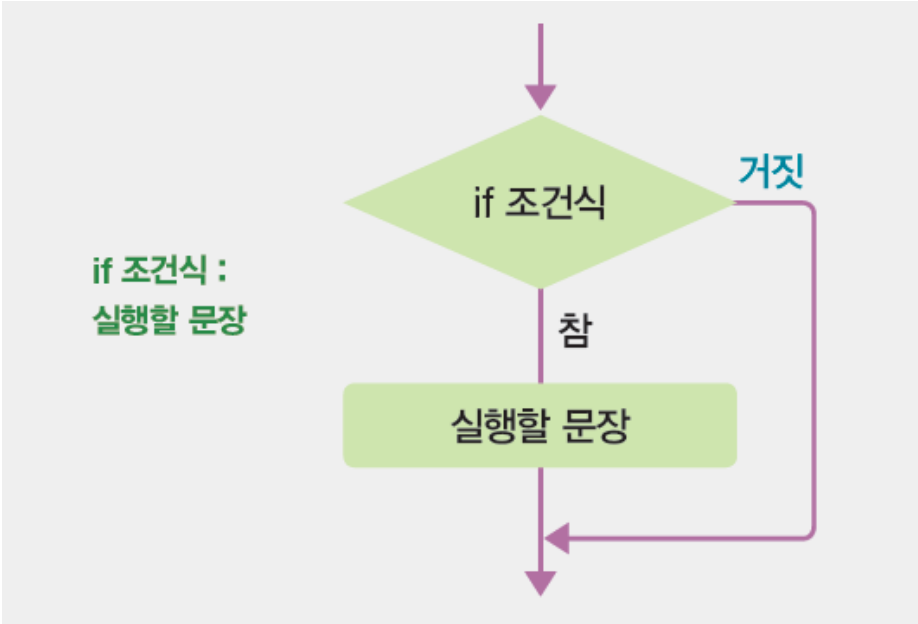
Section 01 if문

if문 (1)

기본 if문

- if 조건식 : 에서 조건식이 참이면 실행할 문장이 처리되고, 거짓이면 아무것도 실행하지 않고 프로그램을 종료

그림 5-3
if문의 형식과 순서도




```
a=99
if a<100 :
   print("100보다 작군요.")
```

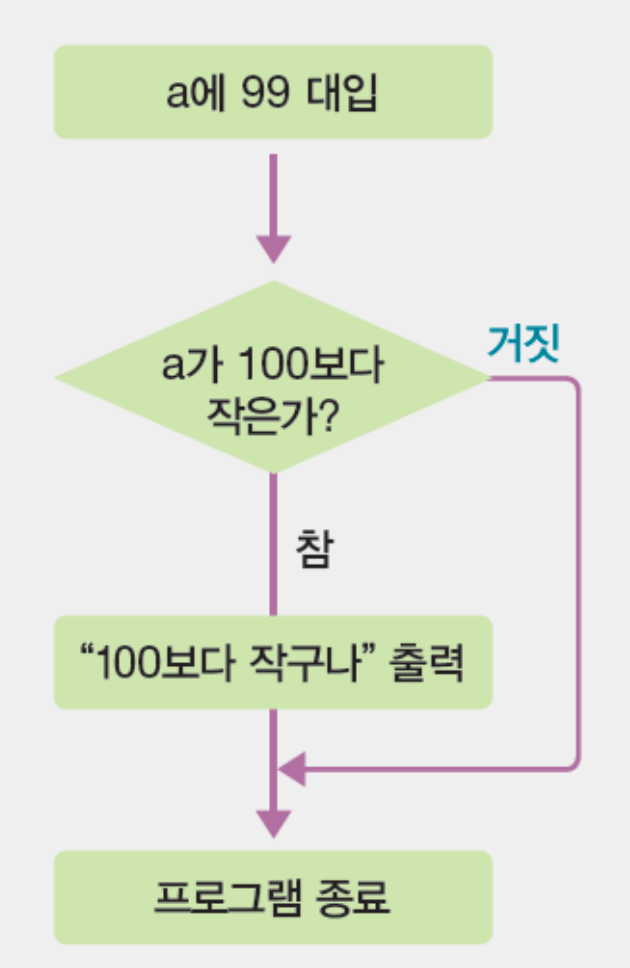
출력 결과

```
100보다 작군요.
```

if문 (2)

- **실행 과정**

그림 5-4
if문의 실행 과정



if문 (3)

소스코드 5-1
(파일명 : 05-01.py)

```
1  a=200
2
3  if a<100 :
4      print("100보다 작군요")
5  print("거짓이므로 이 문장은 안 보이겠죠?")
6
7  print("프로그램 끝")
```

출력 결과

```
거짓이므로 이 문장은 안 보이겠죠?
프로그램 끝
```

- 들여 쓰기를 하지 않아 실행 하지 않아야 할 5행까지 실행 됨.

  다음과 같이 줄 바꿈을 수정하여 실행

if문 (4)

소스코드 5-2
(파일명 : 05-02.py)

```
1  a=200
2
3  if a<100 :
4      print("100보다 작군요")
5      print("거짓이므로 이 문장은 안보이겠죠?")
6
7  print("프로그램 끝")
```

출력 결과

```
프로그램 끝
```

- 들여쓰기 잘못으로 오류 발생한 경우

```
if  a<100 :
      print("100보다 작군요")
    print("거짓이므로 이 문장은 안보이겠죠?")
```

if문 (5)

- **if~else 문**
  - 참일 때 실행하는 문장과 거짓일 때 실행하는 문장이 다를 때 사용

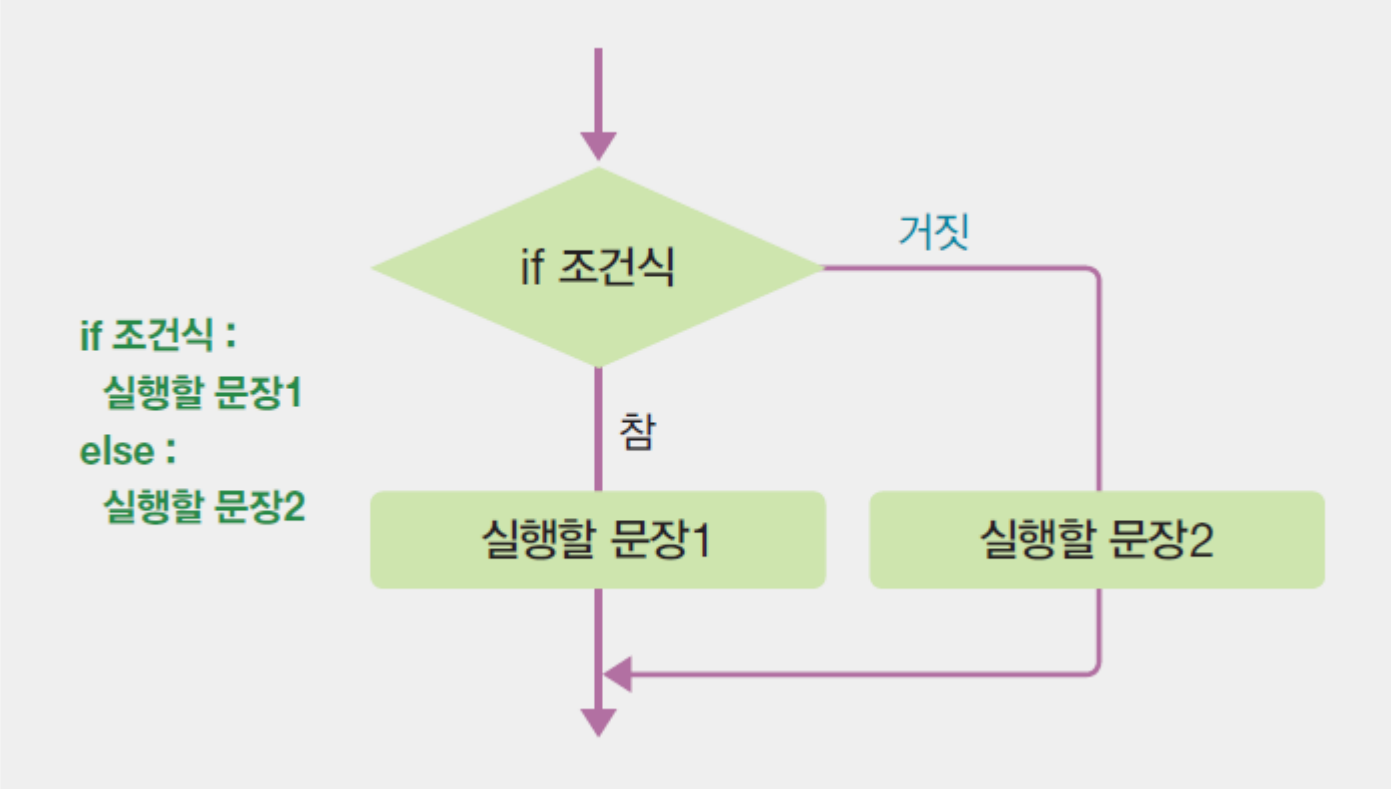


그림 5-5
if~else문의 형식과 순서도

if문 (6)

```
1  a=200
2
3  if a<100 :
4      print("100보다 작군요.")
5  else :
6      print("100보다 크군요.")
```

출력 결과

```
100보다 크군요.
```

소스코드 5-3
(파일명 : 05-03.py)

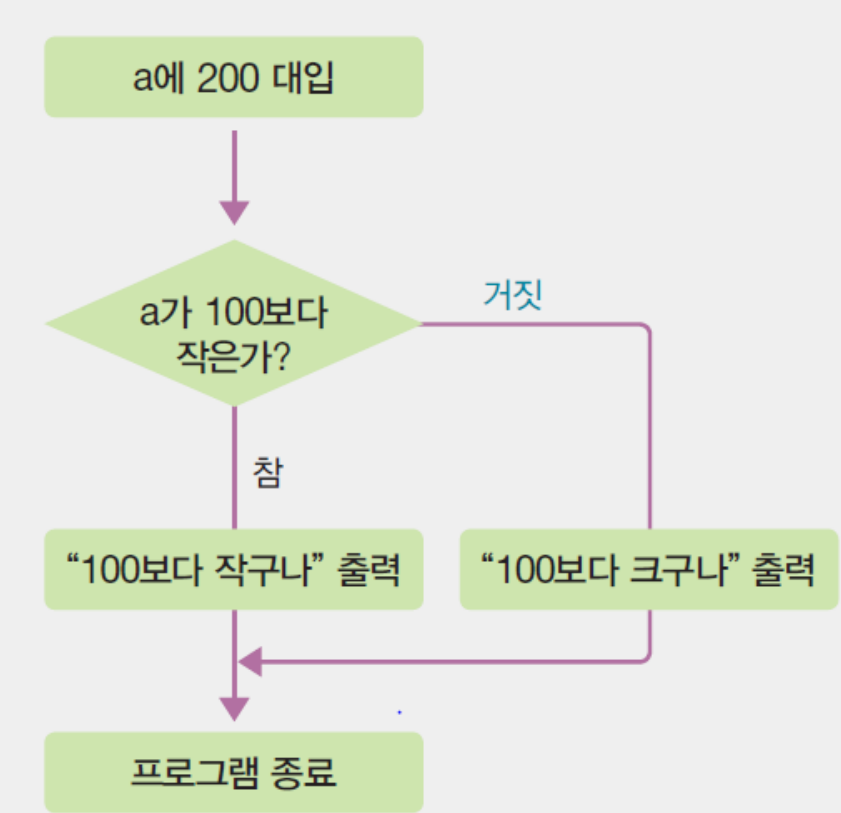


그림 5-6
[소스코드 5-3]의 실행 과정

if문 (7)

소스코드 5-4
(파일명 : 05-04.py)

```
1  a=200
2
3  if a<100 :
4      print("100보다 작군요.")
5      print("참이면 이 문장도 보이겠죠?")
6  else :
7      print("100보다 크군요.")
8      print("거짓이면 이 문장도 보이겠죠?")
9
10 print("프로그램 끝!")
```

출력 결과

```
100보다 크군요.
거짓이면 이 문장도 보이겠죠?
프로그램 끝!
```

if문 (8)

- 입력한 숫자가 짝수인지 홀수인지 계산하는 프로그램

소스코드 5-5
(파일명 : 05-05.py)

```
1  a=int(input("정수를 입력하세요 : "))
2
3  if a%2==0 :
4      print("짝수를 입력했군요.")
5  else :
6      print("홀수를 입력했군요.")
```

출력 결과

```
정수를 입력하세요 : 125  ← 사용자가 입력한 값
홀수를 입력했군요.
```

if문 (9)

- if ~ else ~ if ~ else문

```
if 조건식 1 :
    if 조건식 2 :
        실행할 문장1
    else :
        실행할 문장2
else :
    실행할 문장3
```

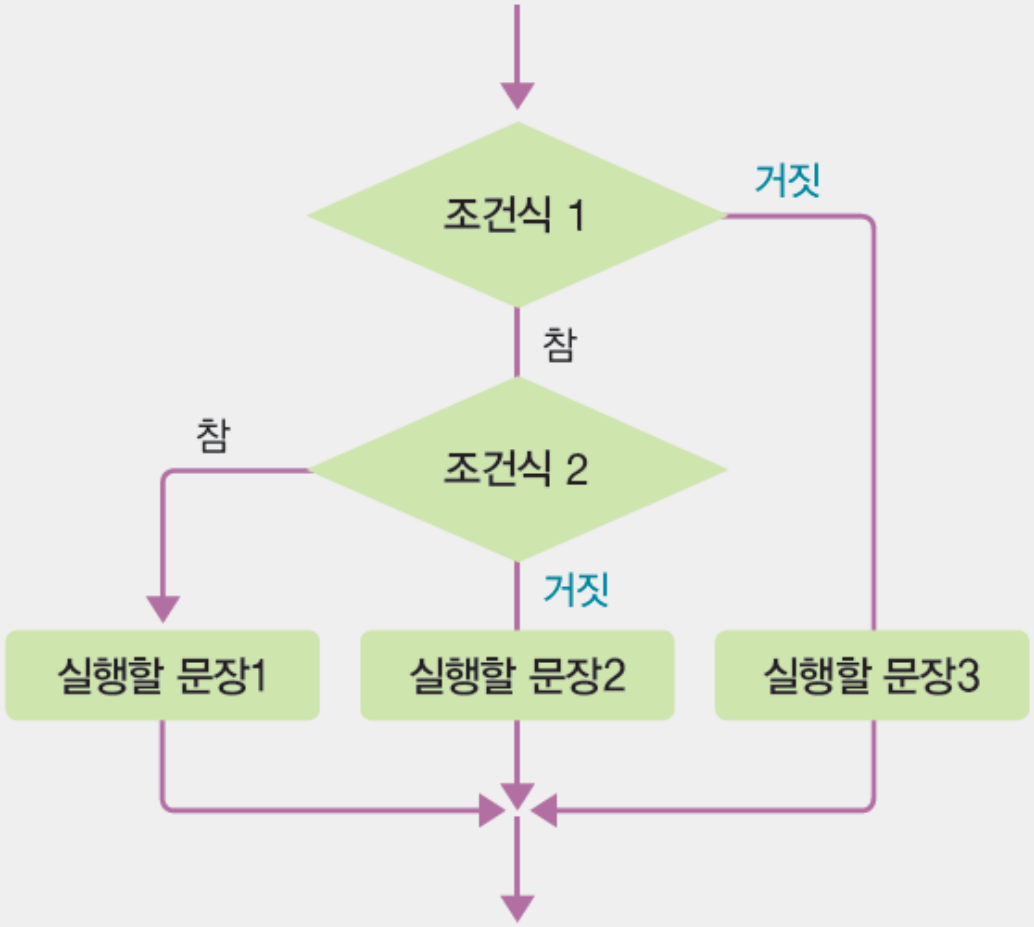


그림 5-7
중첩 if문의 형식과 순서도

if문 (10)

소스코드 5-6
(파일명 : 05-06.py)

```
1  a=75
2
3  if a>50 :
4      if  a<100 :
5          print("50보다 크고 100보다 작군요.")
6      else :
7          print("와~~ 100보다 크군요.")
8  else :
9          print("에고~ 50보다 작군요..")
```

출력 결과

```
50보다 크고 100보다 작군요.
```

- 3행에서 a가 50보다 크면 참이므로 들여쓰기가 된 부분(4행~7행)의 내용을 실행
  그 안에서 a가 100보다 작아서 5행을 출력

if문 (11)

- 중첩 if문의 실제 사례

소스코드 5-7
(파일명 : 05-07.py)

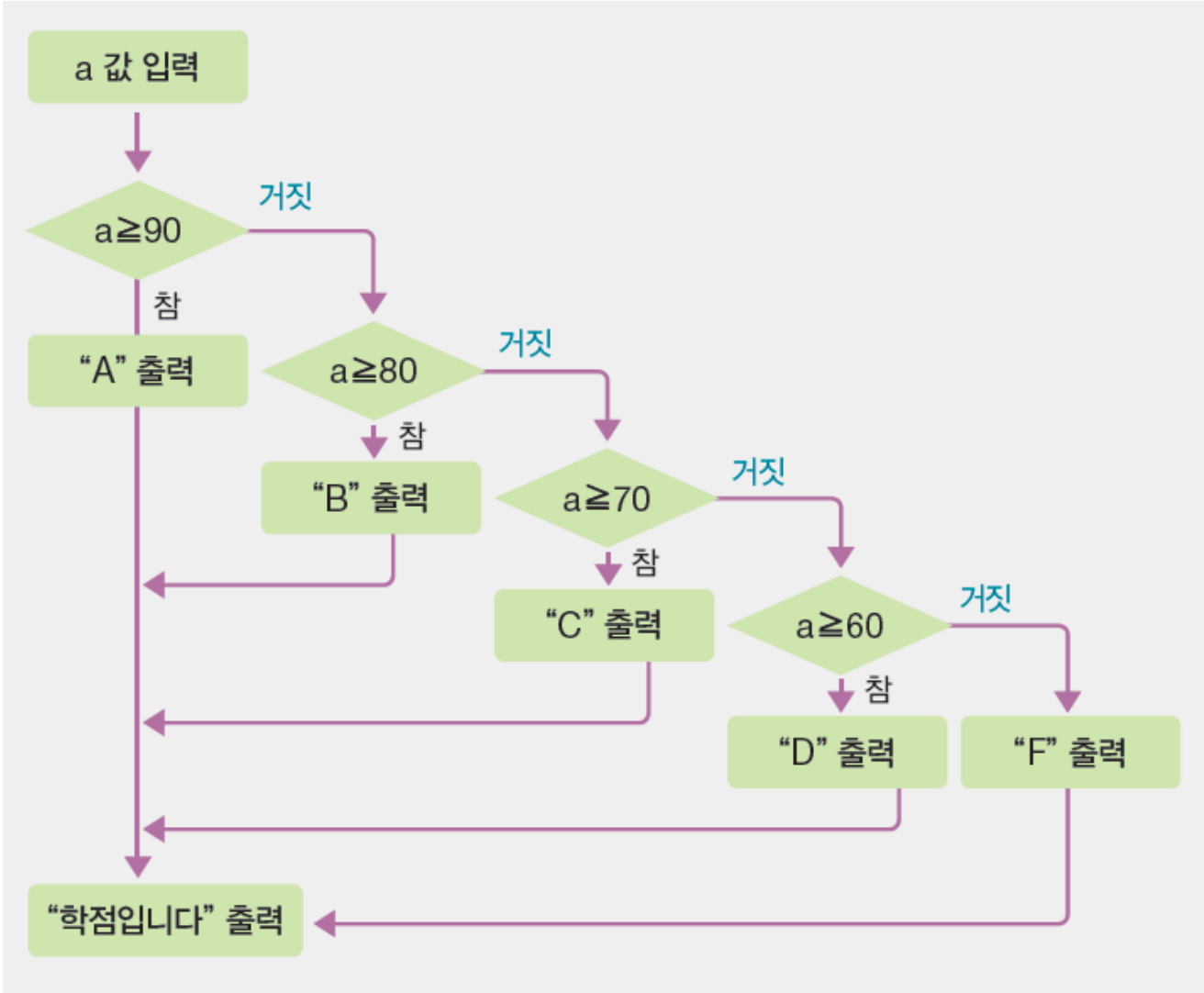
```
1  score=int(input("점수를 입력하세요 : "))
2
3  if score>=90 :
4      print("A")
5  else :
6      if score>=80 :
7          print("B")
8      else :
9          if score>=70 :
10             print("C")
11         else :
12             if score>=60 :
13                 print("D")
14             else :
15                 print("F")
16
17 print("학점입니다. ^^")
```

출력 결과

```
점수를 입력하세요 : 85 ← 사용자가 입력한 값
B
학점입니다. ^^
```

if문 (12)

그림 5-8
[소스코드 5-7]의 실행 과정



if문 (13)

- **if ~ elif ~ else문**

```
1  score=int(input("점수를 입력하세요 : "))
2
3  if   score>=90 :
4      print("A")
5  elif score>=80 :
6      print("B")
7  elif score>=70 :
8      print("C")
9  elif score>=60 :
10     print("D")
11 else :
12     print("F")
13
14 print("학점입니다. ^^")
```

소스코드 5-8
(파일명 : 05-08.py)

if문 (14)

- **간단한 계산기 프로그램 완성**

소스코드 5-9

(파일명 : 05-09.py)

```
1  ## 변수 선언 부분
2  a, b, ch=0, 0, ""
3
4  ## 메인(main) 코드 부분
5  a=int(input("첫 번째 수를 입력하세요 : "))
6  ch=input("계산할 연산자를  입력하세요 : ")
7  b=int(input("두 번째 수를 입력하세요 : "))
8
9  if ch=="+" :
10     print(" %d+%d=%d 입니다. " % (a, b, a+b))
11 elif ch=="-" :
12     print(" %d-%d=%d 입니다. " % (a, b, a-b))
```

if문 (15)

```
13 elif ch=="*" :
14     print(" %d*%d=%d 입니다. " % (a, b, a*b))
15 elif ch=="/" :
16     print(" %d/%d=%f 입니다. " % (a, b, a/b))
17 elif ch=="%" :
18     print(" %d%%%d=%d 입니다. " % (a, b, a%b))
19 elif ch=="//" :
20     print(" %d//%d=%d 입니다. " % (a, b, a//b))
21 elif ch=="**" :
22     print(" %d**%d=%d 입니다. " % (a, b, a**b))
23 else :
24     print(" 알 수 없는 연산자 입니다." )
```

If문 (16)

리스트와 함께 사용

- 리스트(List)는 여러 개를 한곳에 담아놓은 것

```
fruit=['사과', '배', '딸기', '포도']
print(fruit)
```

출력 결과

```
['사과', '배', '딸기', '포도']
```

- '리스트이름.append(항목)' 함수

```
fruit.append('귤')
print(fruit)
```

출력 결과

```
['사과', '배', '딸기', '포도', '귤']
```

If문 (17)

- 리스트 확인은 if문을 사용함

```
if  '딸기' in fruit :
   print ("딸기가 있네요 ^^")
```

출력 결과

```
딸기가 있네요 ^^
```

- 'if 항목 in 리스트 :' 는 리스트 안에 항목이 있으면 True를 반환

If문 (18)

- 0부터 9까지 숫자 중에서 리스트 안에 없는 숫자를 찾아내는 프로그램

소스코드 5-10
(파일명 : 05-10.py)

```
1  import random
2
3  numbers=[]
4  for num in range(0, 10) :
5      numbers.append(random.randrange(0, 10))
6
7  print("생성된 리스트", numbers)
8
9  for num in range(0, 10) :
10     if num not in numbers :
11         print ("%d 숫자는 리스트에 없네요." % num)
```

출력 결과

```
생성된 리스트 [5, 8, 8, 7, 8, 1, 9, 0, 0, 4]
2 숫자는 리스트에 없네요.
3 숫자는 리스트에 없네요.
6 숫자는 리스트에 없네요.
```

If문 (19)

- 'random.randrange(시작, 끝)' 함수는 '시작'부터 '끝-1'까지 숫자 중에서 임의의 숫자를 하나 반환
- 1행 : randrange() 함수를 사용하기 위해 필요
- 4행과 9행 : 0부터 9까지 총 10회를 반복.
- 5행 : random.randrange(0,10)은 0부터 9까지 숫자 중에서 임의의 숫자를 반환
- 7행 : 생성된 리스트 출력, 이 숫자들은 실행할 때마다 다름.
- 9행 : 0부터 9까지 num에 넣은 후 10행에서 numbers 리스트에 그 숫자가 없다면 11행에서 숫자가 없다는 메시지를 출력

If문 (20)

- **종합 계산기 프로그램 완성**

소스코드 5-11
(파일명 : 05-11.py)

```
## 변수 선언 부분
select, answer, numStr, num1, num2 = 0, 0, "", 0, 0

## 메인(main) 코드 부분
select = int(input("1. 수식 계산기  2.두 수 사이 합계  : "))

if  select == 1 :
    numStr = input(" *** 수식을 입력하세요  : ")
    answer = eval(numStr)
    print(" %s 결과는 %5.1f 입니다. " % (numStr, answer))
elif select == 2 :
    num1 = int(input(" *** 첫 번째 숫자를 입력하세요  : "))
    num2 = int(input(" *** 두 번째 숫자를 입력하세요  : "))
    for i in range(num1, num2+1) :
        answer = answer + i
    print(" %d + ... + %d는  %d입니다. " % (num1, num2, answer))
else :
     print("1 또는 2만 입력해야 합니다.")
```

If문 (21)

- 실행 후에 1을 입력하면 수식을 계산하는 계산기로 작동
- 8행 : 수식 자체를 입력
- 9행 : eval(수식) 함수는 수식을 계산해주는데 아주 유용한 함수
- 실행 후에 2를 입력하면 두 숫자 사이의 모든 수의 합계를 구함
- 14행 : for문 안에 range( )로 입력한 숫자 사이를 반복
- 17행 : 1, 2 외의 숫자를 입력하면 처리하는 오류 메시지

Section 02 for문

for문 (1)

- **for문의 개념**

```
for i in range(0, 3, 1) :
    print("안녕하세요? for문을 공부중입니다. ^^")
```

출력 결과

```
안녕하세요? for문을 공부중입니다. ^^
안녕하세요? for문을 공부중입니다. ^^
안녕하세요? for문을 공부중입니다. ^^
```

for문 (2)

for문의 작동

- range( ) 함수는 지정된 범위의 값을 반환

```
형식:
for 변수 in range( 시작값, 끝값+1 , 증가값 ) :
    이 부분을 반복
```

- range(0, 3, 1)은 [0, 1, 2]와 같음

```
for i in range(0, 3, 1) :
    print("안녕하세요? for문을 공부중입니다. ^^")
```

```
for i in [0, 1, 2] :
    print("안녕하세요? for문을 공부중입니다. ^^")
```

```
1회 : i에 0을 대입한 후 print() 수행
2회 : i에 1을 대입한 후 print() 수행
3회 : i에 2를 대입한 후 print() 수행
```

for문 (3)

- print()에서 i값을 사용해서 제일 앞에 숫자를 출력

```
for i in range(0, 3, 1) :
    print("%d : 안녕하세요? for문을 공부중입니다. ^^" % i )
```

출력 결과

```
0 : 안녕하세요? for문을 공부중입니다. ^^
1 : 안녕하세요? for문을 공부중입니다. ^^
2 : 안녕하세요? for문을 공부중입니다. ^^
```

- range() 함수의 시작 값을 2로 하고 i값을 1씩 줄여가며(0이 될 때까지) print() 함수를 세 번 실행하는 프로그램

```
for i in range(2, -1, -1) :
    print("%d : 안녕하세요? for문을 공부중입니다. ^^" % i )
```

출력 결과

```
2 : 안녕하세요? for문을 공부중입니다. ^^
1 : 안녕하세요? for문을 공부중입니다. ^^
0 : 안녕하세요? for문을 공부중입니다. ^^
```

for문 (4)

- 1~5까지 숫자들을 차례대로 출력

```
for i in range(1, 6, 1) :
    print("%d " % i , end=" " )
```

출력 결과

```
1  2  3  4  5
```

- 출력 결과가 한 줄에 나온 이유는 print() 함수의 마지막에 end=" "를 썼기 때문

for문 (5)

- **For문을 활용하여 합계 구하기**
  - 1부터 10까지의 합

```
1부터 10까지 변할 변수 준비(i)

for 변수(i)가 1을 시작으로 10까지 1씩 증가
   hap값에 i값을 더해줌

hap의 값을 출력
```

소스코드 6-2
(파일명 : 06-02.py)

```
1  i=0
2
3  for i in range(1, 11, 1) :
4      hap=hap+i
5
6  print("1에서 10까지의 합 : %d" % hap)
```

출력 결과

```
Traceback (most recent call last):
  File "C:/파이썬코드/06-02.py", line 4, in <module>
    hap=hap+i
NameError: name 'hap' is not defined
```

for문 (6)

- 소스 6-2는 변수 hap을 선언하지 않았기 때문에 오류 발생

소스코드 6-3
(파일명 : 06-03.py)

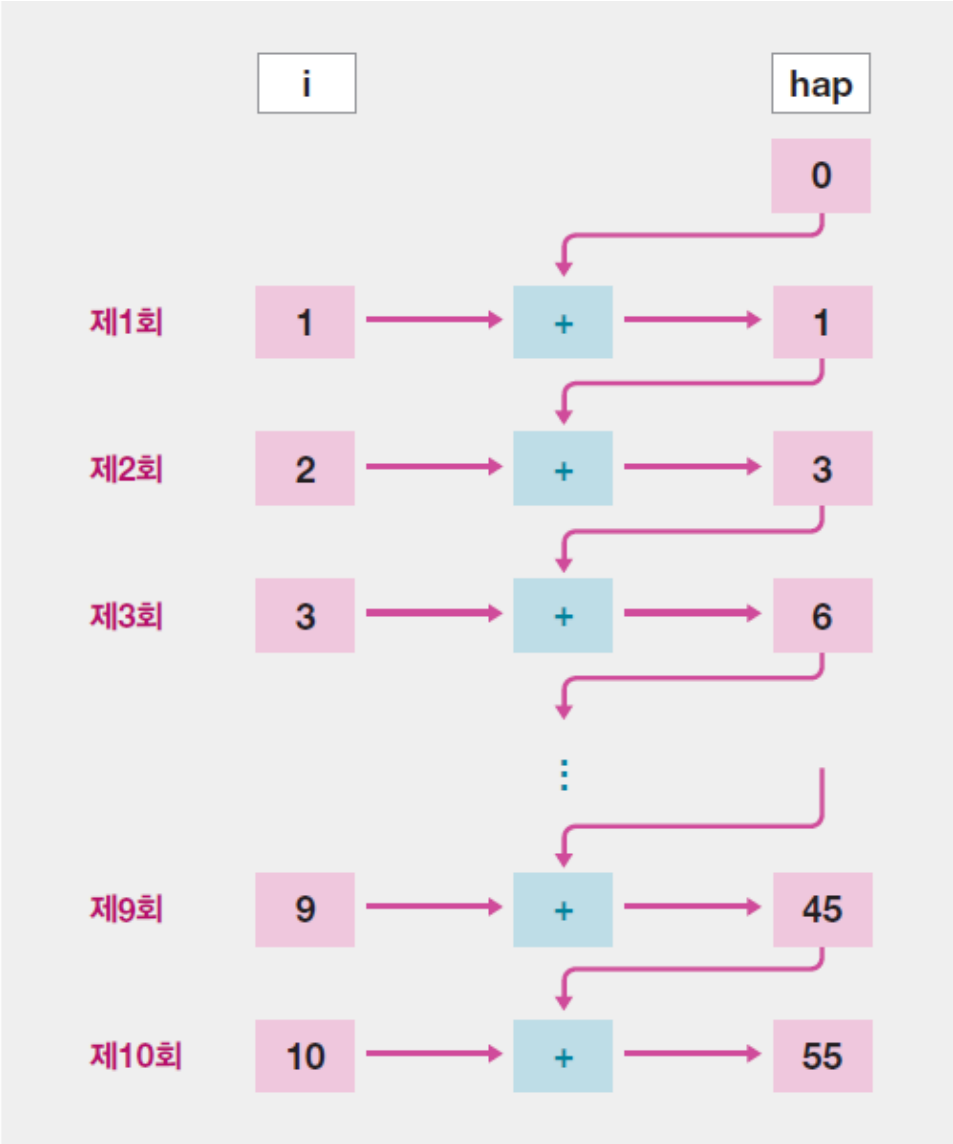
```
1  i, hap = 0, 0
2
3  for i in range(1, 11, 1) :
4      hap = hap + i
5
6  print("1에서 10까지의 합 : %d" % hap)
```

출력 결과

```
1에서 10까지의 합 : 55
```

for문 (7)

그림 6-3
변수 i와 hap의 변화



for문 (8)

- 500과 1000 사이에 있는 홀수의 합을 구하는 프로그램

소스코드 6-4
(파일명 : 06-04.py)

```
1  i, hap=0, 0
2
3  for i in range(501, 1001, 2) :
4      hap=hap+i
5
6  print("500에서 1000까지 홀수의 합 : %d" % hap)
```

출력 결과

```
500에서 1000까지 홀수의 합 : 187500
```

for문 (9)

- **입력한 값까지 for문으로 합계 구하기**
  - 사용자가 원하는 값을 입력하여 1부터 입력한 수까지의 합을 구하는 프로그램

소스코드 6-5
(파일명 : 06-05.py)

```
1  i, hap=0, 0
2  num=0
3
4  num=int(input("값 입력 : "))
5
6  for i in range(1, num+1, 1) :
7      hap=hap+i
8
9  print("1에서 %d까지 합 : %d" % (num, hap))
```

출력 결과

```
값 입력: 100  ← 사용자가 입력한 값
1에서 100까지 합 : 5050
```

for문 (10)

- 시작 값, 끝 값, 증가 값 모두 사용자에게서 입력 받아 계산

소스코드 6-6
(파일명 : 06-06.py)

```
1  i, hap = 0, 0
2  num1, num2, num3 = 0, 0, 0
3
4  num1 = int(input("시작값 입력 : "))
5  num2 = int(input("끝값 입력 : "))
6  num3 = int(input("증가값 입력 : "))
7
8  for i in range(num1, num2 + 1, num3) :
9      hap = hap + i
10
11 print("%d에서 %d까지 %d씩 증가함 값의 합 : %d" % (num1, num2, num3, hap))
```

출력 결과

```
시작값 입력 : 2 ← 사용자가 입력한 값
끝값 입력 : 300 ← 사용자가 입력한 값
증가값 입력 : 3 ← 사용자가 입력한 값
2에서 300까지 3씩 증가함 값의 합 : 15050
```

for문 (11)

- 입력한 숫자의 구구단을 출력하는 프로그램

소스코드 6-7
(파일명 : 06-07.py)

```
1  i, dan=0, 0
2
3  dan=int(input(" 몇 단 ? "))
4
5  for i in range(1, 10, 1) :
6      print(" %d X %d =%2d" % (dan, i, dan * i))
```

출력 결과

```
몇 단 ? 7 ← 사용자가 입력한 값
 7 X 1 = 7
 7 X 2 = 14
~~ 중간 생략 ~~
 7 X 9 = 63
```

for문 (12)

- **중첩 for문의 개념**
  - 중첩 for문은 for문 내부에 또 다른 for문이 들어있는 형태

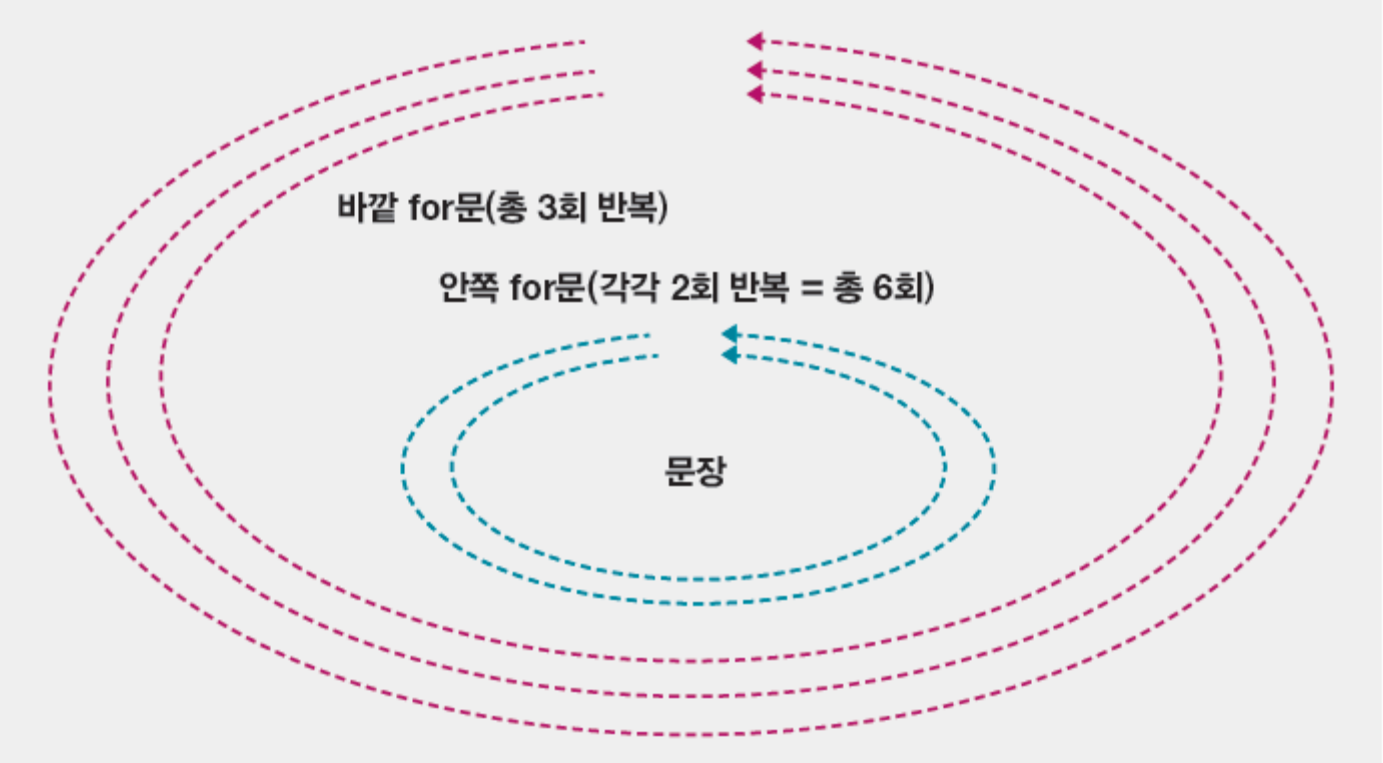


그림 6-4
중첩 for문의 동작 개념

for문 (13)

- 중첩 for문의 기본 코드

```
for i in range (0, 3, 1) :
    for k in range(0, 2, 1) :
            print("파이썬은 꿀잼입니다. ^^ (i값: %d, k값: %d)" % ( i, k ))
```

출력 결과

```
파이썬은 꿀잼입니다. ^^ (i값: 0, k값: 0)
파이썬은 꿀잼입니다. ^^ (i값: 0, k값: 1)
파이썬은 꿀잼입니다. ^^ (i값: 1, k값: 0)
파이썬은 꿀잼입니다. ^^ (i값: 1, k값: 1)
파이썬은 꿀잼입니다. ^^ (i값: 2, k값: 0)
파이썬은 꿀잼입니다. ^^ (i값: 2, k값: 1)
```

for문 (14)

- 중첩 for문의 실행 횟수는 '바깥 for문 반복 횟수 × 안쪽 for문 반복 횟수'
- 처리되는 순서

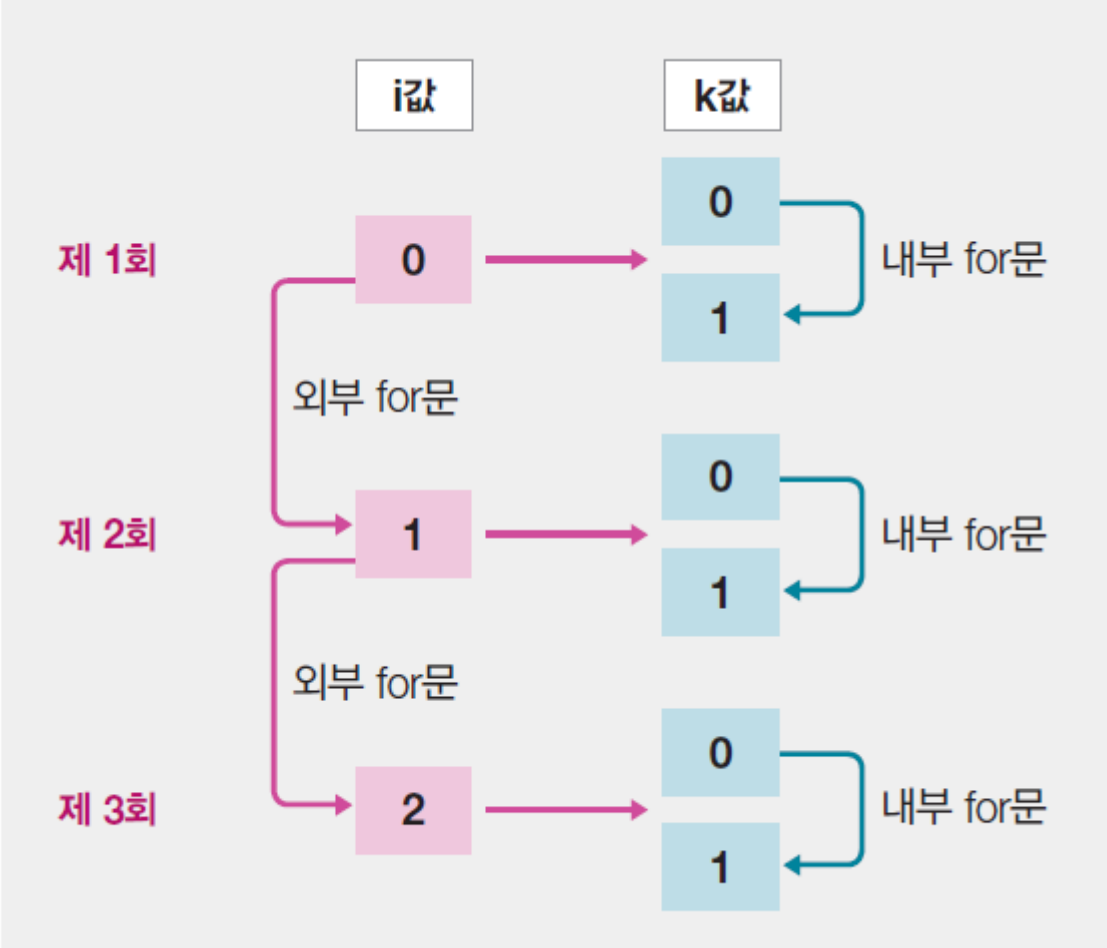
```
(1) 외부 for문 1회:  i에 0을 대입
        내부 for문 1회: k에 0을 대입한 후에 print() 수행
        내부 for문 2회: k에 1을 대입한 후에 print() 수행
```

```
(2) 외부 for문 2회:  i에 1을 대입
        내부 for문 1회: k에 0을 대입한 후에 print() 수행
        내부 for문 2회: k에 1을 대입한 후에 print() 수행
```

```
(2) 외부 for문 3회:  i에 2를 대입
        내부 for문 1회: k에 0을 대입한 후에 print() 수행
        내부 for문 2회: k에 1을 대입한 후에 print() 수행
```

for문 (15)

그림 6-5
중첩 for문에서
i와 k값의 변화



for문 (16)

- **중첩 for문의 활용**
  - 구구단 2단부터 9단까지 출력

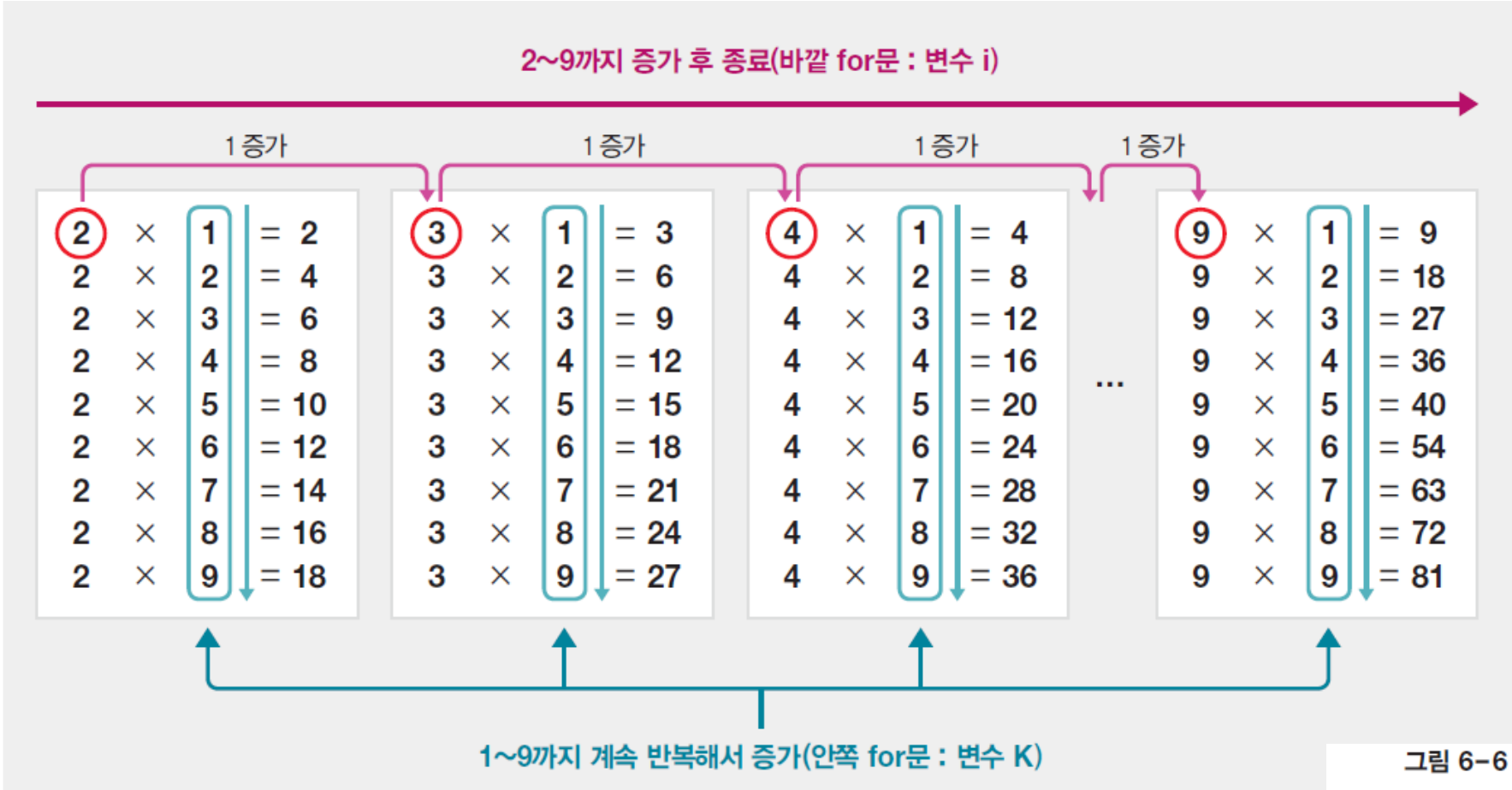


그림 6-6
구구단에서
변수 i와 k의 추출

for문 (17)

소스코드 6-8
(파일명 : 06-08.py)

```
1  i, k=0, 0
2
3  for i in range(2, 10, 1) :
4      for k in range(1, 10, 1) :
5          print(" %d X %d =%2d" % (i, k, i * k))
6      print("")
```

출력 결과

```
 2 X 1 = 2
 2 X 2 = 4
 2 X 3 = 6
 2 X 4 = 8
~~ 중간 생략 ~~
 9 X 8 = 72
 9 X 9 = 81
```

for문 (18)

- **구구단 출력 프로그램 완성**
  - 기존 구구단 결과는 세로로 출력되므로 결과를 보려면 스크롤을 움직여야 함.
  - 오른쪽 여백에 구구단 결과를 보여주기 위해 프로그램을 수정

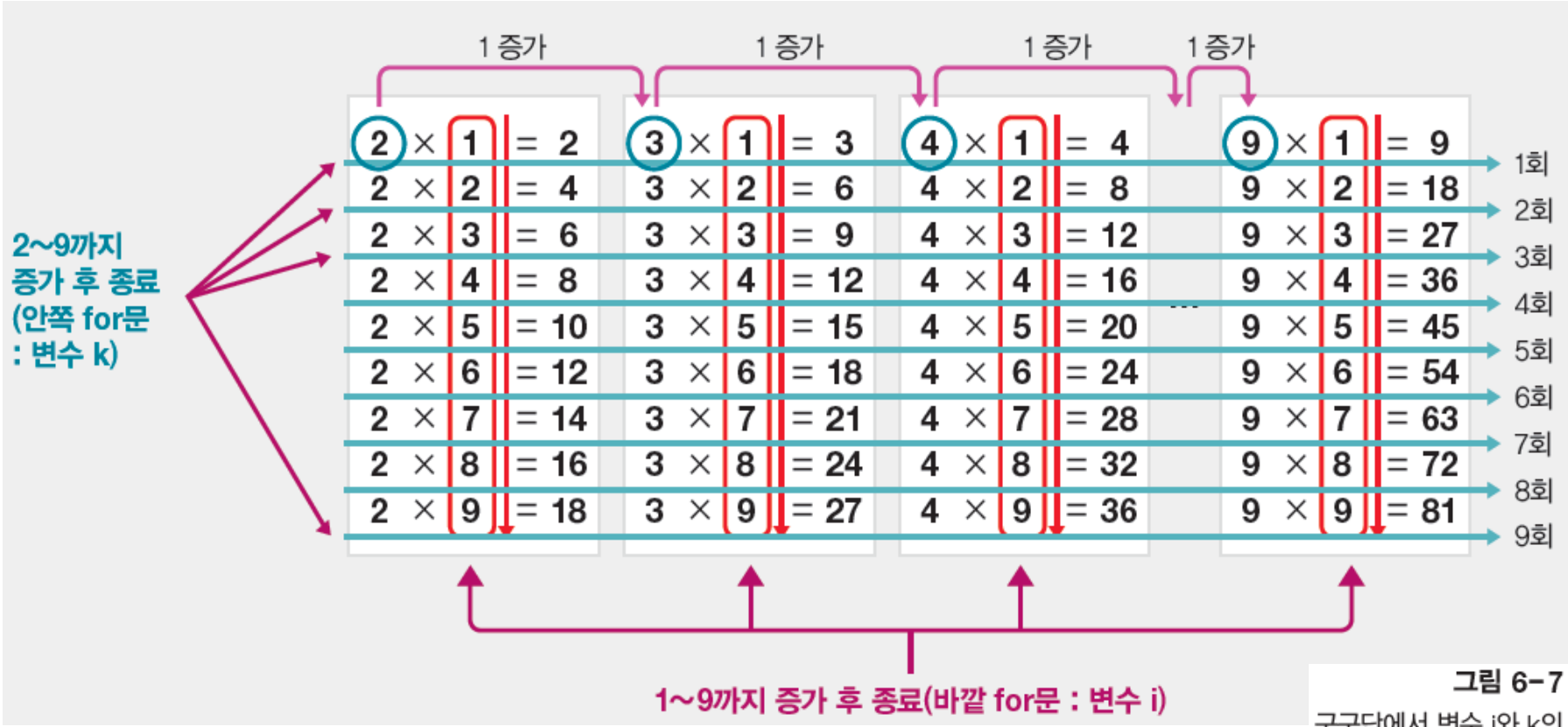


그림 6-7
구구단에서 변수 i와 k의 추출 (단 가로 먼저 출력)

for문 (19)

소스코드 6-9
(파일명 : 06-09.py)

```
1  ## 변수 선언 부분
2  i, k, guguLine=0, 0, ""
3
4  ## 메인(main) 코드 부분
5  for i in range(2, 10) :
6      guguLine=guguLine+(" # %d단 #" % i)
7
8  print(guguLine)
9
10 for i in range(1, 10) :
11     guguLine=""
12     for k in range(2, 10) :
13         guguLine=guguLine+str("%2d X %2d=%2d" % (k, i, k*i))
14     print(guguLine)
```

Section 03 while문

While문 (1)

- **For문과 while문의 비교**
  - for문의 형식

```
for 변수 in range(시작값, 끝값+1, 증가값)
```

  - while문은 while문 안의 조건식을 확인하여 값이 참이면 '문장'을 수행. 조건식이 참인 동안 계속 반복함

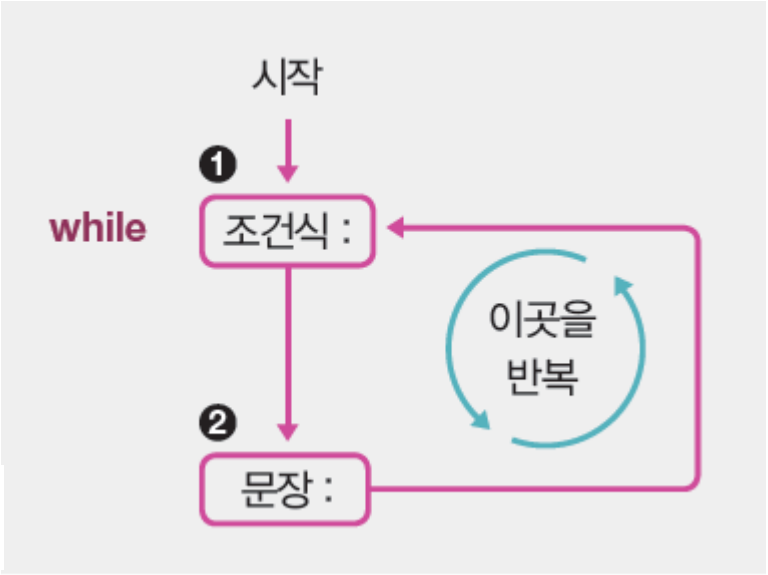


그림 6-8
while문의 실행 순서

```
형식:
변수 = 시작값
while 변수값 < 끝값 :
    이 부분을 반복
    변수 = 변수 + 증가값
```

While문 (2)

- For문과 while문 비교

```
for i in range(0, 3, 1) :
    print("%d : 안녕하세요? for문을 공부중입니다. ^^" % i )
```

출력 결과

```
0 : 안녕하세요? for문을 공부중입니다. ^^
1 : 안녕하세요? for문을 공부중입니다. ^^
2 : 안녕하세요? for문을 공부중입니다. ^^
```

```
i=0
while i<3 :
    print("%d : 안녕하세요? while문을 공부중입니다. ^^" % i )
    i=i+1
```

출력 결과

```
0 : 안녕하세요? while문을 공부중입니다. ^^
1 : 안녕하세요? while문을 공부중입니다. ^^
2 : 안녕하세요? while문을 공부중입니다. ^^
```

While문 (3)

- While문을 이용한 1에서 10까지의 합

소스코드 6-10
(파일명 : 06-10.py)

```
1  i, hap = 0, 0
2
3  i = 1
4  while i < 11 :
5      hap = hap + i
6      i = i + 1
7
8  print("1에서 10까지의 합 : %d" % hap)
```

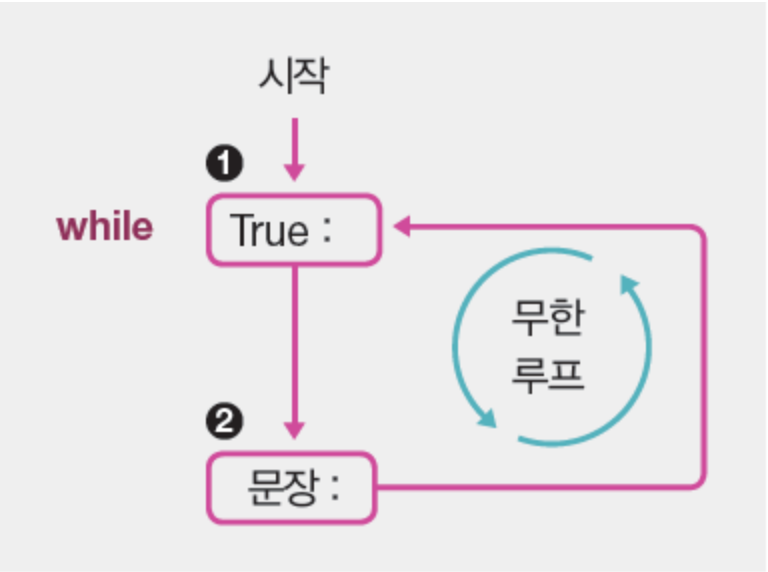
출력 결과

```
1에서 10까지의 합 : 55
```

While문 (4)

- **무한 루프를 위한 while문**
  - 무한 루프를 적용하려면 'while 조건식 : '의 조건식을 True로 지정
  - 무한 루프를 중지하려면 Ctrl + C 를 누름

그림 6-9
while을 이용한 무한 루프




```
while True :
   print("ㅋ ", end=" ")
```

출력 결과

```
ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ  ~~ 무한 반복
```

While문 (5)

- 입력한 두 수의 합계를 반복해서 계산하는 프로그램

소스코드 6-11
(파일명 : 06-11.py)

```
1  hap=0
2  a, b=0, 0
3
4  while True :
5      a=int(input("더할 첫 번째 수 입력 : "))
6      b=int(input("더할 두 번째 수 입력 : "))
7      hap=a+b
8      print("%d+%d=%d" % (a, b, hap))
```

출력 결과

```
더할 첫 번째 수 입력 : 55  ← 사용자가 입력한 값
더할 두 번째 수 입력 : 22  ← 사용자가 입력한 값
55+22=77
더할 첫 번째 수 입력 : 77  ← 사용자가 입력한 값
더할 두 번째 수 입력 : 128  ← 사용자가 입력한 값
77+128=205
더할 첫 번째 수 입력 :
```

While문 (6)

- 덧셈,

  뺄셈,

  곱셈,

  나눗셈,

  나머지 구하기

```
ch=""
a, b=0, 0

while True :
    a=int(input("계산할 첫 번째 수 입력 : "))
    b=int(input("계산할 두 번째 수 입력 : "))
    ch=input("계산할 연산자를 입력 : ")

    if (ch=="+") :
        print("%d+%d=%d 입니다." % (a, b, a+b))
    elif (ch=="-") :
        print("%d-%d=%d 입니다." % (a, b, a-b))
    elif (ch=="*") :
        print("%d*%d=%d 입니다." % (a, b, a*b))
    elif (ch=="/") :
        print("%d/%d=%5.2f 입니다." % (a, b, a/b))
    elif (ch=="%") :
        print("%d%%%d=%d 입니다." % (a, b, a%b))
    elif (ch=="//") :
        print("%d//%d=%d 입니다." % (a, b, a//b))
    elif (ch=="**") :
        print("%d**%d=%d 입니다." % (a, b, a**b))
    else :
        print("연산자를 잘못 입력했습니다.")
```

While문 (7)

출력 결과

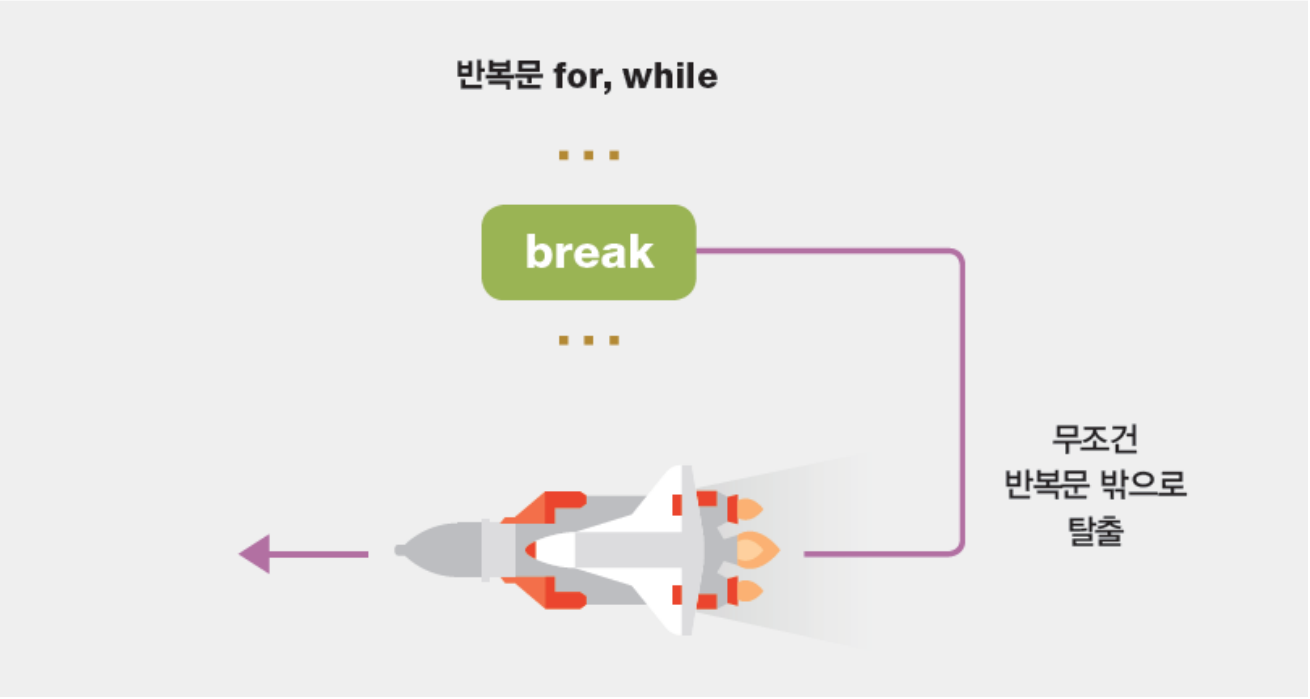
```
계산할 첫 번째 수 입력 : 22  ← 사용자가 입력한 값
계산할 두 번째 수 입력 : 33  ← 사용자가 입력한 값
계산할 연산자를 입력 : *  ← 사용자가 입력한 값
22*33=726 입니다.
계산할 첫 번째 수 입력 : 10  ← 사용자가 입력한 값
계산할 두 번째 수 입력 : 4  ← 사용자가 입력한 값
계산할 연산자를 입력 : %  ← 사용자가 입력한 값
10%4=2 입니다.
계산할 첫 번째 수 입력 :
```

Section 04 기타 제어문

기타 제어문 (1)

- **반복문 탈출하는 break문**

그림 6-10
break문의 작동




```
for i in range(1, 100) :
        print("for문을 %d번 실행했습니다." % i)
        break
```

출력 결과

```
for문을 1번 실행했습니다.
```

기타 제어문 (2)

- [소스코드 6-11]을 break문을 사용하여 수정

소스코드 6-13
(파일명 : 06-13.py)

```
1  hap=0
2  a, b=0, 0
3
4  while True :
5      a=int(input("더할 첫 번째 수 입력 : "))
6      if a==0 :
7          break
8      b=int(input("더할 두 번째 수 입력 : "))
9      hap=a+b
10     print("%d+%d=%d"%(a, b, hap))
11
12 print("0을 입력해서 반복문을 탈출했습니다")
```

출력 결과

```
더할 첫 번째 수 입력 : 55  ← 사용자가 입력한 값
더할 두 번째 수 입력 : 22  ← 사용자가 입력한 값
55+22=77
더할 첫 번째 수 입력 : 77  ← 사용자가 입력한 값
더할 두 번째 수 입력 : 128  ← 사용자가 입력한 값
77+128=205
더할 첫 번째 수 입력 : 0  ← 사용자가 입력한 값
0을 입력해서 반복문을 탈출했습니다
```

기타 제어문 (3)

- 1~100까지 더하되, 누적 합계(hap)가 1000 이상이 되는 시작 지점을 구하는 프로그램

소스코드 6-14
(파일명 : 06-14.py)

```
1  hap, i=0, 0
2
3  for i in range(1,101) :
4      hap+=i
5
6      if hap>=1000 :
7          break
8
9  print("1~100의 합에서 최초로 1000이 넘는 위치 : %d" % i)
```

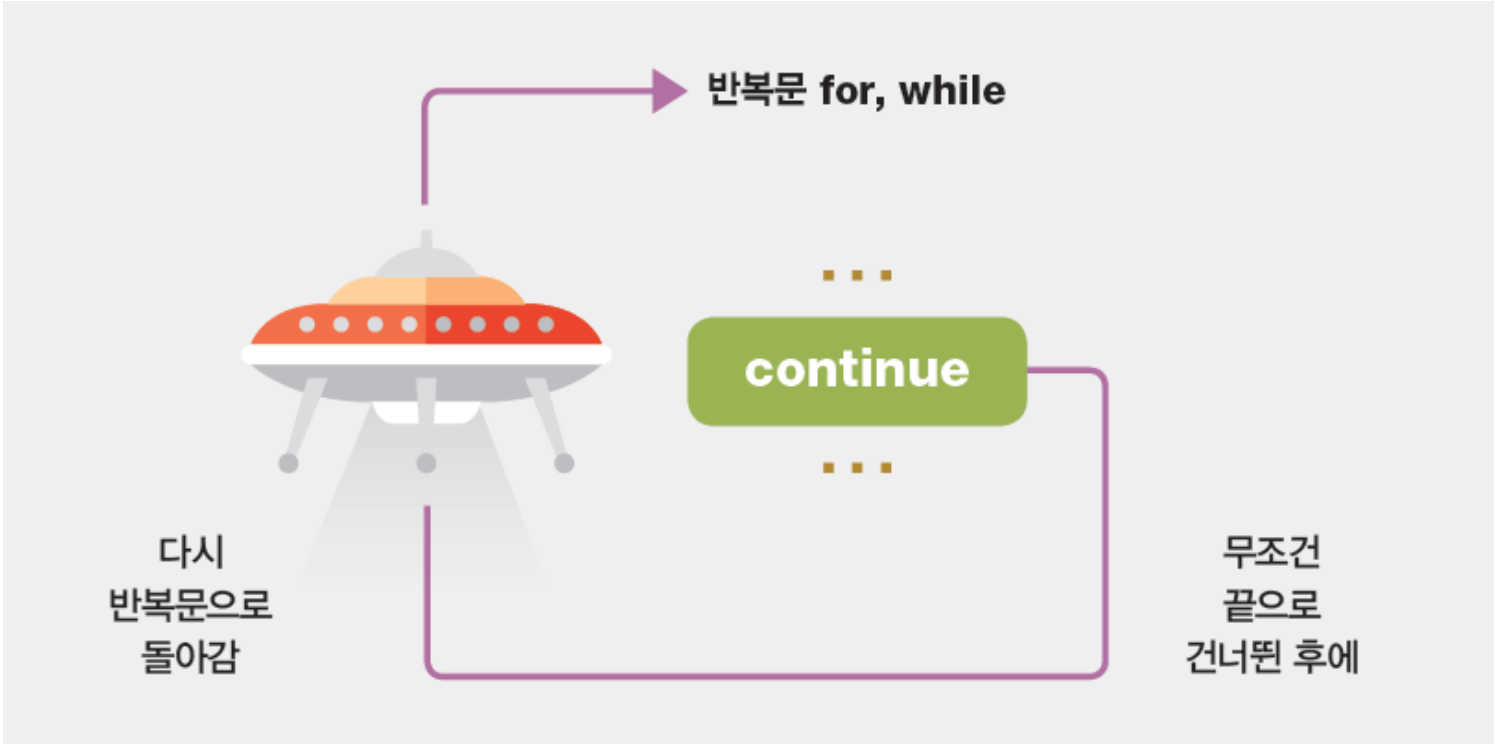
출력 결과

```
1~100의 합에서 최초로 1000이 넘는 위치 : 45
```

기타 제어문 (4)

- **반복문으로 다시 돌아가는 continue문**
  - continue문을 만나면 무조건 블록의 남은 부분을 건너뛰고 반복문의 처음으로 돌아감

그림 6-11
continue문의 작동



기타 제어문 (5)

- 1~100까지의 합을 구하되 1 +2 +4 +5 +7 +8 +10 +...과 같이 3의 배수를 건너뛰고(=제외하고) 더하는 프로그램

소스코드 6-15
(파일명 : 06-15.py)

```
1  hap, i=0, 0
2
3  for i in range(1,101) :
4      if i % 3==0 :
5          continue
6
7      hap+=i
8
9  print("1~100의 합계(3의 배수 제외) : %d" % hap)
```

출력 결과

```
1~100의 합계(3의 배수 제외) : 3367
```

기타 제어문 (6)

- ♥ 출력 프로그램 완성
  - 문자열의 한 글자씩 접근 방법
  - str은 세 개의 문자이므로 str[0], str[1], str[2]로 접근 가능함

```
str = "abc"
print(str[0])
print(str[1])
print(str[2])
```

출력 결과

```
a
b
c
```

기타 제어문 (7)

소스코드 6-16
(파일명 : 06-16.py)

```
1  ## 변수 선언 부분
2  i, k, heartNum=0, 0, 0
3  numStr, ch, heartStr="", "", ""
4
5  ## 메인(main) 코드 부분
6  numStr=input("숫자를 여러 개 입력하세요 : ")
7  print("")
8
9  i=0
10 ch=numStr[i]
11 while True :
12     heartNum=int(ch)
13
14     heartStr=""
```

기타 제어문 (8)

```
15      for k in range (0, heartNum) :
16          heartStr += "\u2665"
17          k += 1
18
19      print(heartStr)
20
21      i += 1
22      if (i > len(numStr) - 1 ) :
23          break
24
25      ch = numStr[i]
```

Thank You