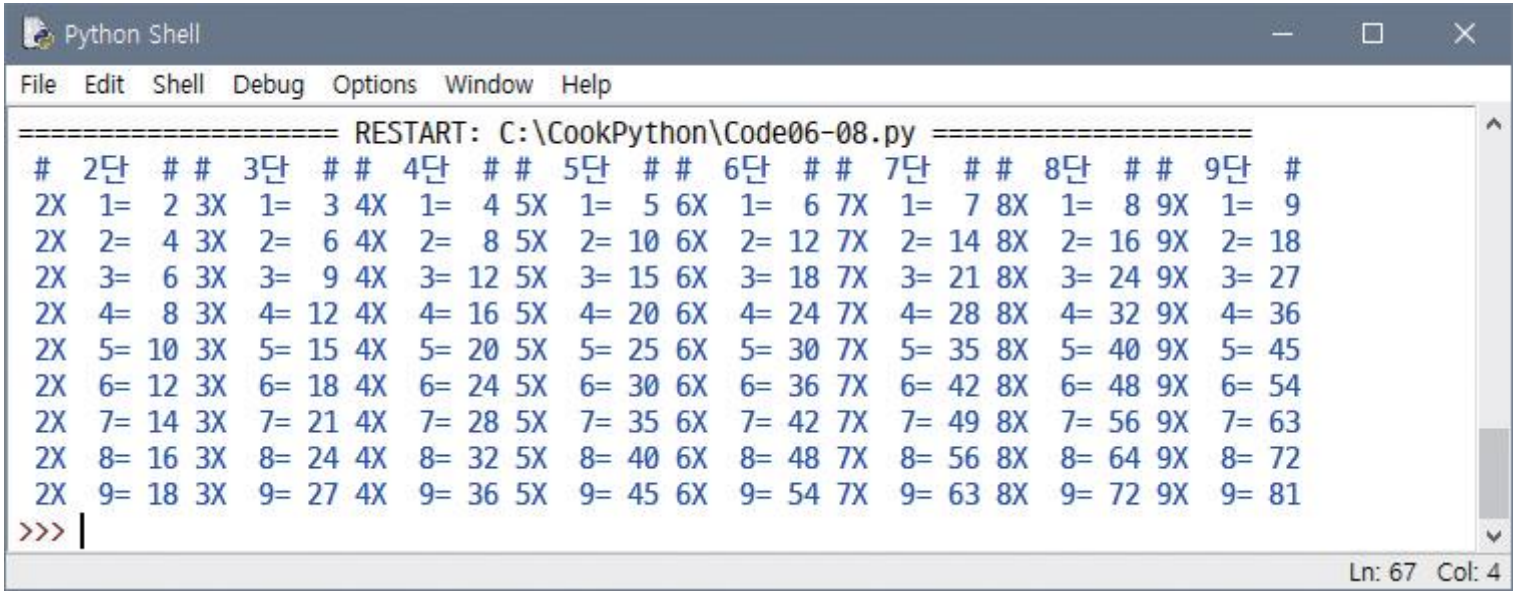
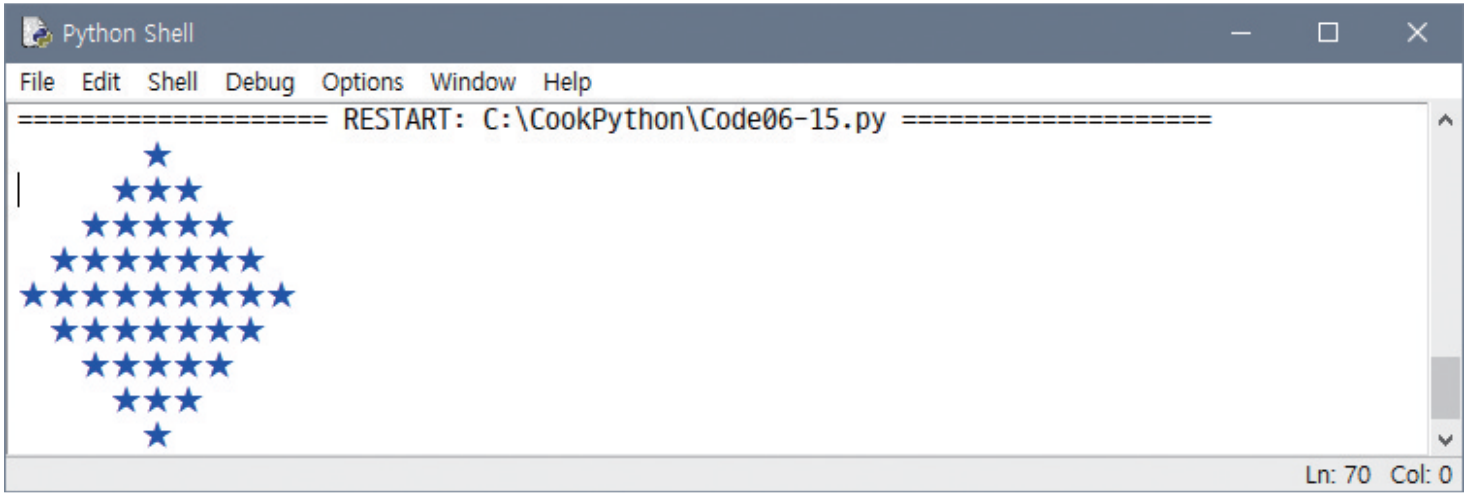
Section01 이 장에서 만들 프로그램

- **[프로그램 1] 구구단 출력**
  - for 문을 사용



Section01 이 장에서 만들 프로그램

- **[프로그램 2] 마름모 모양 출력**
  - while 문 활용



Section02 기본 for 문

■ 반복문의 개념과 필요성

- 예 : 반복문 사용 않는 경우

출력 결과

```
안녕하세요? for 문을 공부 중입니다. ^^
안녕하세요? for 문을 공부 중입니다. ^^
안녕하세요? for 문을 공부 중입니다. ^^
```

Code06-01(1).py

```
1  print("안녕하세요? for 문을 공부 중입니다. ^^")
2  print("안녕하세요? for 문을 공부 중입니다. ^^")
3  print("안녕하세요? for 문을 공부 중입니다. ^^")
```

- 예 : 반복문 사용한 경우

Code06-01(2).py

```
1  for i in range(0, 3, 1) :
2      print("안녕하세요? for 문을 공부 중입니다. ^^")
```

출력 결과

```
안녕하세요? for 문을 공부 중입니다. ^^
안녕하세요? for 문을 공부 중입니다. ^^
안녕하세요? for 문을 공부 중입니다. ^^
```

Section02 기본 for 문

for 문의 개념

- 기본 형식

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

range(3)은 range(0, 3, 1)과 같다

- 예 : range() 함수 사용과 내부적 변경

```
for i in range(0, 3, 1) :
    print("안녕하세요? for 문을 공부 중입니다. ^^")
```

```
for i in [0, 1, 2] :
    print("안녕하세요? for 문을 공부 중입니다. ^^")
```

Section02 기본 for 문

- i값 코드 내부 사용

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

출력 결과

```
0 : 안녕하세요? for 문을 공부 중입니다. ^^
1 : 안녕하세요? for 문을 공부 중입니다. ^^
2 : 안녕하세요? for 문을 공부 중입니다. ^^
```

Tip • _(언더바) : i를 사용하지 않으려면 i 대신 _(언더바) 사용

```
for _ in range(0, 3, 1) :
    print("안녕하세요? for 문을 공부 중입니다. ^^")
```

Section02 기본 for 문

- 예 : range() 함수의 시작값 2, i값을 1씩 줄여(0이 될 때까지) print() 함수 3번 실행

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

출력 결과

```
2 : 안녕하세요? for 문을 공부 중입니다. ^^
1 : 안녕하세요? for 문을 공부 중입니다. ^^
0 : 안녕하세요? for 문을 공부 중입니다. ^^
```

- 예 : 1~5의 숫자들을 차례로 출력

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

출력 결과

```
1 2 3 4 5
```

for 문을 활용한 합계 구하기

- for 문을 배우기 전의 방식으로 1~10 의 합계를 구하는 프로그램 만들기

```
hap = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
print("1에서 10까지의 합계 : %d" % hap)
```

출력 결과

```
1에서 10까지의 합계 : 55
```

- for 문 작성 내용

```
1부터 10까지 변할 i 변수 준비

for i 변수가 1을 시작으로 10까지 1씩 증가
    hap값에 i값을 더해 줌

hap값 출력
```

Section02 기본 for 문

- for 문 내용을 코드로 작성

Code06-02(1).py

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

4행 'hap=hap+i'에서 hap에 어떤 값이 있어야 다시 누적
hap 자체가 존재하지 않아 더할 것이 없어서 오류가 발생

출력 결과

```
Traceback (most recent call last):
    File "C:\CookPython\Code06-02(1).py", line 4, in <module>
        hap = hap + i
NameError: name 'hap' is not defined
```

Section02 기본 for 문

- Code06-02(1).py 1행의 hap 초기화 코드 추가

Code06-02(2).py

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

주의 : i 변수와 hap 변수의 값

출력 결과

```
1에서 10까지의 합계 : 55
```

Section02 기본 for 문

- i 와 hap 변수값의 변화

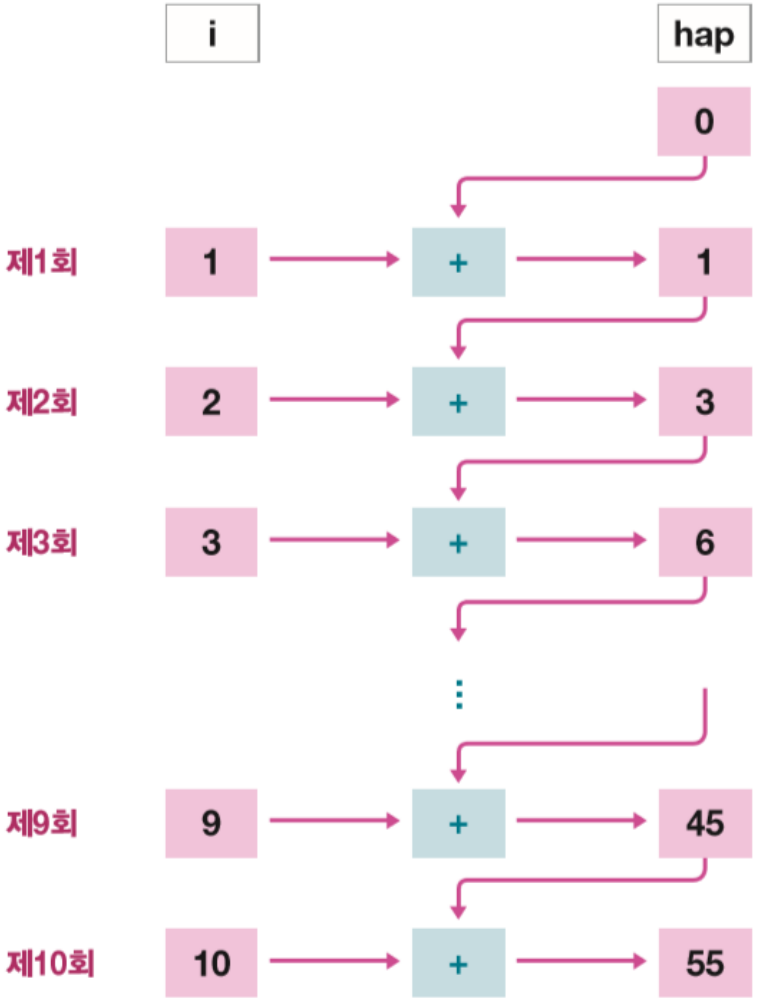


그림 6-1 i 와 hap 변수값의 변화

Section02 기본 for 문

- 예 : 500과 1000 사이에 있는 홀수의 합계

Code06-03.py

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

출력 결과

```
500과 1000 사이에 있는 홀수의 합계 : 187500
```

SELF STUDY 6-1

Code06-03.py를 0과 100 사이에 있는 7의 배수 합계를 구하도록 수정해 보자.

출력 결과

```
0과 100 사이에 있는 7의 배수 합계 : 735
```

키보드로 입력한 값까지 합계 구하기

키보드로 입력한 수까지의 합계 구하기

- input() 함수로 1부터 사용자가 입력한 수까지 합계 구하는 프로그램

Code06-04.py

```
1  i, hap = 0, 0
2  num = 0
3
4  num = int(input("값을 입력하세요 : "))
5
6  for i in range(1, num + 1, 1) :
7      hap = hap + i
8
9  print("1에서 %d까지의 합계 : %d" % (num, hap))
```

출력 결과

```
값을 입력하세요 : 100
1에서 100까지의 합계 : 5050
```

■ 키보드로 입력한 값까지 합계 구하기

▪ 키보드로 입력한 수까지의 합계 구하기

- input() 함수로 1부터 사용자가 입력한 수까지 합계 구하는 프로그램

Code06-04.py

```
1  i, hap = 0, 0
2  num = 0
3
4  num = int(input("값을 입력하세요 : "))
5
6  for i in range(1, num + 1, 1) :
7      hap = hap + i
8
9  print("1에서 %d까지의 합계 : %d" % (num, hap))
```

2행 : 사용자가 입력한 값 저장할 num 변수 선언
4행 : input() 함수로 사용자가 입력한 숫자를 num에 대입
6행 : range(1, 입력숫자+1, 1)을 사용해 1부터 사용자가 입력한 숫자(num)까지 1씩 증가하면서 for 문 반복
9행 : 사용자가 입력한 숫자까지 합계를 구해 사용자가 입력한 숫자와 함께 출력

출력 결과

```
값을 입력하세요 : 100
1에서 100까지의 합계 : 5050
```

Section02 기본 for 문

- 예: 시작값과 끝값, 증가값까지 사용자 입력

Code06-05.py

```
1  i, hap = 0, 0
2  num1, num2, num3 = 0, 0, 0
3
4  num1 = int(input("시작값을 입력하세요 : "))
5  num2 = int(input("끝값을 입력하세요 : "))
6  num3 = int(input("증가값을 입력하세요 : "))
7
8  for i in range(num1, num2 + 1, num3) :
9      hap = hap + i
10
11 print("%d에서 %d까지 %d씩 증가시킨 값의 합계 : %d" % (num1, num2, num3, hap))
```

4~6행 : 값 3개 입력
8행 : 입력한 값 사용 range() 지정

출력 결과

```
시작값을 입력하세요 : 2
끝값을 입력하세요 : 300
증가값을 입력하세요 : 3
2에서 300까지 3씩 증가시킨 값의 합계 : 15050
```

Section02 기본 for 문

- 예: 사용자가 입력한 숫자의 단에서 구구단을 출력

Code06-06.py

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

1행 : 출력하려는 단을 입력받을 변수 선언
3행 : 키보드로 입력
5행 : i는 1에서 9까지 증가
6행 : 구구단의 각 행 출력

출력 결과

```
단을 입력하세요 : 7
7  X  1  =   7
7  X  2  =  14
… 중략 …
7  X  9  =  63
```

SELF STUDY 6-2

Code06-06.py를 수정해서 입력한 단을 거꾸로 출력하도록 해보자.

출력 결과

```
단을 입력하세요 : 7
9 X 7 = 63
8 X 2 = 56
… 중략 …
1 X 9 = 7
```

Section03 중첩 for 문

- **중첩 for 문의 개념**

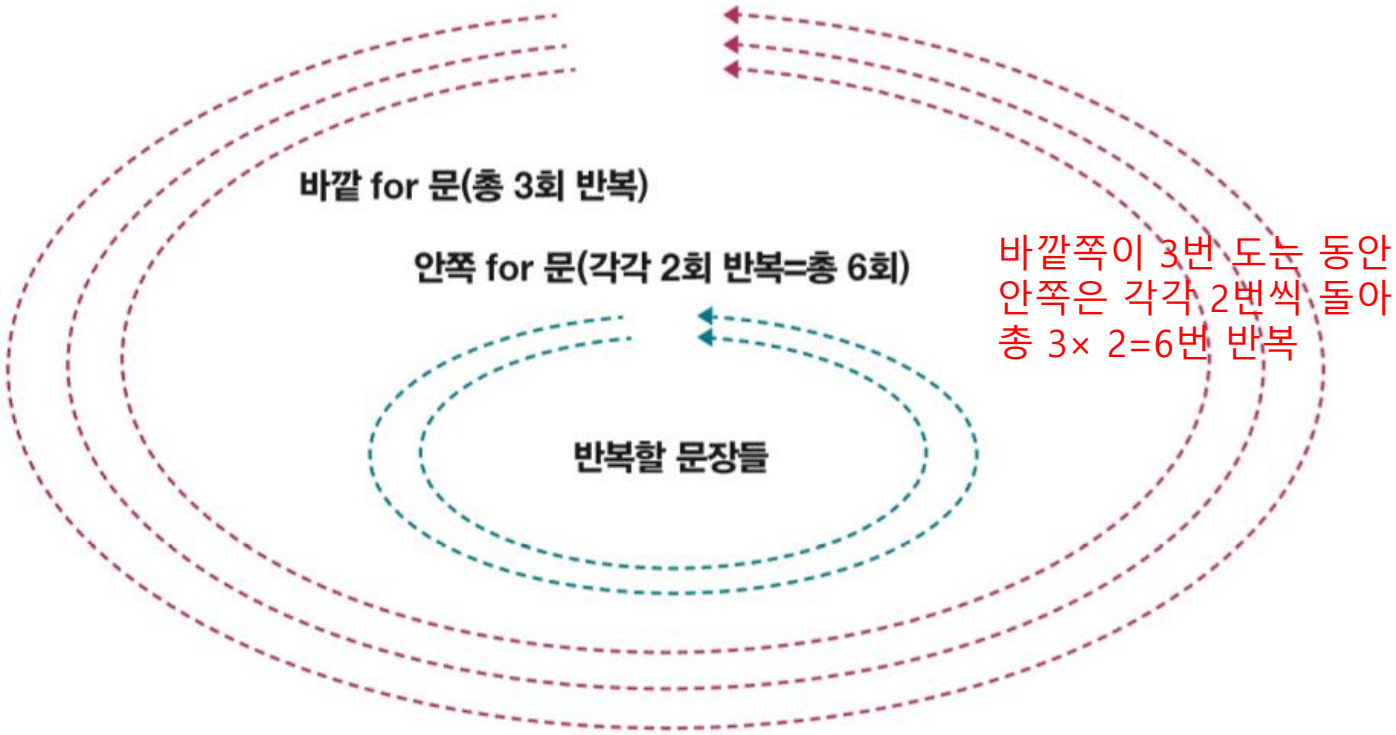


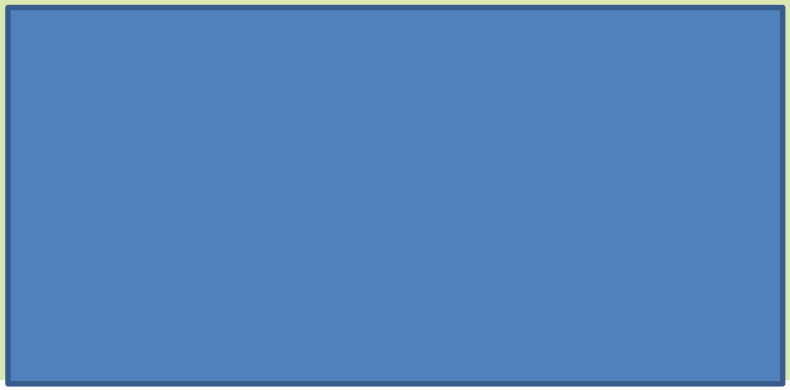
그림 6-2 중첩 for 문의 작동 개념

Section03 중첩 for 문

- 중첩 for 문의 기본 형식

```
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



Section03 중첩 for 문

- 처리 순서
  - 외부 변수인 i는 계속 0, 1, 2로 변경된 후 끝나지만, 내부 변수인 k는 0과 1을 계속 반복

```
❶ 외부 for 문 1회 : i에 0을 대입
   내부 for 문 1회 : k에 0을 대입 후 print() 함수 수행
   내부 for 문 2회 : k에 1을 대입 후 print() 함수 수행

❷ 외부 for 문 2회 : i에 1을 대입
   내부 for 문 1회 : k에 0을 대입 후 print() 함수 수행
   내부 for 문 2회 : k에 1을 대입 후 print() 함수 수행

❸ 외부 for 문 3회 : i에 2를 대입
   내부 for 문 1회 : k에 0을 대입 후 print() 함수 수행
   내부 for 문 2회 : k에 1을 대입 후 print() 함수 수행
```

Section03 중첩 for 문

- 중첩 for 문에서 i와 k값 변화

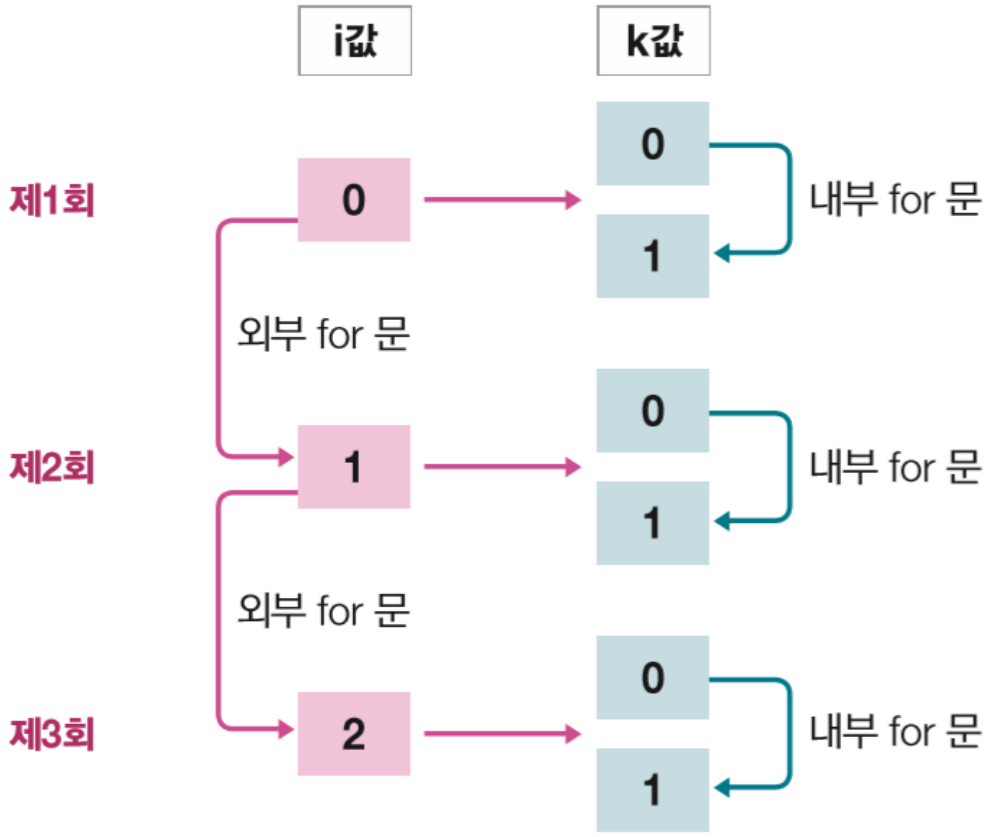


그림 6-3 중첩 for 문에서 i와 k값 변화

중첩 for 문의 활용

- 예 : 중첩 for 문 활용 2단부터 9단까지 구구단 출력

그림 6-4 구구단에서 i와 k 변수 추출

Section03 중첩 for 문

Code06-07.py

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

3행 : 2단에서 9단까지 반복
행 : 각 단의 뒷자리 숫자 1에서 9까지 반복

5행 : 구구단을 형식에 맞추어 출력
6행 : 각 단이 끝나면 한 줄 띄우려고 사용

출력 결과

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
… 중략 …
9 X 8 = 72
9 X 9 = 81
```

SELF STUDY 6-3

Code06-07.py를 각 단의 제목이 출력되도록 수정해 보자.

출력 결과

```
## 2단 ##
2 X 1 =  2
2 X 2 =  4
2 X 3 =  6
```

- **[프로그램 1]의 완성**
  - 가로 먼저 출력 : 일단 세로 방향으로 한 번 출력하면 다시 위로 올라가서 출력 불가

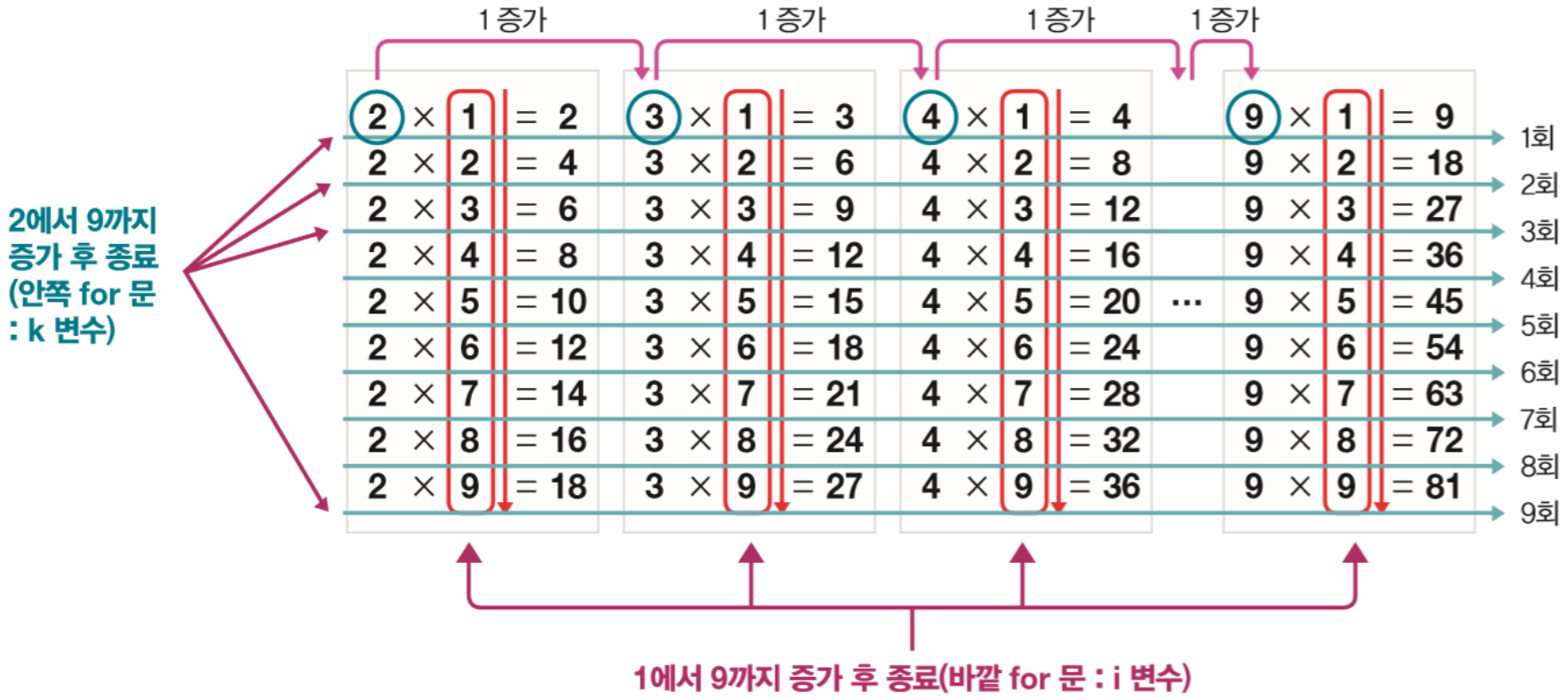


그림 6-5 구구단에서 i와 k 변수 추출(단 가로 먼저 출력)

Section03 중첩 for 문

Code06-08.py

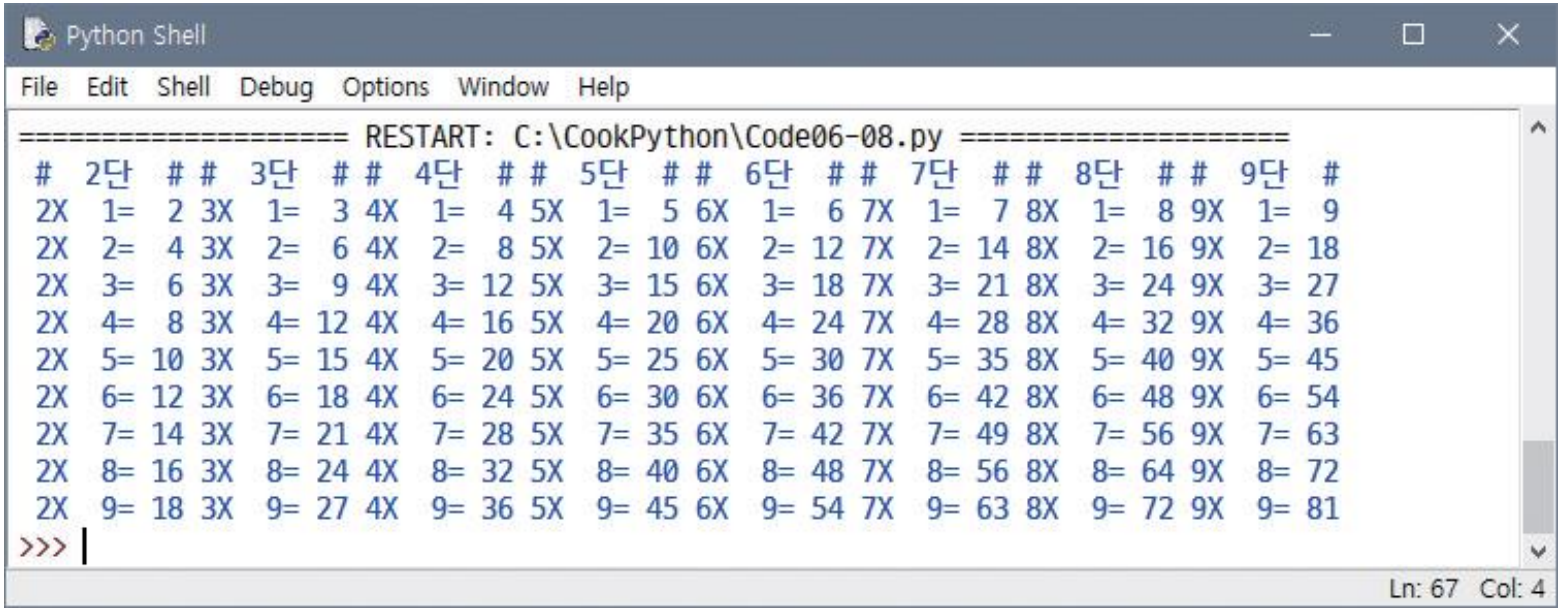
```
1  ## 전역 변수 선언 부분 ##
2  i, k, guguLine = 0, 0, ""
3
4  ## 메인 코드 부분 ##
5  for i in range(2, 10) :
6      guguLine = guguLine + ("#  %d단  #" % i)
7
8  print(guguLine)
9
10 for i in range(1, 10) :
11     guguLine = ""
12     for k in range(2, 10) :
13         guguLine = guguLine + str("%2dX %2d= %2d"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

2행 : 각 줄에 출력될 문자열 저장하는 guguLine 변수 준비

5~6행 : 맨 뒤의 단 제목 출력, '# 2단 #' 하나 출력, '# 3단 #' 하나 출력 아니라 guguLine에 각 단의 제목을 문자열로 모두 넣은 후 8행에서 한 번에 출력

10~14행 : 중첩 for 문으로 구구단 출력

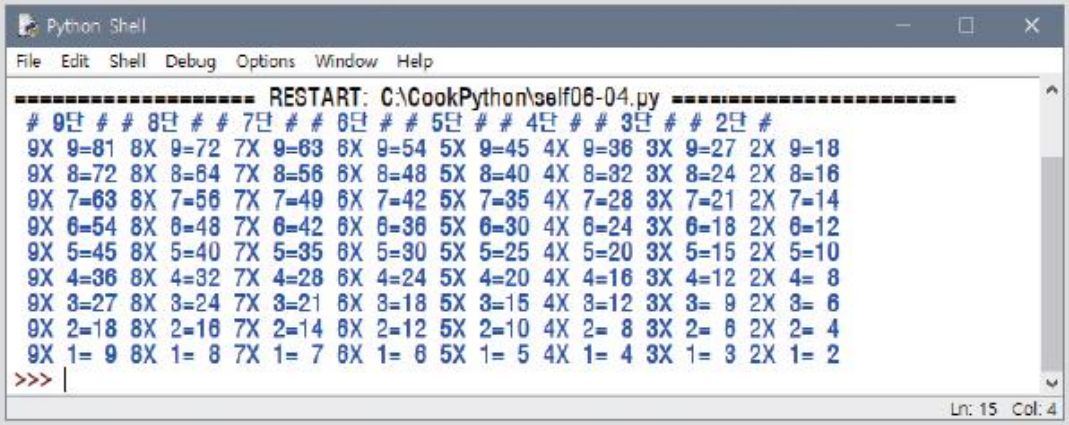
Section03 중첩 for 문



SELF STUDY 6-4

Code06-08.py를 구구단이 거꾸로 출력되도록 수정해 보자.

힌트 range( ) 함수의 값을 큰 값에서 작은 값으로 변경되도록 해야 한다.



for 문과 while 문 비교

- for 문의 형식

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

- for 문은 반복할 횟수를 range() 함수에서 결정 후 그 횟수만큼 반복, while 문은 반복 횟수를 결정하기보다는 조건식이 참일 때 반복하는 방식

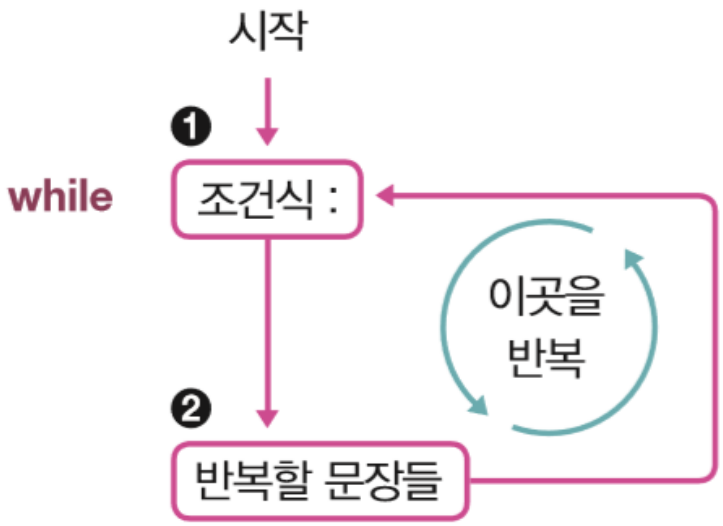


그림 6-6 while 문의 형식과 순서도

Section04 while 문

- for 문과 비슷하게 사용할 수 있는 while 문의 형식

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

- for 문으로 '안녕하세요?~' 문장을 3회 출력하는 코드

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

Section04 while 문

- 문장을 3회 반복하도록 while 문

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

for 문에서 사용한 변수와 시작값을 i=0으로 while 문 위에 작성
for 문의 끝값 while 문의 조건식인 i<3로 지정
for 문의 증가값 while 문의 마지막에 i=i+1 로 작성

출력 결과

```
0 : 안녕하세요? while 문을 공부 중입니다. ^^
1 : 안녕하세요? while 문을 공부 중입니다. ^^
2 : 안녕하세요? while 문을 공부 중입니다. ^^
```

Section04 while 문

- 예 : Code06-02(2).py에서 for 문으로 작성한 1에서 10까지의 합계 구하기

Code06-09.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 합계 : %d" % hap)
```

3행 : i의 시작값을 1로 지정
4행 : i가 11보다 작으면 참, i가 10일 때까지 5~6행 반복
5행 : hap에 i값(처음에는 1)을 누적
6행 : i를 1 증가

출력 결과

```
1에서 10까지의 합계 : 55
```

SELF STUDY 6-5

Code06-05.py를 while 문으로 수정해 보자.

힌트 Code06-09.py를 참고한다.

- **무한 루프를 하는 while 문**
  - 무한 루프 적용 : 'while 조건식 :'에 들어가는 조건식을 True로 지정

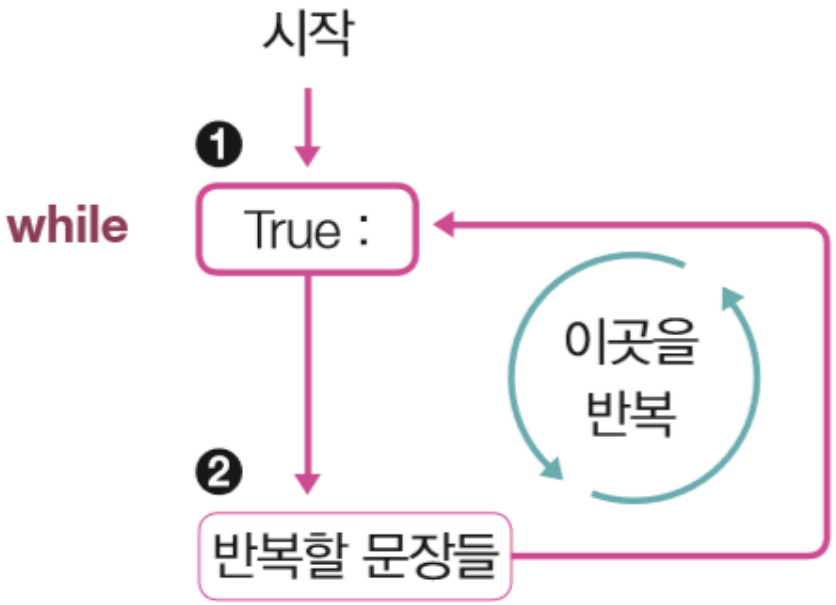


그림 6-7 while 문을 이용한 무한 루프

Section04 while 문

- 예 : 무한 루프

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

출력 결과

ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ~~~ 무한 반복

Section04 while 문

- 예 : 무한 루프를 사용해 입력한 두 숫자의 합계를 반복해서 계산

Code06-10.py

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

4행의 무한 반복문 때문에 사용자가 Ctrl + C 를 누를 때까지 5~8행 반복

출력 결과

```
더할 첫 번째 수를 입력하세요 : 55
더할 두 번째 수를 입력하세요 : 22
55 + 22 = 77
더할 첫 번째 수를 입력하세요 : 77
더할 두 번째 수를 입력하세요 : 128
77 + 128 = 205
더할 첫 번째 수를 입력하세요 :
```

Section04 while 문

- 예 : 사용자가 Ctrl + C 를 누를 때까지 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지까지 계산

Code06-11.py

```
1  ch = ""
2  a, b = 0, 0
3
4  while True :
5      a = int(input("계산할 첫 번째 수를 입력하세요 : "))
6      b = int(input("계산할 두 번째 수를 입력하세요 : "))
7      ch = input("계산할 연산자를 입력하세요 : ")
8
9      if (ch == "+") :
10         print("%d + %d = %d" % (a, b, a + b))
11     elif (ch == "-") :
12         print("%d - %d = %d" % (a, b, a - b))
13     elif (ch == "*") :
14         print("%d * %d = %d" % (a, b, a * b))
15     elif (ch == "/") :
16         print("%d / %d = %5.2f" % (a, b, a / b))
17     elif (ch == "%") :
18         print("%d %% %d = %d" % (a, b, a % b))
19     elif (ch == "//") :
20         print("%d // %d = %d" % (a, b, a // b))
21     elif (ch == "**") :
```

5~6행 : 두 숫자를 입력
7행 : 연산자를 입력

```
22          print("%d ** %d = %d" % (a, b, a ** b))
23      else :
24          print("연산자를 잘못 입력했습니다.")
```

출력 결과

```
계산할 첫 번째 수를 입력하세요 : 22
계산할 두 번째 수를 입력하세요 : 33
계산할 연산자를 입력하세요 : *
22 * 33 = 726
계산할 첫 번째 수를 입력하세요 : 10
계산할 두 번째 수를 입력하세요 : 4
계산할 연산자를 입력하세요 : %
10 % 4 = 2
계산할 첫 번째 수를 입력하세요 :
```

Section05 break 문과 continue 문

- **반복문을 탈출시키는 break 문**
  - 계속되는 반복을 논리 적으로 빠져나가는 방법

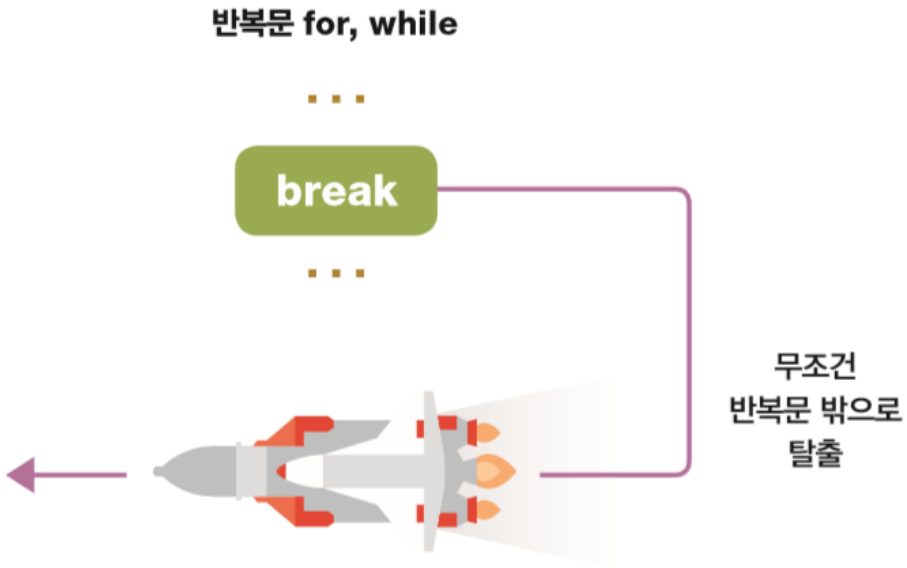


그림 6-8 break 문의 작동

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

출력 결과

```
for 문을 1번 실행했습니다.
```

Section05 break 문과 continue 문

- 예 : Code06-10.py를 break 문으로 첫 번째 수에 0이 입력될 때 자동으로 종료

Code06-12.py

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

4행 : 무한 반복 하도록 했다
5행 : a값을 입력
6행 : 입력한 a값이 0이면 7행 실행한 후 break 문으로 while 문을 탈출해 11행으로 건너뜀
11행에는 아무것도 없으므로 자연스럽게 12행 실행

출력 결과

```
더할 첫 번째 수를 입력하세요 : 55
더할 두 번째 수를 입력하세요 : 22
55 + 22 = 77
더할 첫 번째 수를 입력하세요 : 77
더할 두 번째 수를 입력하세요 : 128
77 + 128 = 205
더할 첫 번째 수를 입력하세요 : 0
0을 입력해 반복문을 탈출했습니다.
```

SELF STUDY 6-6

'$'를 입력하면 while 문을 빠져나가도록 Code06-12.py를 수정해 보자.

Section05 break 문과 continue 문

- 예 : 누적 합계(hap)가 1000 이상이 되는 시작 지점 알기

Code06-13.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 합계를 최초로 1000이 넘게 하는 숫자 : %d" % i)
```

3행 : i값이 1부터 100까지 변경되어 100회 실행하고, hap에 i값을 누적
6행 : hap이 1000보다 크거나 같으면 for 문을 탈출해서 8행으로
9행 : i값을 출력

출력 결과

```
1~100의 합계를 최초로 1000이 넘게 하는 숫자 : 45
```

SELF STUDY 6-7

Code06-13.py를 while 문으로 변경해 보자. 출력 결과는 동일하다.

Section05 break 문과 continue 문

- 반복문으로 다시 돌아가게 하는 continue 문

그림 6-9 continue 문의 작동

Section05 break 문과 continue 문

- 예: 1~100의 합계를 구하되, 3의 배수 (제외하고) 더하기

Code06-14.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
```

- **[프로그램 2]의 완성**
  - 별 모양의 글자 출력하는 코드

```
print('\u2605')
```

  - While 문으로 구현

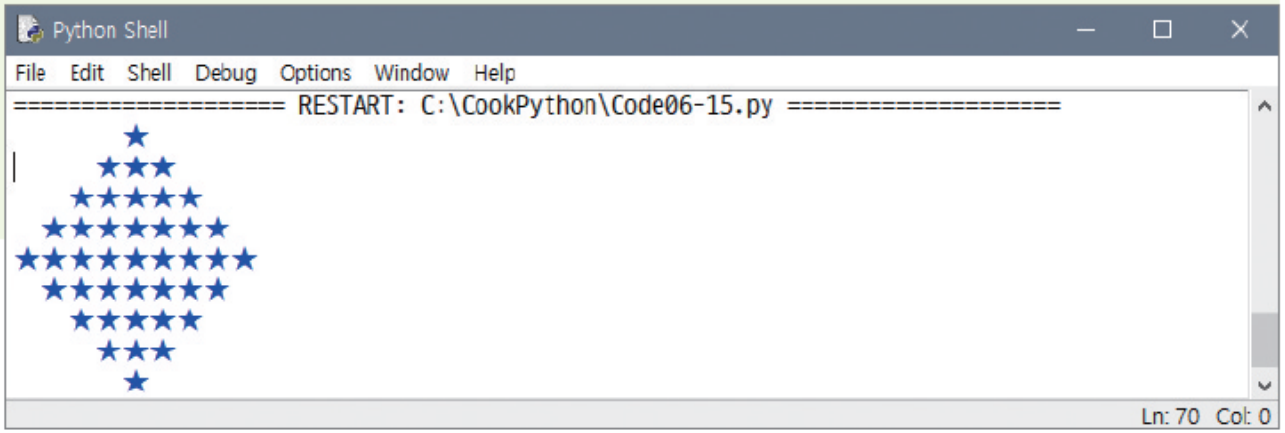
Code06-15.py

```
1  ## 전역 변수 선언 부분 ##
2  i, k = 0, 0
3
4  ## 메인 코드 부분 ##
5  i = 0
6  while i < 9 :
7      if i < 5 :
8          k = 0
9          while k < 4 - i :
10             print(' ', end = '')
11             k += 1
12         k = 0
13         while k < i * 2 + 1 :
14             print('\u2605', end = '')
```

6~26행 : 9번 반복되어 출력 줄이 9개 표시
다섯 줄은 7~15행이 출력하고, 나머지 네 줄은 16~24행이 출력
각 줄이 출력될 때 공백과 별의 개수는 9행, 13행, 18행, 22행이 결정

Section05 break 문과 continue 문

```
15              k += 1
16      else :
17          k = 0
18          while k < i - 4 :
19              print('  ', end = '')
20              k += 1
21          k = 0
22          while k < (9 - i) * 2 - 1 :
23              print('\u2605', end = '')
24              k += 1
25      print()
26      i += 1
```



SELF STUDY 6-8

Code06-15.py를 모두 for 문으로 변경해 보자. 출력은 하트 모양을 사용하는데, 하트 모양의 유니코드는 16진수 '2665'이다.