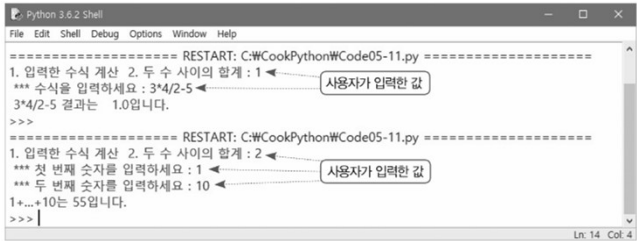


Section01 예제 프로그램

- [프로그램] 종합 계산기
- 기능이 두 가지인 종합 계산기 프로그램



Section02 기본 if 문

- if 문

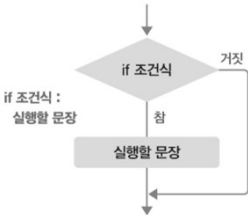


그림 5-1 if 문의 형식과 순서도

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

출력 결과
100보다 작군요.

Tip • 파이썬은 들여쓰기가 매우 중요. if 문 다음에 '실행할 문장'은 if 문 다음 줄에서 들여쓰기를 해서 작성. 들여쓰기 할 때는 Tab 보다 Space Bar 를 눌러 4칸 정도로 들여쓰기 권장, 대화형 모드에서는 '실행할 문장' 모두 끝나고 Enter 2번 눌러야 if 문이 끝나는 것으로 간주

Section02 기본 if 문

- if 문 실행 과정

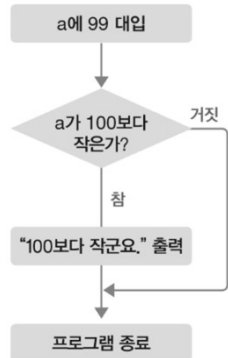


그림 5-2 if 문 실행 과정

Section02 기본 if 문

- 예 : 조건이 참이고 실행할 문장이 2개일 때

Code05-01.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

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

Section02 기본 if 문

예 : if 문에서 두 문장 이상을 실행하고자 할 때

Code05-02.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

프로그램 끝

Tip • 들여쓰기 오류 예

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

Section02 기본 if 문

if~else 문

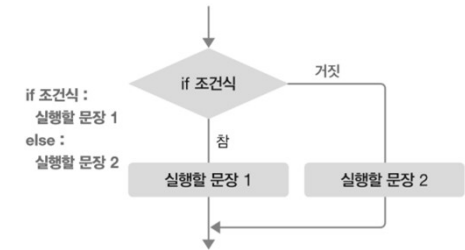


그림 5-3 if~else 문의 형식과 순서도

조건이 참일 때와 거짓일 때 실행할 문장이 다름

Section02 기본 if 문

Code05-03.py

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

출력 결과

100보다 크군요.

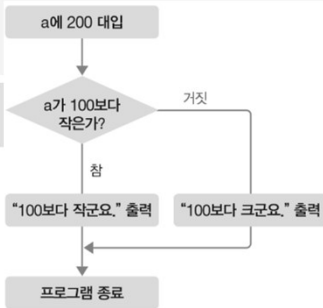


그림 5-4 Code05-03.py 실행 과정

Section02 기본 if 문

Code05-04.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

Section02 기본 if 문

- 예 : 입력 숫자가 짝수인지 홀수인지 계산

Code05-05.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
홀수를 입력했군요.

Section03 중첩 if 문

- if~else~if~else 문
- if 문을 한 번 실행한 후 그 결과에서 if 문을 다시 실행하는 것

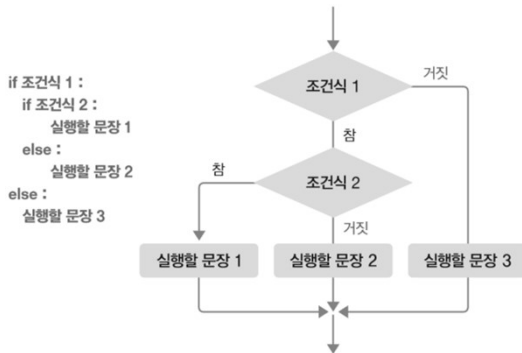


그림 5-5 중첩 if 문의 형식과 순서도

Section03 중첩 if 문

Code05-06.py

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

출력 결과

50보다 크고 100보다 작군요.

Section03 중첩 if 문

- Code05-07.py를 그림으로 표현

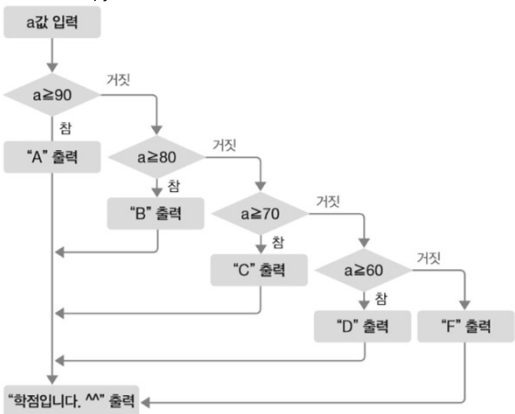


그림 5-6 Code05-07.py 실행 과정

Section03 중첩 if 문

if~elif~else 문

```
Code05-08.py
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. ^^")
```

Section03 중첩 if 문

SELF STUDY 5-1

Code05-08.py를 다음 조건처럼 좀 더 세분화해 보자.
95점 이상 : A+, 90점 이상 : A0, 85점 이상 : B+, 80점 이상 : B0,
75점 이상 : C+, 70점 이상 : C0, 65점 이상 : D+, 60점 이상 : D0, 60점 미만 : F



Section03 중첩 if 문

삼항 연산자를 사용한 if 문

```
1 jumsu = 55
2 res = ''
3 if jumsu >= 60 :
4     res = '합격'
5 else :
6     res = '불합격'
7 print(res)
```

3~6행 줄임

```
res = '합격' if jumsu >= 60 else '불합격'
```

Section04 if 문 응용

리스트와 함께 사용

- 리스트(List) : 데이터 여러 개를 한곳에 담아 놓은 것
- 방법 : 대괄호 []로 묶고 그 안에 필요한 것들을 한꺼번에 넣음
- 예 : fruit 변수에 값 4개를 리스트 하나로 묶어 대입

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

출력 결과
['사과', '배', '딸기', '포도']

추가

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

출력 결과
['사과', '배', '딸기', '포도', '귤']

Section04 if 문 응용

- if 항목 in 리스트 : 리스트에 해당 항목이 있다면 True를 반환

```
출력 결과
딸기가 있네요. ^^

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

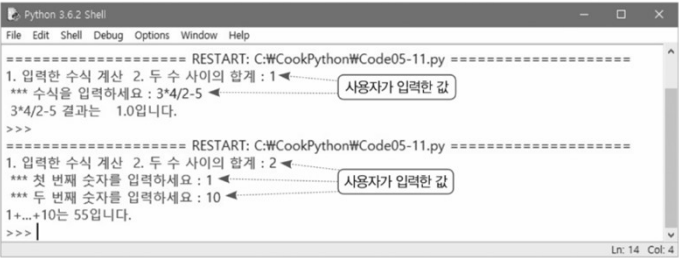
Section04 if 문 응용

- 예 : 0부터 9까지 숫자 중에서 리스트 안에 없는 숫자 찾기
Code05-10.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("숫자 %d는(은) 리스트에 없네요." %num)
```

```
출력 결과
생성된 리스트 [5, 8, 8, 7, 8, 1, 9, 0, 0, 4]
숫자 2는(은) 리스트에 없네요.
숫자 3는(은) 리스트에 없네요.
숫자 6는(은) 리스트에 없네요.
```

Section04 if 문 응용




```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookPython\Code05-11.py =====
1. 입력한 수식 계산 2. 두 수 사이의 합계 : 1
*** 수식을 입력하세요 : 3*4/2-5
3*4/2-5 결과는 1.0입니다.
>>>
===== RESTART: C:\CookPython\Code05-11.py =====
1. 입력한 수식 계산 2. 두 수 사이의 합계 : 2
*** 첫 번째 숫자를 입력하세요 : 1
*** 두 번째 숫자를 입력하세요 : 10
1+...+10는 55입니다.
>>>
```

Section04 if 문 응용

SELF STUDY 5-2

[프로그램 2]의 두 번째 기능처럼 두 숫자를 입력받고 두 숫자 사이의 합계를 구하는 프로그램을 만들어 보자. 단 1씩 증가하지 않고 증가하는 숫자도 입력받는다. 예를 들어 1, 100, 3을 입력하면 1+4+...+100의 합계를 구한다.

인 range(시작값, 끝값+1, 증가값) 형식으로 사용한다.



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookPython\self05-02.py =====
*** 첫 번째 숫자를 입력하세요 : 1
*** 두 번째 숫자를 입력하세요 : 100
*** 더할 숫자를 입력하세요 : 3
1+4+...+100는 1717입니다.
>>>
```

Section04 어떤 종류의 학생인지 맞추기

실습 내용

- 조건문을 이용하여 '어떤 종류의 학생인지 맞추는 프로그램'을 만들어 보자.
- 이 프로그램을 작성하는 규칙은 다음과 같다.

- 나이는 (2020 - 태어난 연도 + 1)로 계산
- 26세 이하 20세 이상이면 '대학생'
- 20세 미만 17세 이상이면 '고등학생'
- 17세 미만 14세 이상이면 '중학생'
- 14세 미만 8세 이상이면 '초등학생'
- 그 외의 경우는 '학생이 아닙니다.' 출력

```

=====
당신이 태어난 연도를 입력하세요.   ~ 입력 대기
1982                                ~ 당신이 태어난 연도 입력
학생이 아닙니다.                    ~ 어떤 종류의 학생인지 출력
=====
```

Section01 예제 프로그램

[프로그램 1] 구구단 출력

- for 문을 사용

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
=====
# 2단 ## 3단 ## 4단 ## 5단 ## 6단 ## 7단 ## 8단 ## 9단 #
2X 1= 2 3X 1= 3 4X 1= 4 5X 1= 5 6X 1= 6 7X 1= 7 8X 1= 8 9X 1= 9
2X 2= 4 3X 2= 6 4X 2= 8 5X 2= 10 6X 2= 12 7X 2= 14 8X 2= 16 9X 2= 18
2X 3= 6 3X 3= 9 4X 3= 12 5X 3= 15 6X 3= 18 7X 3= 21 8X 3= 24 9X 3= 27
2X 4= 8 3X 4= 12 4X 4= 16 5X 4= 20 6X 4= 24 7X 4= 28 8X 4= 32 9X 4= 36
2X 5= 10 3X 5= 15 4X 5= 20 5X 5= 25 6X 5= 30 7X 5= 35 8X 5= 40 9X 5= 45
2X 6= 12 3X 6= 18 4X 6= 24 5X 6= 30 6X 6= 36 7X 6= 42 8X 6= 48 9X 6= 54
2X 7= 14 3X 7= 21 4X 7= 28 5X 7= 35 6X 7= 42 7X 7= 49 8X 7= 56 9X 7= 63
2X 8= 16 3X 8= 24 4X 8= 32 5X 8= 40 6X 8= 48 7X 8= 56 8X 8= 64 9X 8= 72
2X 9= 18 3X 9= 27 4X 9= 36 5X 9= 45 6X 9= 54 7X 9= 63 8X 9= 72 9X 9= 81
>>> |
```

Section01 예제 프로그램

[프로그램 2] 마름모 모양 출력

- while 문 활용

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
=====
|
  ★
 ★★
 ★★★
★★★★
★★★★★
★★★★★★
★★★★★★★
★★★★★★★
★★★★★
 ★★
  ★
=====
Ln: 70 Col: 0
```

Section05 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. ^^
```

Section05 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. ^^")
```

Section05 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. ^^")
```

Section05 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
```

Section05 기본 for 문

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

Section05 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      4행 'hap=hap+i'에서 hap에 어떤 값이 있어야 다시 누적
5                       hap 자체가 존재하지 않아 더할 것이 없어서 오류가 발생
6 print("1에서 10까지의 합계 : %d" % hap)
```

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

Section05 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      주의 : i 변수와 hap 변수의 값
5
6 print("1에서 10까지의 합계 : %d" % hap)
```

출력 결과

1에서 10까지의 합계 : 55

Section05 기본 for 문

- i 와 hap 변수값의 변화

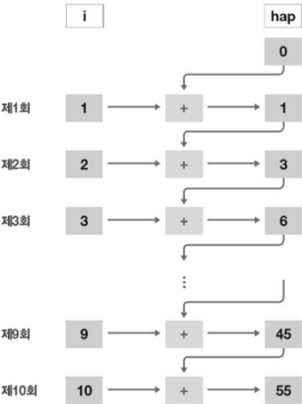


그림 6-1 i 와 hap 변수값의 변화

Section05 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

Section05 기본 for 문

- 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

Section05 기본 for 문

- 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

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

Section05 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))
```

출력 결과

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

Section05 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))
```

출력 결과

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

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

Section06 중첩 for 문

중첩 for 문의 개념

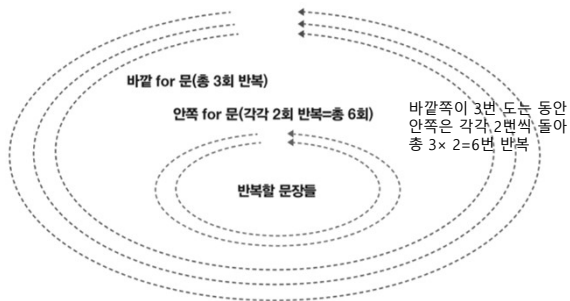


그림 6-2 중첩 for 문의 작동 개념

Section06 중첩 for 문

중첩 for 문의 기본 형식

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

출력 결과

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

Section06 중첩 for 문

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

- 1 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
- 2 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
- 3 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

Section06 중첩 for 문

중첩 for 문에서 i와 k값 변화



그림 6-3 중첩 for 문에서 i와 k값 변화

Section06 중첩 for 문

중첩 for 문의 활용

예 : 중첩 for 문 활용 2단부터 9단까지 구구단 출력



그림 6-4 구구단에서 i와 k 변수 추출

Section06 중첩 for 문

Code06-07.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("")
```

3행 : 2단에서 9단까지 반복
4행 : 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

Section06 중첩 for 문

[프로그램 1]의 완성

가로 먼저 출력 : 일단 세로 방향으로 한 번 출력하면 다시 위로 올라가서 출력 불가

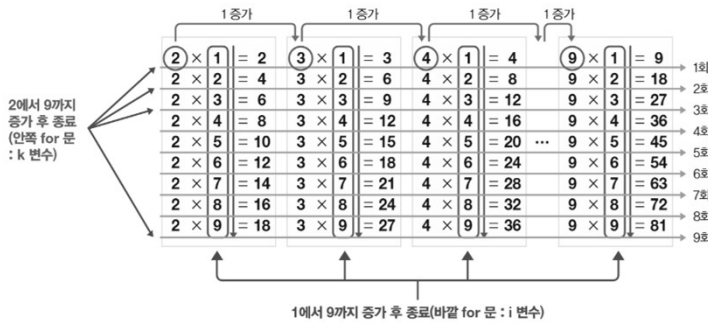


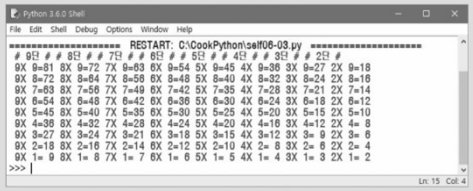
그림 6-5 구구단에서 i와 k 변수 추출(단 가로 먼저 출력)

Section06 중첩 for 문

SELF STUDY 6-3

Code06-08.py를 구구단이 거꾸로 출력되도록 수정해 보자.

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



Section07 while 문

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

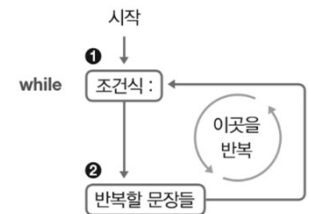


그림 6-6 while 문의 형식과 순서도

Section07 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)
```

Section07 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. ^^

Section07 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 6-4

Code06-05.py를 while 문으로 수정해 보자.
힌트 Code06-09.py를 참고한다.

Section07 while 문

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

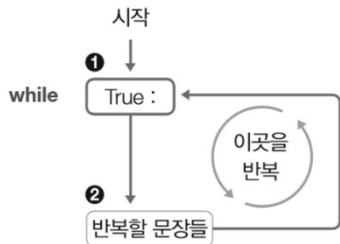


그림 6-7 while 문을 이용한 무한 루프

Section07 while 문

- 예 : 무한 루프

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

출력 결과
ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ㅋ ~~~ 무한 반복

Section07 while 문

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

Code06-10.py

```
1 hap = 0
2 a, b = 0, 0    4행의 무한 반복문 때문에 사용자가 Ctrl + C 를 누를 때까지 5~8행 반복
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))
```

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 :

Section07 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 : "))    5~6행 : 두 숫자를 입력
7     ch = input("계산할 연산자를 입력하세요 : ")           7행 : 연산자를 입력
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 == "**") :
```

Section07 while 문

```
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 :

Section07 for문과 while문 상호 변환 가능

여기서 잠깐!

- for문과 while문은 기본적으로 유사하며, 서로 변환이 가능하다. 하지만 두 구문의 쓰임에는 차이가 있다. For문은 일반적으로 반복 횟수를 정확하게 알고 있고, 반복 횟수가 변하지 않을 때 사용한다. 반면, while문은 반복 실행 횟수가 명확하지 않고 어떤 조건을 만족하면 프로그램을 종료하고자 할 때 사용한다.
- 예를 들어, 학생들의 성적을 채점하는 프로그램을 작성한다고 하자. 이미 학생이 총 몇 명인지 명확하게 알고 있으므로 for문을 사용하는 것이 좋다. 하지만 가위바위보를 한다고 가정했을 때 '이기면 종료하라'라는 조건을 주면 언제 이길지 모르므로 while문을 사용하는 것이 낫다.

```
for i in range(0.5):
    print (i)
```

(a) 반복 실행 횟수를 명확히할 때

```
i = 0
while i < 5:
    print(i)
    i = i + 1
```

(b) 반복 실행 횟수가 명확하지 않을 때

Section08 break 문과 continue 문

반복문을 탈출시키는 break 문

- 계속되는 반복을 논리 적으로 빠져나가는 방법

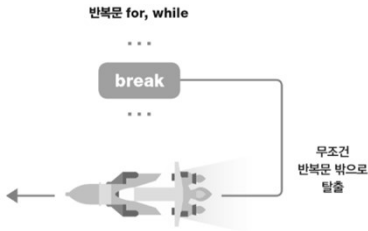


그림 6-8 break 문의 작동

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

출력 결과

for 문을 1번 실행했습니다.

Section08 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 건너뛴다
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.

Section08 break 문과 continue 문

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

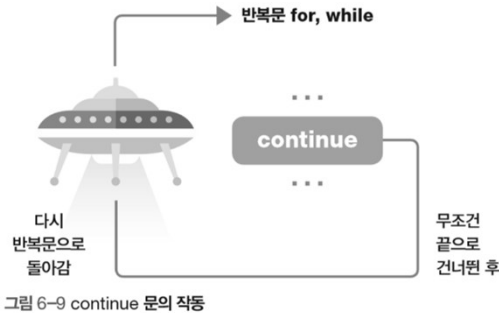
```
Code06-13.py
1 hap, i = 0, 0
2
3 for i in range(1, 101):
4     hap += i      3행 : i값이 1부터 100까지 변경되어 100회 실행하고, hap에 i값을 누적
5                 6행 : hap이 1000보다 크거나 같으면 for 문을 탈출해서 8행으로
6     if hap >= 1000: 9행 : i값을 출력
7         break
8
9 print("1~100의 합계를 최초로 1000이 넘게 하는 숫자 : %d" % i)
```

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

SELF STUDY 6-6
Code06-13.py를 while 문으로 변경해 보자. 출력 결과는 동일하다.

Section08 break 문과 continue 문

- 반복문으로 다시 돌아가게 하는 continue 문



Section08 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

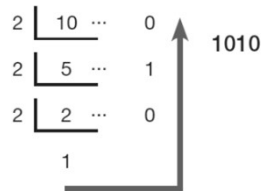
Section08 LAB 문자열 역순 출력

- reverse_sentence : 입력된 문자열을 역순으로 출력하는 변수이다.

Loop	reverse_sentence ¹	reverse_sentence ²	char
0		I	I
1	I	I	
2	I	II	I
3	II	oI	o
4	oI	volI	v
5	volI	evolI	e
6	evolI	evolI	
7	evolI	yevolI	y
8	yevolI	oyevolI	o
9	oyevolI	uoyevolI	u

Section08 십진수를 이진수로 변환

- 십진수 숫자를 2로 계속 나눈 후, 그 나머지를 역순으로 취하면 이진수가 된다.



Loop	decimal ¹	remainder	decimal ²	result ¹	result ²
0	10	0	5		0
1	5	1	2	0	10
2	2	0	1	10	010
3	1	1	0	010	1010

61/63

Section08 숫자 찾기 게임

■ 실습 내용

- 이번 Lab에서는 지금까지 배운 반복문과 조건문을 토대로 숫자 찾기 게임 프로그램을 만들어 보자.
- 이 프로그램의 규칙은 다음과 같다.

- 먼저 컴퓨터가 1에서 100까지 중 임의의 숫자를 생성한다.
- 다음으로 사용자가 추측하는 숫자를 입력하면 컴퓨터가 생성한 임의의 숫자보다 큰지, 작은지를 계속 비교해 준다.
- 정답을 맞힐 때까지 계속하다가 맞으면 '정답입니다. 입력한 숫자는 n입니다.'를 출력한다.

62/63

Section08 숫자 찾기 게임

```
숫자를 맞춰 보세요. (1 ~ 100)
230
숫자가 너무 큼니다.
20
숫자가 너무 큼니다.
10
숫자가 너무 큼니다.
1
숫자가 너무 작습니다.
3
정답입니다. 입력한 숫자는 3입니다.
```

63/63