



Python Programming for Beginners

04 Conditional Statement (조건문)

2023학년도 2학기
Suk-Hwan Lee

**Computer Engineering
Artificial Intelligence**

Creating the Future

Dong-A University

**Division of Computer Engineering &
Artificial Intelligence**

Section01 기본 if 문

■ if 문

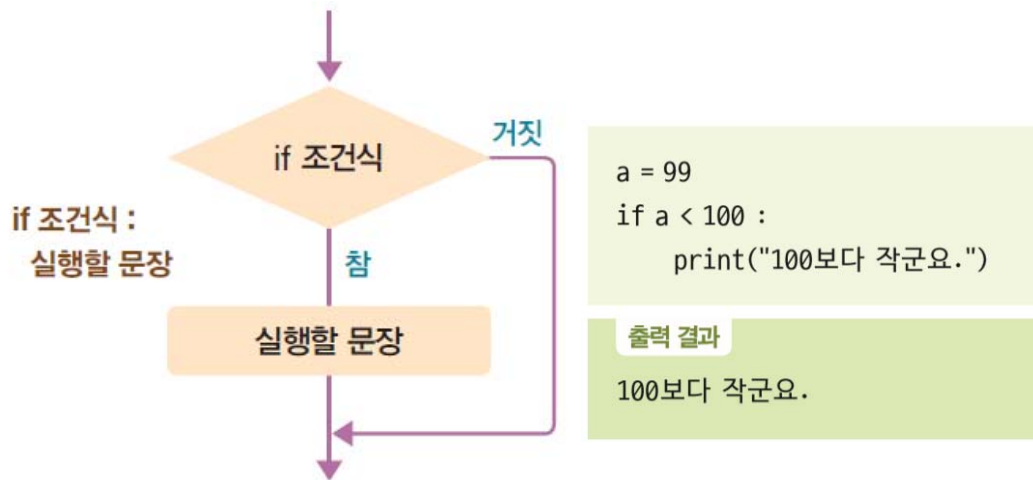


그림 5-1 if 문의 형식과 순서도

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

■ if 문 실행 과정

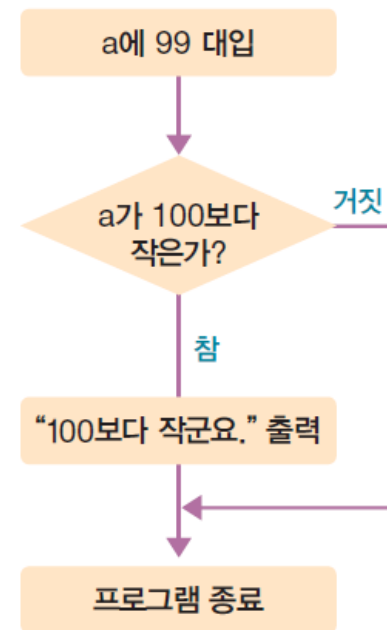


그림 5-2 if 문 실행 과정

Section01 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")
```

- 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?")
```

Section01 기본 if 문

■ if~else 문

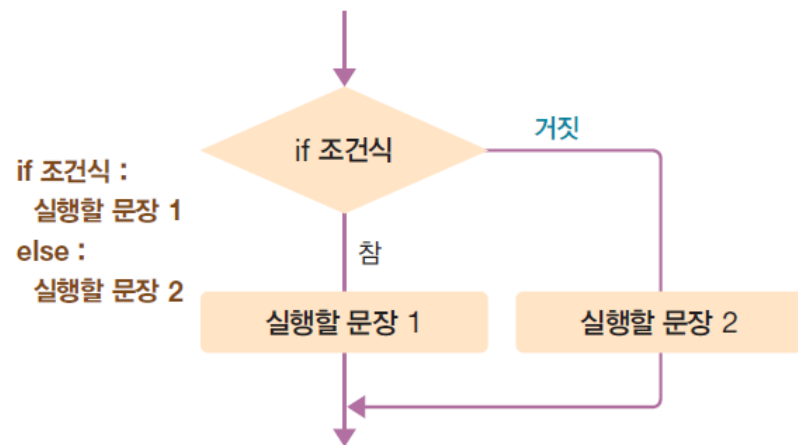


그림 5-3 if~else 문의 형식과 순서도

- 조건이 참일 때와 거짓일 때 실행할 문장이 다름

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.

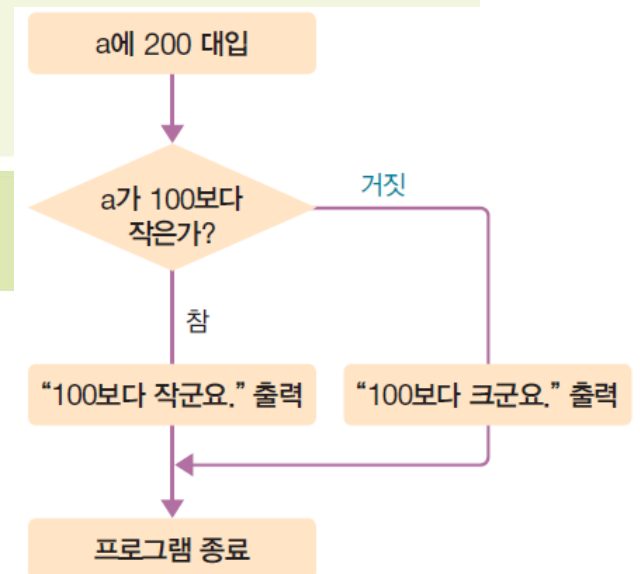


그림 5-4 Code05-03.py 실행 과정

Section01 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
```

- 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.
```

- 조건이 참일 때와 거짓일 때 실행할 문장이 다름

■ **조건문의 판단 : 비교 연산자**

- 비교 연산자(또는 조건 연산자): 어떤 것이 큰지 작은지 같은지를 비교하는 것으로, 그 결과는 참(True)이나 거짓(False)이 된다.

비교 연산자	비교 상태	설명
<code>x < y</code>	~보다 작음	x가 y보다 작은지 검사
<code>x > y</code>	~보다 큼	x가 y보다 큰지 검사
<code>x == y</code>	같음	x와 y의 값이 같은지 검사
<code>x is y</code>	같음(메모리 주소)	x와 y의 메모리 주소가 같은지 검사
<code>x != y</code>	같지 않음	x와 y의 값이 같지 않은지 검사
<code>x is not y</code>	같지 않음(메모리 주소)	x와 y의 메모리 주소가 같지 않은지 검사
<code>x >= y</code>	크거나 같음	x가 y보다 크거나 같은지 검사
<code>x <= y</code>	작거나 같음	x가 y보다 작거나 같은지 검사

- 같음을 의미하는 = 연산자와 비교하여, 할당의 의미로 같음을 표현할 때는 ==과 같은 새로운 연산자를 사용한다.
- 다음 코드에서 두 값이 모두 같으니 결과는 True이다. 사실 조건문 코드를 볼 때, 언제나 이러한 코드가 True 또는 False로 치환된다고 생각하면 이해하기 쉽다.

```
>>> 7 == 7
```

[비교 연산자]

Section02 조건문

■ 조건의 판단 : 비교 연산자

- is 연산자는 ==처럼 두 변수가 같음을 비교하지만, ==과 다르게 메모리의 주소를 비교한다.
- 파이썬은 처음 인터프리터를 시작할 때 -5~256까지 변하지 않는 메모리(정적 메모리) 주소에 값을 할당한다. 그리고 해당 값을 다른 변수가 사용할 때, 그 메모리 주소를 반환한다.
- 다음 코드에서 a와 b, 둘 다 100일 때는 is와 ==이 모두 같다고 나오지만, 둘 다 300일 경우에는 값만 같고 메모리 주소는 다르다고 나온다. is not도 마찬가지로 사용된다.

```
a = 100  
b = 100
```

```
a == b
```

```
True
```

```
a is b
```

```
True
```

```
print(id(a))  
print(id(b))
```

```
140710653078368
```

```
140710653078368
```

```
a = 300  
b = 300
```

```
a == b
```

```
True
```

```
a is b
```

```
False
```

```
print(id(a))  
print(id(b))
```

```
2116774623600
```

```
2116774623440
```

- Python은 C 언어와 달리 포인터라는 개념이 없다. 하지만 내부적으로는 메모리를 사용하고, 이에 대한 주소 정보를 가지고 있음
- Python에서 해당 메모리 주소는 변수를 구별하기 위한 용도로 사용하고 있음
- Python의 built-in 함수 중 `id()` 를 이용하면 현재 확인하고자 하는 변수의 메모리 주소를 확인할 수 있다. (CPython의 경우 메모리 주소를 반환하지만, 다른 파이썬 인터프리터는 메모리 주소 이외의 다른 값을 반환할 수 있음)

- `id(object)`
- Return the “identity” of an object. This is an integer which is guaranteed to be [unique and constant for this object during its lifetime](#). Two objects with non-overlapping lifetimes may have the same `id()` value.
- CPython implementation detail: [This is the address of the object in memory](#).

Section02 조건문

■ 조건의 판단 : True와 False의 치환

- 컴퓨터는 기본적으로 이진수만 처리할 수 있으며, True는 1로, False는 0으로 처리한다.
- 아래 코드를 실행하면 True가 출력된다. 그 이유는 앞서 설명한 것처럼 컴퓨터는 존재하면 True, 존재하지 않으면 False로 처리하기 때문이다.

```
>>> if 1: print("True")  
... else: print("False")
```

- 아래 코드를 실행하면 True가 출력된다. 먼저 $3 > 5$ 는 False이고 False는 결국 0으로 치환된다. 그래서 이것을 다시 치환하면 $(0) < 10$ 이 되고, 이 값은 참이므로 True가 반환된다.

```
>>> (3 > 5) < 10
```


Section02 조건문

■ 조건의 판단 : 논리 연산자

- 논리 연산자는 and · or · not문을 사용해 조건을 확장할 수 있다.
- and는 둘 다 참이어야 True, or는 둘 중 하나만 참이어도 True, not은 참이면 False이고 거짓이면 True를 출력한다.

연산자	설명	예시
and	두 값이 모두 참일 경우 True, 그렇지 않을 경우 False	(7 > 5) and (10 > 5)는 True (7 > 5) and (10 < 5)는 False
or	두 값 중 하나만 참일 경우 True, 두 값 모두 거짓일 경우 False	(7 < 5) or (10 > 5)는 True (7 < 5) or (10 < 5)는 False
not	값을 역으로 반환하여 판단	not (7 < 5)는 True not (7 > 5)는 False

[논리 연산자]

```
>>> a = 8
>>> b = 5
>>> a == 8 and b == 4
False
>>> a > 7 or b > 7
True
>>> not (a > 7)
False
```

Section03 중첩 if 문

■ if~else~if~else 문

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

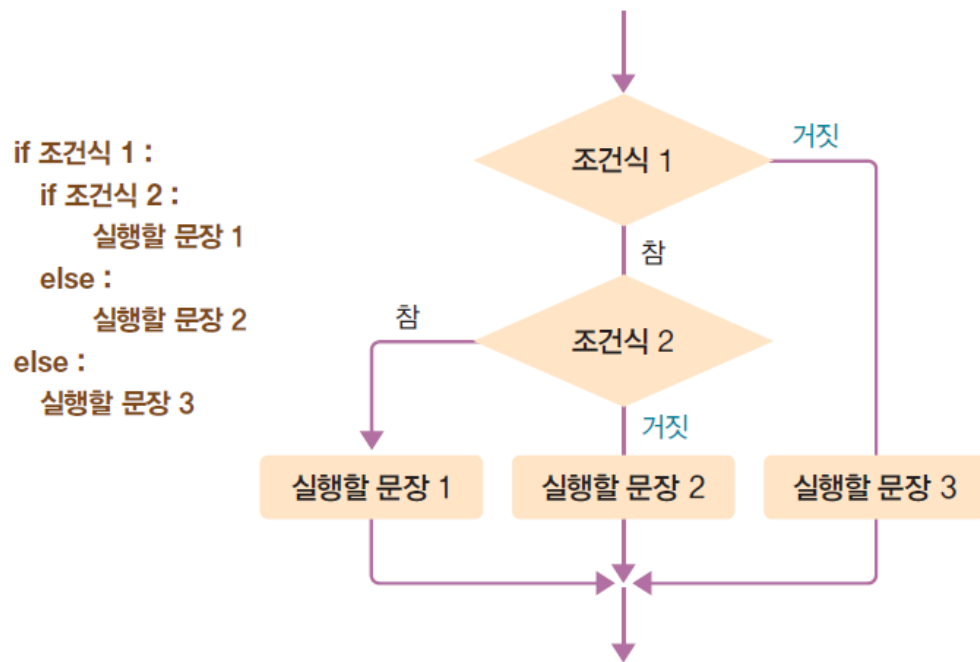


그림 5-5 중첩 if 문의 형식과 순서도

Code05-06.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.

Section03 중첩 if 문

- Code05-07.py를 그림으로 표현

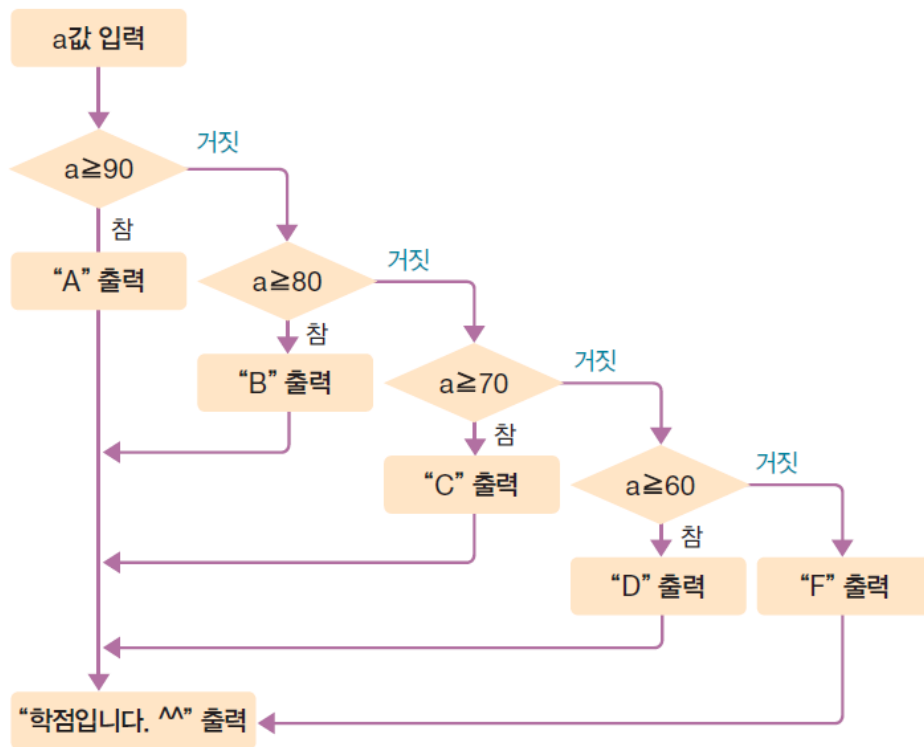


그림 5-6 Code05-07.py 실행 과정

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. ^^")
```

- python tutorial if statement

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

There can be zero or more `elif` parts, and the `else` part is optional. The keyword `'elif'` is short for `'else if'`, and is useful to avoid excessive indentation. An `if ... elif ... elif ...` sequence is a substitute for the `switch` or `case` statements found in other languages.

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'
```

Section03 중첩 if 문

- if 문을 사용해 터틀 그래픽에서 무지개 색상 의 원을 그리는 프로그램

Code05-09.py

```
1 import turtle
2
3 ## 전역 변수 선언 부분 ##
4 swidth, sheight = 500, 500
5
6 ## 메인 코드 부분 ##
7 turtle.title('무지개색 원그리기')
8 turtle.shape('turtle')
9 turtle.setup(width = swidth + 50, height = sheight + 50)
10 turtle.screensize(swidth, sheight)
11 turtle.penup()
12 turtle.goto(0, -sheight / 2)
13 turtle.pendown()
14 turtle.speed(10)
15
```

4행: 창 크기에 사용할 변수 준비

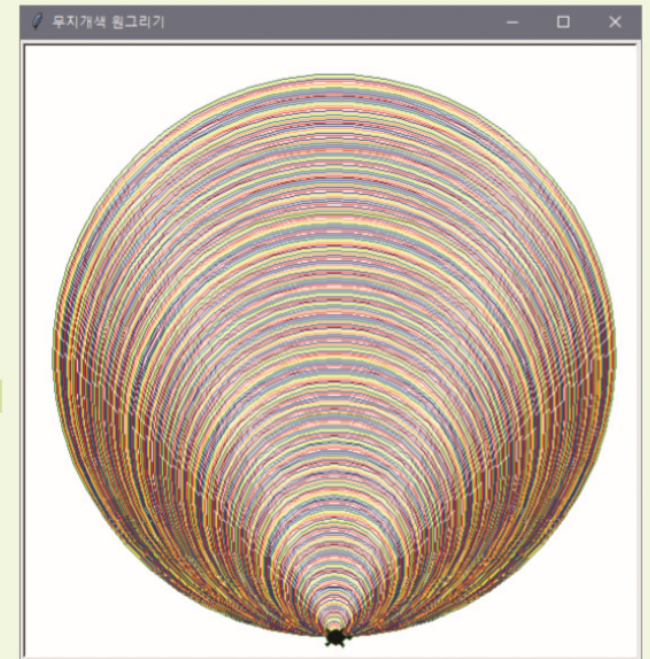
7~10행: 윈도우창 설정

11~13행: 거북이를 가운데 아래쪽으로 이동

14행: 거북이 속도 설정

```
16 for radius in range(1, 250) :
17     if radius % 6 == 0 :
18         turtle.pencolor('red')
19     elif radius % 5 == 0 :
20         turtle.pencolor('orange')
21     elif radius % 4 == 0 :
22         turtle.pencolor('yellow')
23     elif radius % 3 == 0 :
24         turtle.pencolor('green')
25     elif radius % 2 == 0 :
26         turtle.pencolor('blue')
27     elif radius % 1 == 0 :
28         turtle.pencolor('navyblue')
29     else :
30         turtle.pencolor('purple')
31
32     turtle.circle(radius)
33
34 turtle.done()
```

16~32행: 반지름(radius) 1에서 249까지 원을 반복해 그림
17~30행: 반지름에 따라 빨주노초파남보 색상이 반복 설정
32행: 거북이 원을 그림



Section03 중첩 if 문

■ if 문을 사용해 터틀 그래픽에서 무지개 색상 의 원을 그리는 프로그램

CookPython(2019.10.15) > Code05-09.py > ...

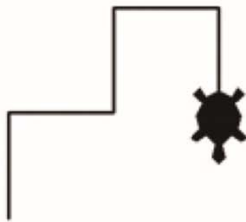
```
1  import turtle
2  import random
3
4  ## 전역 변수 선언 부분 ##
5  swidth, sheight = 500, 500
6
7  turtle.title('무지개색 원그리기')
8  turtle.setup(width = swidth + 50, height = sheight + 50)
9  turtle.screensize(swidth, sheight)
10
11 ## 메인 코드 부분 ##
12 myT = turtle.Turtle()
13 myT.shape('turtle')
14 myT.penup()
15 myT.goto(0, -sheight / 2)
16 myT.pendown()
17 myT.speed(10)
18
19 for i in range(50):
20     radius = random.randrange(30, 256)
21     if radius % 6 == 0 :
22         myT.pencolor('red')
23     elif radius % 5 == 0 :
24         myT.pencolor('orange')
25     elif radius % 4 == 0 :
26         myT.pencolor('yellow')
27     elif radius % 3 == 0 :
28         myT.pencolor('green')
29     elif radius % 2 == 0 :
30         myT.pencolor('blue')
31     elif radius % 1 == 0 :
32         myT.pencolor('navyblue')
33     else :
34         myT.pencolor('purple')
35
36     myT.circle(radius)
37
38 turtle.done()
```

Section03 중첩 if 문

LAB⁴⁻³ 거북이 제어하기

파이썬 셸에서 left를 의미하는 "l"을 입력하면 거북이가 왼쪽으로 100픽셀 이동하고 right를 의미하는 "r"을 입력하면 거북이가 오른쪽으로 100픽셀 이동하는 프로그램을 작성하여 보자.

원하는 결과



```
File Edit Shell Debug Options Window Help
===== RESTART: E:/5.py =====
명령을 입력하시오: l
명령을 입력하시오: r
명령을 입력하시오: l
명령을 입력하시오: r
명령을 입력하시오: r
명령을 입력하시오: r
Ln: 156 Col: 11
```

```
import turtle
```

```
t = turtle.Turtle()
# 커서의 모양을 거북이로 한다.
t.shape("turtle")
```

```
# 거북이가 그리는 선의 두께를 3으로 한다.
```

```
t.width(3)
```

```
# 거북이를 3배 확대한다.
```

```
t.shapesize(3, 3)
```

```
# 무한 루프로 진입한다. 이 루프는 Ctrl+C를 입력받아 종료된다.
```

```
while True:
```

```
    command = input("명령을 입력하시오: ")
```

```
    if command == "l":           # 사용자가 "l"을 입력하였으면
                                # 왼쪽으로 90도 회전
```

```
        t.left(90)
```

```
        t.forward(100)
```

```
    if command == "r":           # 사용자가 "r"을 입력하였으면
                                # 오른쪽으로 90도 회전
```

```
        t.right(90)
```

```
        t.forward(100)
```

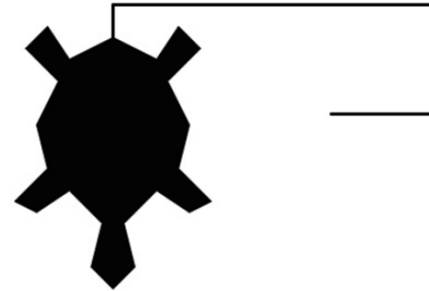

Section03 중첩 if 문

도전문제



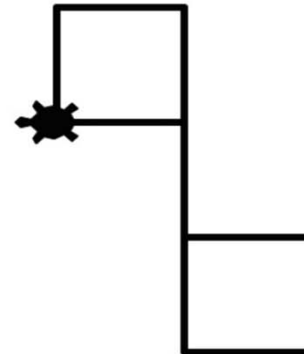
도전문제 4.7

- (1) 사용자가 'f'를 입력하면 현재 방향에서 100픽셀 전진하도록 소스를 약간 변경해보자.
- (2) 사용자가 'h'를 입력하면 헬크모드가 되어 오른쪽 그림과 같이 크기가 가로 세로로 10배 커지도록 하자. 반대로 'n'을 입력하면 원래의 크기인 가로, 세로로 3배 커진 상태가 되도록 하자.
- (3) 정수 1에서 9사이의 값을 입력으로 받아서 입력값에 따라서 거북이의 크기가 원래 크기의 1에서 9배 사이로 커지도록 하자.



도전문제 4.8

`random.randrange(2)`를 이용하여 10개의 난수를 생성하도록 하자. `random` 모듈의 `randrange(2)` 함수는 0과 1 중에서 하나의 정수를 생성한다. `if-else`문을 10개 사용하여 0이 나타날 경우 `right`로 50픽셀 이동하고 1이 나타날 경우 `left`로 50픽셀을 이동하도록 하여라. 이와 같이 자유롭게 10회 이동한 후 거북이 그래픽이 그리는 그림을 다음과 같이 보여라.



Section03 중첩 if 문



잠깐 - 여러가지 터틀 그래픽 명령어를 알아보자.

지금까지 우리는 간단한 터틀 그래픽 명령어인 forward(), left(), right()등을 알아보았다. 터틀 그래픽은 이것보다 훨씬 많은 다양한 기능이 있다. 아래 표는 터틀 그래픽에서 사용할 수 있는 명령어와 하는 일이다.

명령	하는 일
begin_fill() ... end_fill()	begin_fill()과 end_fill() 사이의 코드에 나타난 부분을 색칠한다. ... 으로 표시된 부분에는 터틀의 좌표나 모양을 기술할 수 있다.
color(c)	터틀의 색깔을 변경한다. c 값으로 'red', 'green', 'blue', 'black', 'gray', 'pink'...등의 여러가지 색상을 선택할 수 있다.
shape(s)	터틀의 모양을 변경한다. s 값으로는 'arrow', 'turtle', 'circle', 'square', 'triangle', 'classic' 등이 있다.
shapex(s), shapex(w, h)	터틀의 크기를 변경한다.
pos(), position()	터틀의 현재 위치를 구한다.
xcor()	터틀의 x 좌표를 구한다.
ycor()	터틀의 y 좌표를 구한다.
heading()	터틀이 현재 바라보는 각도를 구한다.
distance(x, y)	현재 터틀이 있는 위치에서 특정 위치까지의 거리를 구한다.
penup(), pu(), up()	펜을 올린다(그림을 그릴 수 없는 상태로 만든다).
pendown() , pd(), down()	펜을 내린다(그릴 수 있는 상태로 만든다).
pensize(w), width(w)	펜 굵기를 변경한다.
circle(r)	현재 위치에서 지정된 r 값 반지름 크기를 가지는 원을 그린다.
goto(x, y), setpos(x,y), setposition(x,y)	커서를 특정 위치(좌표)로 보낸다. 이때 penup() 상태이면 선이 그려지지 않으며, pendown() 상태이면 선이 그려진다.
stamp()	현재 커서의 위치에 지정된 크기와 색상, 모양의 터틀을 표시한다.
home()	터틀의 위치와 방향을 초기화한다.
textinput()	텍스트 입력을 받는 대화창을 표시하고 이 창에서 문자열을 입력 받는다.(주의: 반드시 turtle.textinput() 과 같이 사용)

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']
```

- 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

```
import random

numbers=[]
for num in range(0, 10) :
    numbers.append(random.randrange(0, 10))

print("생성된 리스트", numbers)

for num in range(0, 10) :
    if num not in numbers :
        print("숫자 %d는(은) 리스트에 없네요." %num)
```

생성된 리스트 [9, 2, 4, 7, 1, 1, 8, 2, 0, 4]

숫자 3는(은) 리스트에 없네요.

숫자 5는(은) 리스트에 없네요.

숫자 6는(은) 리스트에 없네요.

3행 : 빈 리스트인 numbers 준비

4행과 9행 : 각각 0부터 9까지 총 10회를 반복

5행 : 0~9의 숫자 총 10개를 numbers 리스트에 만들

7행 : 생성된 리스트 출력

9행 : 0부터 9까지의 숫자를 num에 넣음

10행 : numbers 리스트에 해당 숫자가 없다면 11행에서 숫자 없다는
메시지 출력

Section04 if 문 응용

■ 기능이 두 가지인 종합 계산기 프로그램

CookPython(2019.10.15) > Code05-11.py > ...

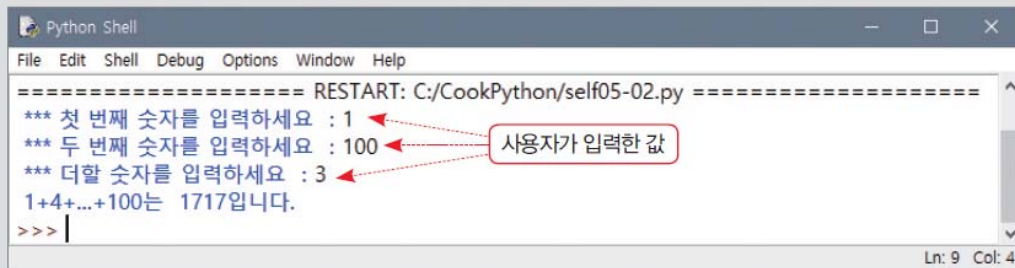
```
1  ## 변수 선언 부분 ##
2  select, answer, numStr, num1, num2 = 0, 0, "", 0, 0
3
4  ## 메인 코드 부분 ##
5  select = int(input("1. 입력한 수식 계산 2. 두 수 사이의 합계 : "))
6
7  if select == 1 :
8      numStr = input(" *** 수식을 입력하세요 : ")
9      answer = eval(numStr)
10     print(" %s 결과는 %.1f입니다. " %(numStr, answer))
11 elif select == 2 :
12     num1 = int(input(" *** 첫 번째 숫자를 입력하세요 : "))
13     num2 = int(input(" *** 두 번째 숫자를 입력하세요 : "))
14     for i in range(num1, num2+1) :
15         answer = answer + i
16     print("%d+...+d는 %d입니다. " %(num1, num2, answer))
17 else :
18     print("1 또는 2만 입력해야 합니다.")
19
```

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 $1+4+\dots+100$ 의 합계를 구한다.

힌트 range(시작값, 끝값+1, 증가값) 형식으로 사용한다.

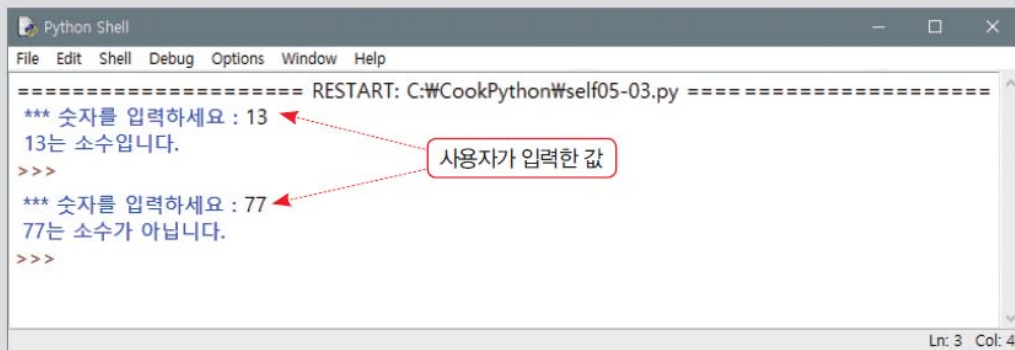


A screenshot of a Python Shell window titled 'Python Shell'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main text area shows the output of a program restart: '==== RESTART: C:/CookPython/self05-02.py ====='. The program prompts the user for three inputs: '첫 번째 숫자를 입력하세요 : 1', '두 번째 숫자를 입력하세요 : 100', and '더할 숫자를 입력하세요 : 3'. A red box labeled '사용자가 입력한 값' (Value entered by the user) has arrows pointing to these three input lines. The program then outputs '1+4+...+100는 1717입니다.' (1+4+...+100 is 1717). The prompt '>>>' is visible at the bottom left. The status bar at the bottom right shows 'Ln: 9 Col: 4'.

SELF STUDY 5-3

숫자를 하나 입력받고, 그 숫자가 소수인지를 체크하는 프로그램을 만들어 보자.

힌트 소수란 2부터 자기자신-1까지의 숫자로 나뉘어 나누어 떨어지는 숫자가 하나도 없는 수를 말한다. 예를 들어 7의 경우 2, 3, 4, 5, 6으로 각각 나누어서 나누어 떨어지는 수가 하나도 없으므로 소수다.



A screenshot of a Python Shell window titled 'Python Shell'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main text area shows the output of a program restart: '==== RESTART: C:\CookPython\self05-03.py ====='. The program prompts the user for an input: '숫자를 입력하세요 : 13'. A red box labeled '사용자가 입력한 값' (Value entered by the user) has an arrow pointing to this input line. The program then outputs '13는 소수입니다.' (13 is a prime number). The prompt '>>>' is visible at the bottom left. The status bar at the bottom right shows 'Ln: 3 Col: 4'.

Section04 if 문 응용

LAB⁴⁻⁸ 승부차기 게임

난수를 이용하여 간단한 축구 게임을 작성하여 보자. 사용자가 컴퓨터를 상대로 페널티킥을 시도한다고 생각하자. 사용자는 다음의 '왼쪽', '중앙', '오른쪽'의 3가지 영역 중에서 하나를 선택하여 페널티킥을 한다. 컴퓨터도 난수를 생성하여 3개의 영역 중에서 하나를 수비한다. 출력은 다음과 같이 나타나도록 하자.

원하는 결과

어디를 공격하시겠어요?(왼쪽, 중앙, 오른쪽) : 왼쪽
축하합니다!! 공격에 성공하였습니다.
컴퓨터의 수비위치 : 오른쪽



```
import random

n = random.randint(1, 3) # 랜덤하게 1, 2, 3 중 하나의 값을 생성
if n == 1:
    computer_choice = '왼쪽'
elif n == 2:
    computer_choice = '중앙'
else:
    computer_choice = '오른쪽'

user_choice = input('어디를 공격하시겠어요?(왼쪽, 중앙, 오른쪽) : ')
if computer_choice == user_choice:
    print('공격에 실패하였습니다.')
else :
    print('축하합니다!! 공격에 성공하였습니다.')
print('컴퓨터의 수비위치 :', computer_choice)
```