

조건으로 따져 실행하자

3가지의 기본 제어 구조

◆ 순차 구조(sequence)

- ✓ 명령들이 순차적으로 실행되는 구조

◆ 선택 구조(selection)

- ✓ 둘 중의 하나의 명령을 선택하여 실행되는 구조

◆ 반복 구조(iteration)

- ✓ 동일한 명령이 반복되면서 실행되는 구조

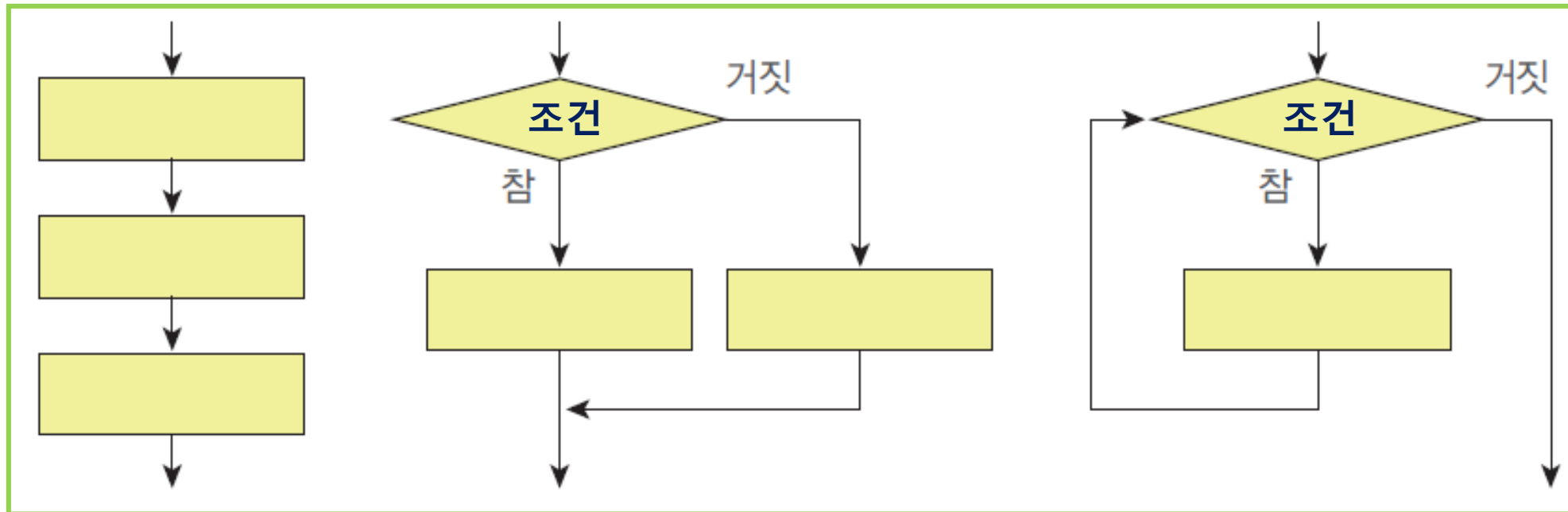
알고리즘

문제를 해결하기 위한 방법 또는 절차

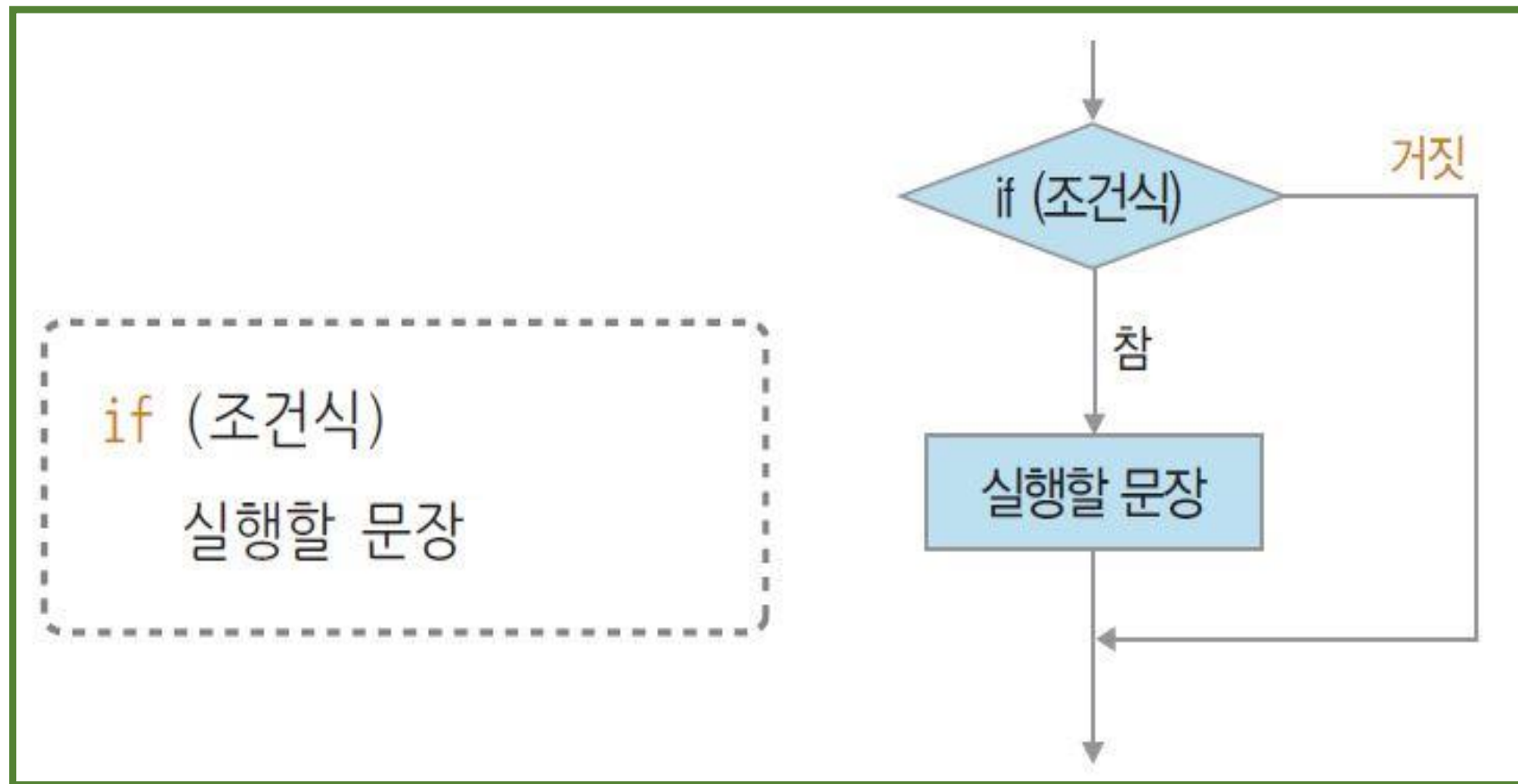
순차 구조(sequence)

선택 구조(selection)

반복 구조(iteration)



if문



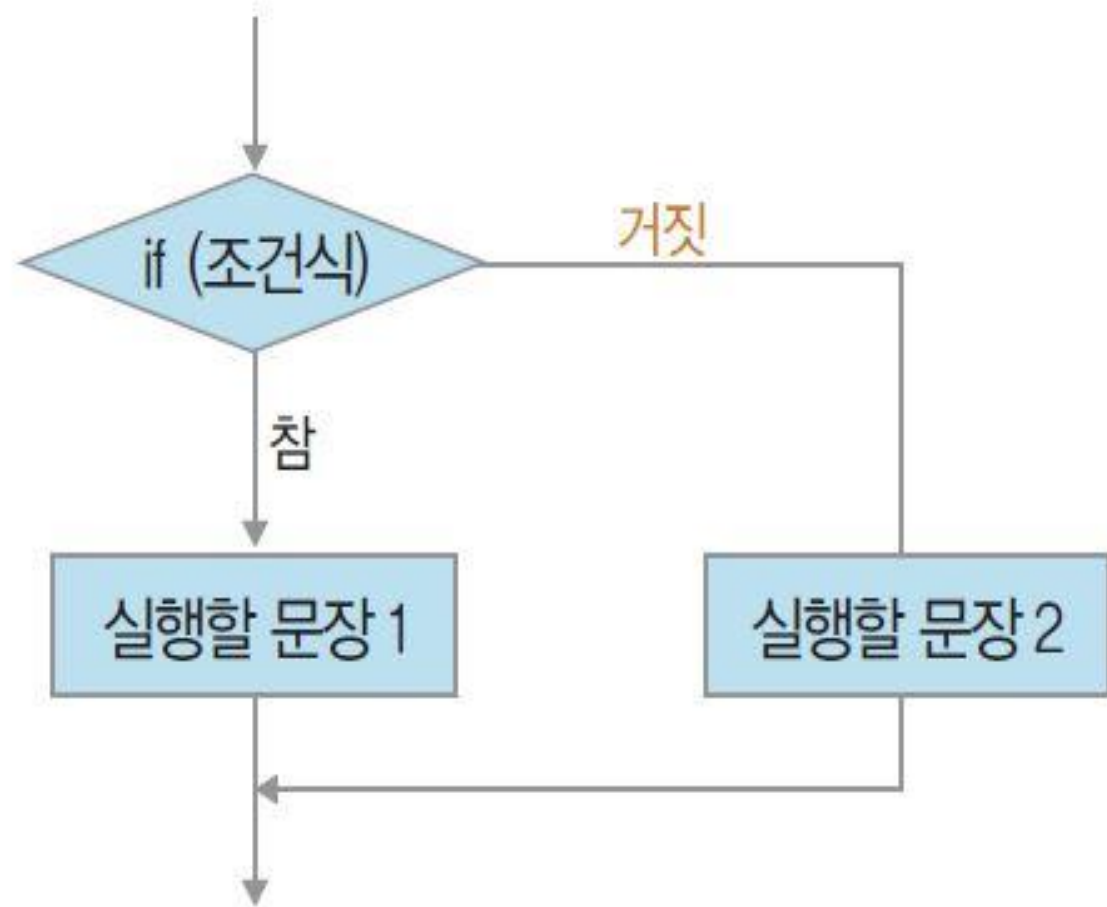
```
score = int(input("성적을 입력하시오: "))
```

```
if score >= 85:  
    print("합격입니다.")
```

```
print("next")
```

조건에 따라 실행하는 if-else 문

```
if (조건식)  
    실행할 문장 1  
else  
    실행할 문장 2
```



사용자로부터 성적을 입력받아 합격여부를 판단하는 프로그램 작성

```
score = int(input("성적을 입력하시오: "))
```

```
if score >= 90:
```

```
    print("합격입니다.")
```

```
else:
```

```
    print("불합격입니다.")
```

합격여부를 판단하는 프로그램 작성 (블록이용)

```
score = int(input("성적을 입력하시오: "))
```

```
if score >= 90:  
    print("합격입니다.")  
    print("장학금도 받을 수 있어요 ")
```

```
else:  
    print("불합격입니다.")  
    print("파이팅하세요")
```


도전

사용자로부터 정수를 입력받아 짝수·홀수 판단하는 프로그램 작성

```
num = int(input("정수를 입력: "))
```

```
if num % 2 == 0 :
```

```
    print("짝수입니다.")
```

```
else:
```

```
    print("홀수입니다.")
```

도전

사용자로부터 정수를 입력받아 짝수·홀수 판단하는 프로그램 작성

```
num = int(input("정수를 입력: "))
```

```
result = num % 2 == 0
```

```
if result == True:
```

```
    print("%d는 짝수" % num)
```

```
else:
```

```
    print("%d는 홀수" % num)
```

pass : 아무 일도 하지 않게 설정하고 싶다면?

```
data = ['paper', 'money', 'cellphone']
```

```
if 'money' in data:
```

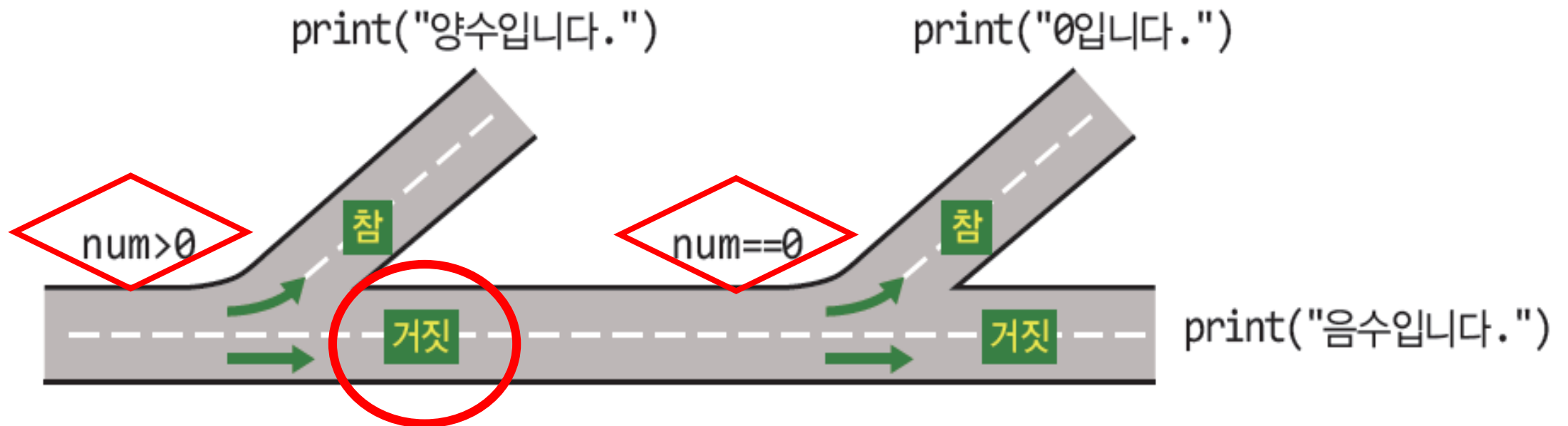
```
    pass
```

```
else:
```

```
    print("카드사용")
```

조건을 연속하여 검사

사용자에게 정수를 입력받아 양수인지, 0인지, 음수인지 판별



If ~else 문에서 조건이 거짓일 때, 다른 조건을 검사 (**elif : else if**)

```
num = int(input("정수를 입력하시오: "))
```

```
if num > 0:
```

```
    print("양수입니다.")
```

```
elif num == 0:
```

```
    print("0입니다.")
```

```
else:
```

```
    print("음수입니다.")
```

도전

1~999 사이의 정수를 입력받아,
입력 받은 숫자가
3자리의 정수인지 2자리 정수 인지 1자리 정수인지 판단
(**elif** 사용)

```
number = int(input("1~999 사이의 정수를 입력하세요 : "))
```

```
if number >= 100:
```

```
    print(" 입력한 정수는 3자리 숫자입니다")
```

```
elif number >= 10:
```

```
    print(" 입력한 정수는 2자리 숫자입니다")
```

```
else:
```

```
    print(" 입력한 정수는 1자리 숫자입니다")
```

```
print()
```

도전

키보드로 부터 jumsu를 입력 받아
Jumsu가 90점 ~100점 이면 "A등급입니다" 출력
80점 ~ 89점 이면 "B등급입니다" 출력
70점 ~ 79점 이면 "C등급입니다" 출력
69이하이면 "F등급입니다" 출력

(elif 사용)

점수	학점
90~100	A
80~89	B
70~79	C
0~69	D


```
jumsu = int(input("성적을 입력하시오: "))
```

```
if jumsu >= 90:  
    print("A 등급입니다")
```

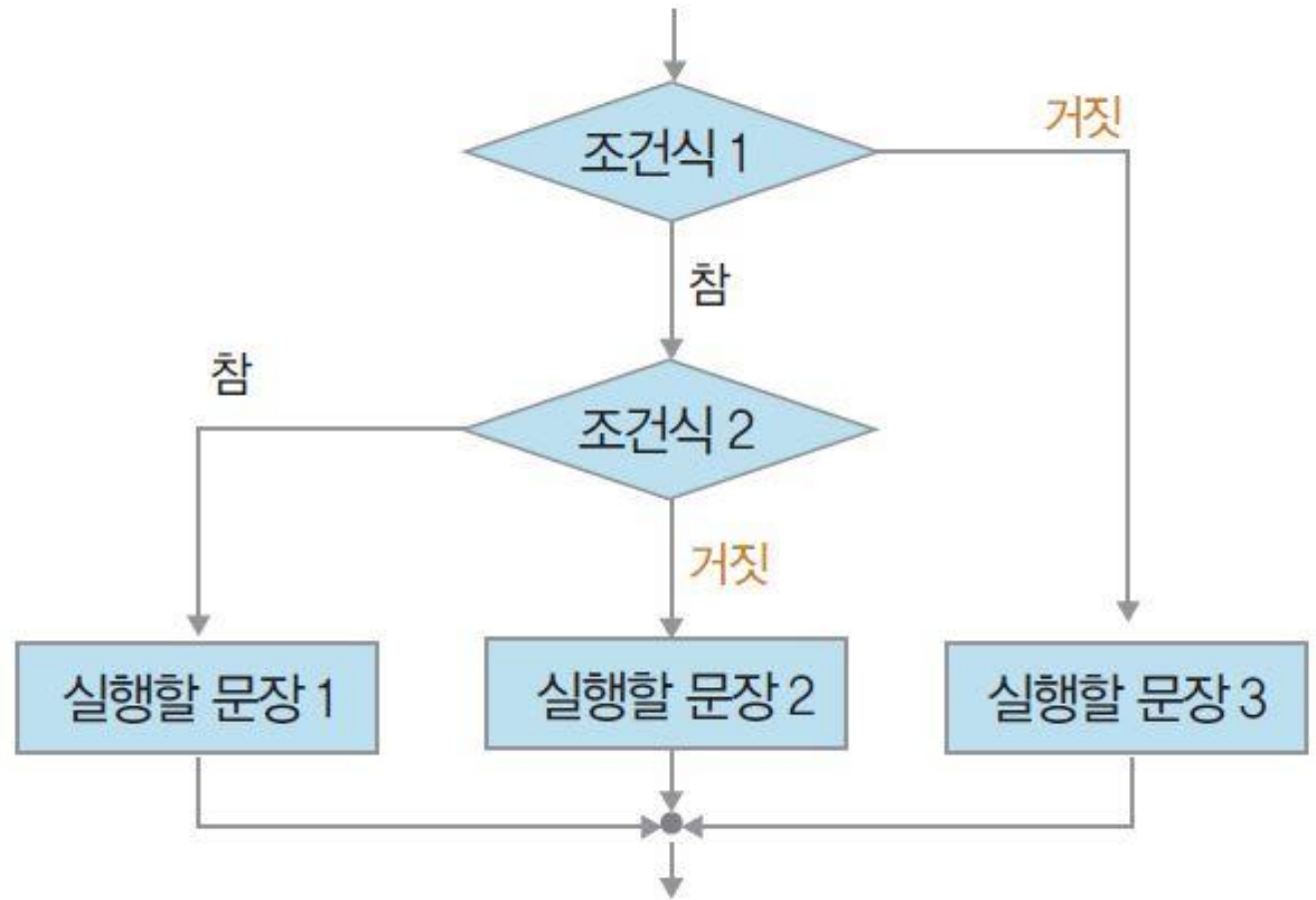
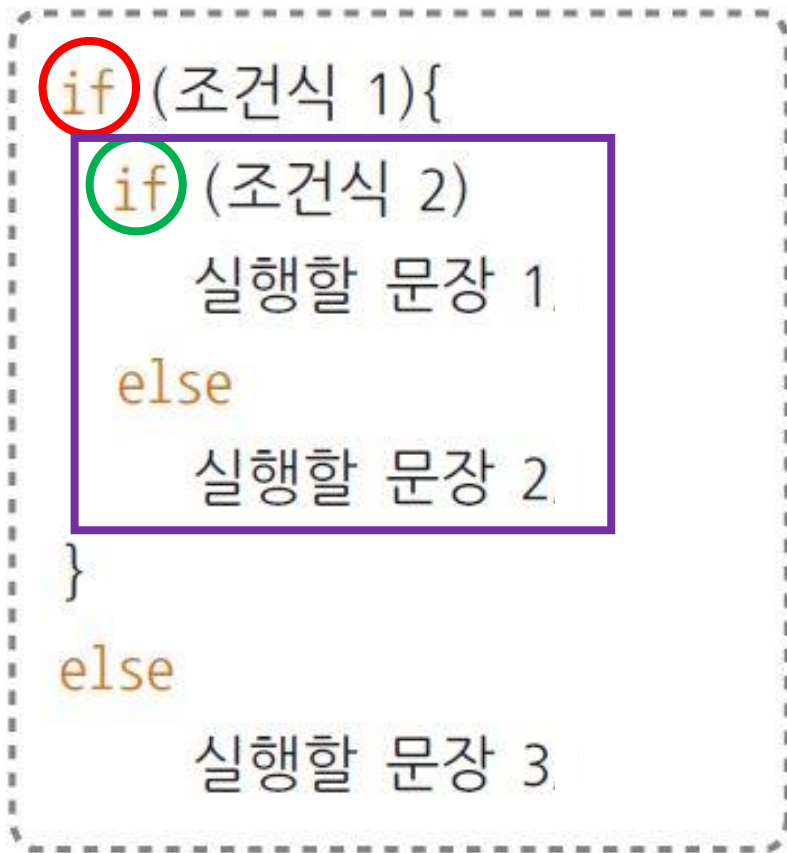
```
elif jumsu >= 80:  
    print("B 등급입니다")
```

```
elif jumsu >= 70:  
    print("C 등급입니다")
```

```
else:  
    print("F 등급입니다")
```

중첩 if ~else

if 문 안에 다른 if 문이 들어갈 수도 있다.



사용자에게 정수를 입력받아 양수인지, 0인지, 음수인지 판별

중첩 if ~ else문을 사용하여 코딩

```
num = int(input("정수를 입력하시오: "))
```

```
if num >= 0:
```

```
    if num == 0:
```

```
        print("0입니다.")
```

```
    else:
```

```
        print("양수입니다.")
```

```
else:
```

```
    print("음수입니다.")
```

리스트 []

- ◆ 여러 개의 자료들을 모아서 하나의 묶음으로 저장
- ◆ 하나의 변수에 여러 개의 값을 저장
- ◆ 리스트 내의 개별 데이터 - 항목 또는 요소

```
city_list = [ '광주', ' 서울', ' 수원', 제주' ]
```

index	city_list[0]	city_list[1]	city_list[2]	city_list[3]
	광주	서울	수원	제주

리스트 []

◆인덱스를 이용하여 리스트 항목에 접근

◆음수 인덱스

letters	letters[0]	letters[1]	letters[2]	letters[3]	letters[4]	letters[5]
	'A'	'B'	'C'	'D'	'E'	'F'
인덱스	0	1	2	3	4	5
음수인덱스	-6	-5	-4	-3	-2	-1

```
letters= ['A', 'B', 'C', 'D', 'E', 'F']  
print(letters)
```

리스트 [] ◆ `append()` , '+' 연산자

- ◆ 객체와 관련된 함수나 변수를 사용하기 위해서는 **점(.)**을 붙인 후에 함수이름이나 변수 이름을 적는다

```
fruits_list=[]  
fruits_list.append("사과")  
fruits_list.append("바나나")  
fruits_list.append("망고")
```

```
print(fruits_list)
```

fruits_list[0]
사과

fruits_list[0]	fruits_list[1]
사과	바나나

fruits_list[0]	fruits_list[1]	fruits_list[2]
사과	바나나	망고

```
fruits_list = fruits_list+['수박','자몽']  
print(fruits_list)
```

fruits_list[0]	fruits_list[1]	fruits_list[2]	fruits_list[3]	fruits_list[4]
사과	바나나	망고	수박	자몽

range() 함수 - 숫자를 생산하는 공장

◆ range(start, stop, step)

start에서 시작하여 (stop-1)까지 step 간격으로 정수들이 생성

◆ stop 값은 반드시 지정 (start, step은 생략 가능)

```
print(list(range(0,5,1)))  
print(list(range(0,5)))  
print(list(range(5)))  
print(list(range(1,11,2)))
```


random 모듈

◆ 난수와 관련한 함수를 제공

```
import random
```

```
print(random.random())  
print(random.randint(1,7))  
print()
```

```
print(random.randrange(7))  
print(random.randrange(1,7))  
print(random.randrange(0,10,2))  
print()
```

0 이상 1 미만의 임의의 실수를 반환

이 함수는 매번 다른 실수를 반환

1 이상 7 이하(7을 포함)의 임의의 정수를 반환

0 이상 7 미만(7을 포함하지 않음)의 임의의 정수를 반환

1 이상 7 미만(7을 포함안함)의 임의의 정수를 반환

0, 2, 4, 8 중(10은 포함 안함) 하나를 반환함

list와 range() 함수

```
print(list(range(0,5,1)))  
print(list(range(0,5)))  
print(list(range(5)))
```

[0, 1, 2, 3, 4]

```
li = list(range(1,11,2))  
print(li)  
print(len(li))  
print(max(li))  
print(min(li))
```

[1, 3, 5, 7, 9]
5
9
1

도전 윤년 판단

입력된 연도가 윤년인지 아닌지 판단하는 프로그램을 작성

연도를 입력하시오: 2012
2012 년은 윤년입니다.

- ✓ 연도가 4로 나누어 떨어지면서
100으로 나누어 떨어지는 않는 연도는 윤년이다.
- ✓ 400으로 나누어 떨어지는 연도는 윤년이다.



```
year = int(input("연도를 입력하시오: "))
```

```
if (year % 4 == 0 and year % 100 != 0):  
    print(year, "년은 윤년입니다.")
```

```
elif year % 400 == 0:  
    print(year, "년은 윤년입니다.")
```

```
else :  
    print(year, "년은 윤년이 아닙니다.")
```

2

SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

- ✓ 연도가 4로 나누어 떨어지면서
100으로 나누어 떨어지는 않는 연도는 윤년이다.
- ✓ 400으로 나누어 떨어지는 연도는 윤년이다.

```
year = int(input("연도를 입력하시오: "))
```

```
if ( (year % 4 == 0 and year % 100 != 0) or year % 400 == 0):
```

```
    print(year, "년은 윤년입니다.")
```

```
else :
```

```
    print(year, "년은 윤년이 아닙니다.")
```

FEBRUARY 2012						
SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

math 모듈

```
print(2**5)
```

```
print(4**(1/2))
```

```
print(abs(-5))
```

32

2.0

5

```
import math
```

```
print(math.sqrt(4))
```

```
print(math.sqrt(9))
```

```
print(math.sqrt(100))
```

```
print(math.sqrt(121))
```

2.0

3.0

10.0

11.0

```
print(math.pow(2,5))
```

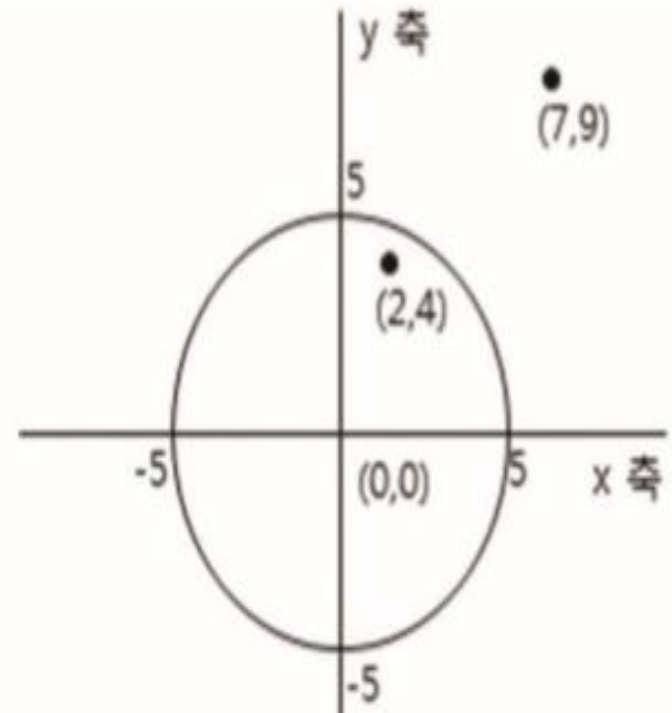
```
print(math.pow(2,10))
```

32.0

1024.0

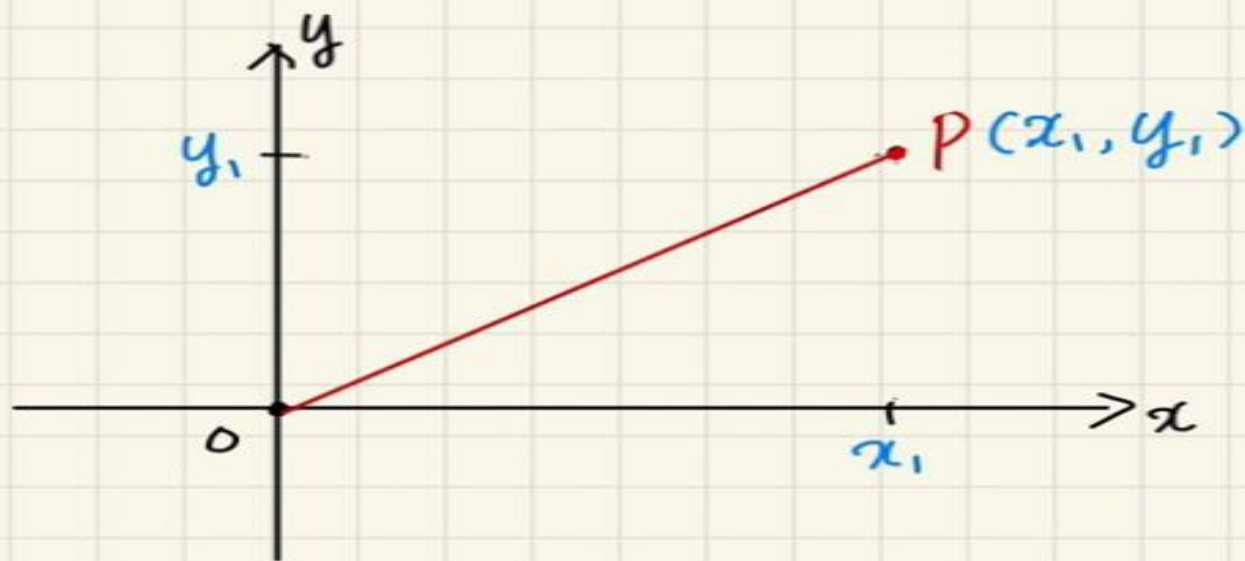
도전 원의 내부에 있는 점일까? 외부에 있는 점일까?

중심이 원점 $(0,0)$ 에 있고 반지름이 5인 원이 있다고 가정하자. 사용자로부터 x 와 y 좌표를 입력받은 후, 입력받은 점의 좌표 (x, y) 가 원의 내부에 있으면 '원의 내부에 있음', 원의 외부에 있으면 '원의 외부에 있음'을 출력하는 다음과 같은 프로그램을 작성하시오.(힌트 : 어떤 점이 원점과의 거리가 5보다 클 경우 원의 외부에 있으며, 5보다 작거나 같을 경우 원의 내부에 있다고 판단할 수 있다. 점



* 좌표평면 위의 두 점 사이의 거리

1. 원점 O 와 점 $P(x_1, y_1)$ 사이의 거리



$$\overline{OP} = \sqrt{x_1^2 + y_1^2}$$


```
import math
```

```
x = int(input('x좌표를 입력:'))
```

```
y = int(input('y좌표를 입력:'))
```

```
# 아래 부분을 완성하세요
```

```
result = math.sqrt(x*x + y*y) > math.sqrt(5*5)
```

```
print(result)
```

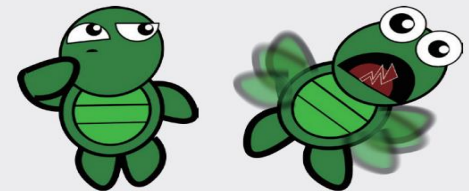
```
if result == True:
```

```
    print("원의 밖에 있음")
```

```
else:
```

```
    print("원의 안에 있음")
```

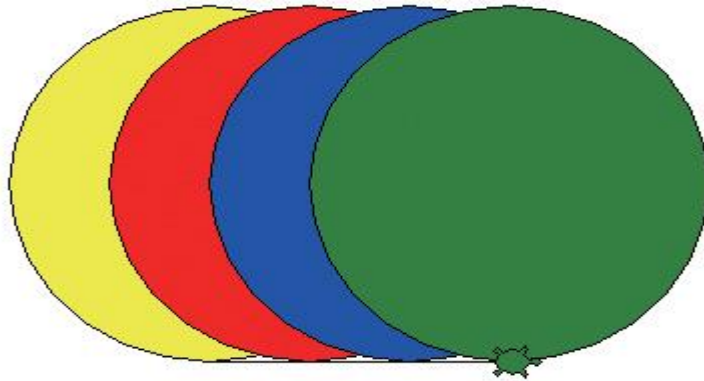
명령	하는 일
begin_fill() ... end_fill()	begin_fill()과 end_fill() 사이의 코드에 나타난 부분을 색칠한다. ... 으로 표시된 부분에는 터틀의 좌표나 모양을 기술할 수 있다.
color(c)	터틀의 색깔을 변경한다. c 값으로 'red', 'green', 'blue', 'black', 'gray', 'pink',...등의 여러가지 색상을 선택할 수 있다.
shape(s)	터틀의 모양을 변경한다. s 값으로는 'arrow', 'turtle', 'circle', 'square', 'triangle', 'classic' 등이 있다.
shapeseize(s), shapeseize(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.



<code>penup()</code> , <code>pu()</code> , <code>up()</code>	펜을 올린다(그림을 그릴 수 없는 상태로 만든다).
<code>pendown()</code> , <code>pd()</code> , <code>down()</code>	펜을 내린다(그릴 수 있는 상태로 만든다).
<code>pensize(w)</code> , <code>width(w)</code>	펜 굵기를 변경한다.
<code>circle(r)</code>	현재 위치에서 지정된 <code>r</code> 값 반지름 크기를 가지는 원을 그린다.
<code>goto(x, y)</code> , <code>setpos(x,y)</code> , <code>setposition(x,y)</code>	커서를 특정 위치(좌표)로 보낸다. 이때 <code>penup()</code> 상태이면 선이 그려지지 않으며, <code>pendown()</code> 상태이면 선이 그려진다.
<code>stamp()</code>	현재 커서의 위치에 지정된 크기와 색상, 모양의 터틀을 표시한다.
<code>home()</code>	터틀의 위치와 방향을 초기화한다.
<code>textinput()</code>	텍스트 입력을 받는 대화창을 표시하고 이 창에서 문자열을 입력 받는다.(주의: 반드시 <code>turtle.textinput()</code> 과 같이 사용)

도전 리스트에 저장된 색상으로 원그리기

리스트에 색상을 문자열로 저장
하나씩 꺼내서 거북이의 채우기 색상으로 설정하고 원을 그려 보자.



```
t.setheading(90)  
t.speed(1) #10
```

```
import turtle  
t = turtle.Turtle()  
t.shape("turtle")
```

```
color_list = [ "orange", "cyan", "yellowgreen" ]
```

```
t.fillcolor(color_list[0]) # 색상 선택
```

```
t.begin_fill()    # 채우기를 시작.  
t.circle(100)     # 속이 채워진 원이 그려진다.  
t.end_fill()      # 채우기를 종료.
```



```
turtle.done()
```

```
t.fd(50)
```

```
t.fillcolor(color_list[1])
```

```
t.begin_fill()
```

```
t.circle(100)
```

```
t.end_fill()
```

```
import random  
n = random.randint(0,2)
```

```
t.fd(50)
```

```
t.fillcolor(color_list[n])
```

```
t.begin_fill()
```

```
t.circle(100)
```

```
t.end_fill()
```


도전 부호에 따라 거북이를 움직이자

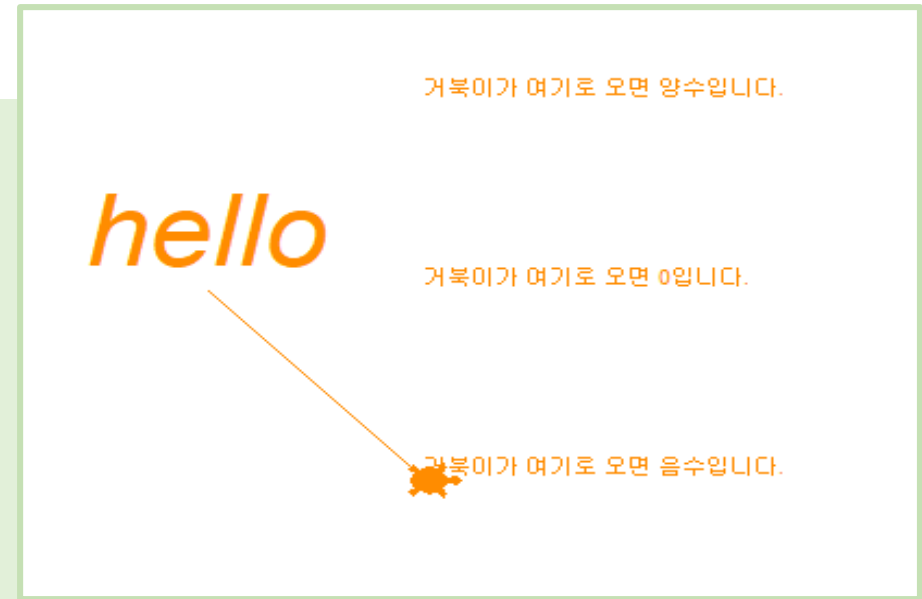
사용자로부터 정수를 받아서 입력값에 따라
거북이를(100, 100), (100, 0), (100,-100)으로 움직이는 프로그램을 작성



```
import turtle
t = turtle.Turtle()
t.shape("turtle")

t.color('dark orange')
style = ('Arial', 30, 'italic')

t.write('hello', font=style, align='center')
```



```
t.penup()  
t.goto(100, 100)  
t.write("거북이가 여기로 오면 양수입니다.")
```

```
t.goto(100, 0)  
t.write("거북이가 여기로 오면 0입니다.")
```

```
t.goto(100, -100)  
t.write("거북이가 여기로 오면 음수입니다.")
```

```
t.goto(0, 0)  
t.pendown()
```

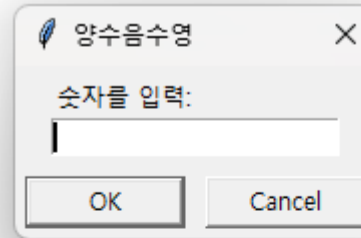
```
turtle.done()
```



```
s = turtle.textinput( " 양수음수영", "숫자를 입력: ")
```

```
n=int(s)
```

```
turtle.done()
```



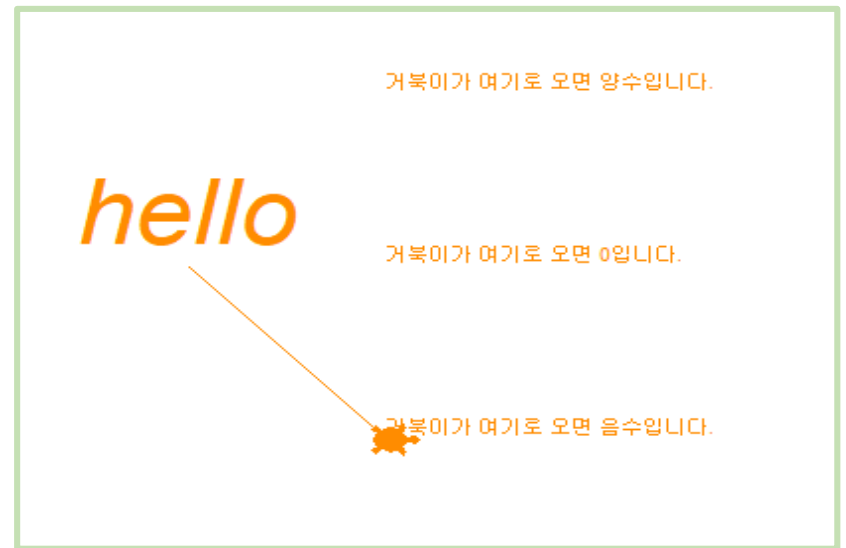
거북이가 여기로 오면 양수입니다.

hello
▶

거북이가 여기로 오면 0입니다.

거북이가 여기로 오면 음수입니다.

```
if( n > 0 ):
    t.goto(100, 100)
    t.stamp()
elif( n == 0 ):
    t.goto(100, 0)
    t.stamp()
else:
    t.goto(100, -100)
    t.stamp()
```



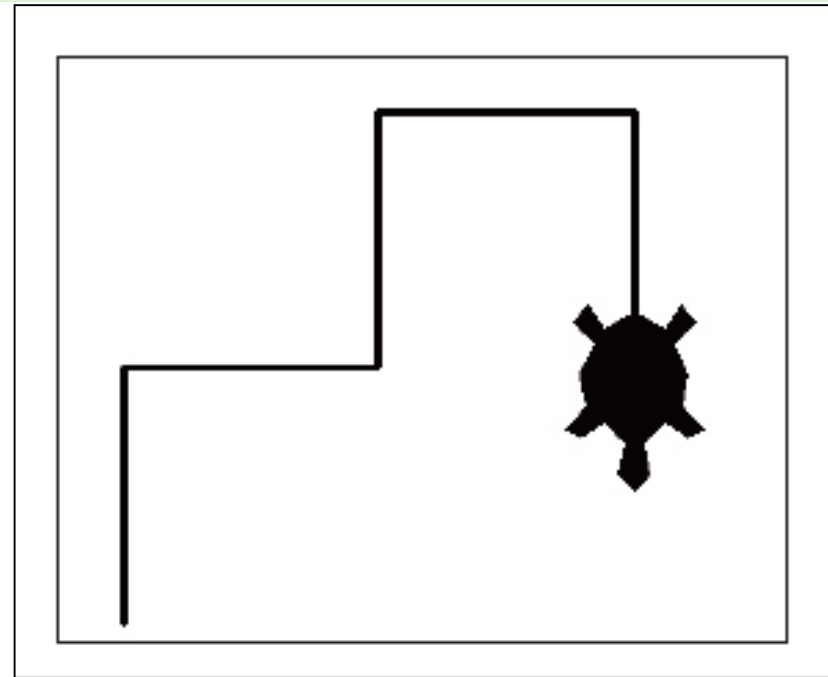
도전 부호에 따라 거북이를 움직이자

"l" (엘)을 입력하면 거북이가 왼쪽으로 100픽셀 이동.

"r"을 입력하면 거북이가 오른쪽으로 100픽셀 이동.

"s"을 입력하면 반복문 탈출

```
File Edit Shell Debug Options Window
=====
=====
명령을 입력하시오: l
명령을 입력하시오: r
명령을 입력하시오: l
명령을 입력하시오: r
명령을 입력하시오: r
```



무한 반복

다음과 같은 코드를 사용하면 무한 반복할 수 있다.

```
while True:
```

```
    ...
```

```
    ...
```

```
    ...
```

```
import turtle
t = turtle.Turtle()
t.width(3)
t.shape("turtle")
t.shapesize(2, 2)
```



```
while True:
```

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

```
    if command == "l": #엘
```

```
        t.lt(90)
```

```
        t.fd(100)
```

```
    if command == "r":
```

```
        t.rt(90)
```

```
        t.fd(100)
```

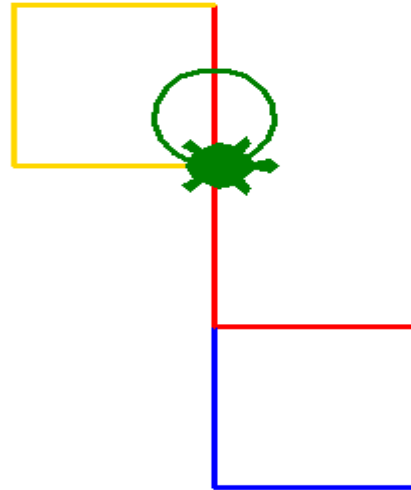
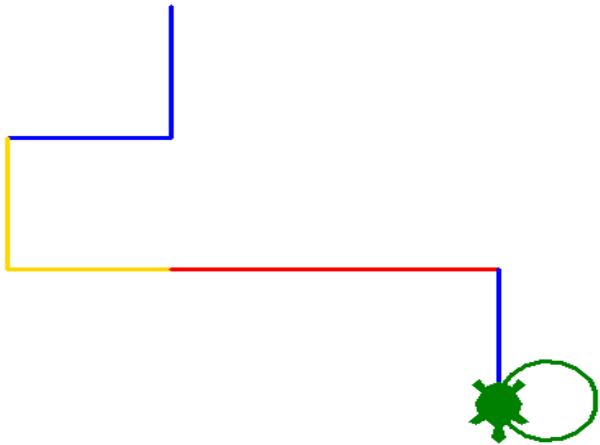
```
    if command == "s":
```

```
        break
```

```
turtle.done()
```

도전 거북이 제어

"l" (엘)을 입력하면 거북이가 왼쪽으로 100픽셀 이동
"r"을 입력하면 거북이가 오른쪽으로 100픽셀 이동
"f"을 입력하면 거북이가 앞으로 100픽셀 이동
"s"을 입력하면 , 타원을 그리고 stop



명령을 입력하시오: *f*
명령을 입력하시오: *f*
명령을 입력하시오: *r*
명령을 입력하시오: *r*
명령을 입력하시오: *r*
명령을 입력하시오: *f*
명령을 입력하시오: *f*
명령을 입력하시오: *c*
명령을 입력하시오: *r*
명령을 입력하시오: *r*
명령을 입력하시오: *r*
명령을 입력하시오: *f*
명령을 입력하시오: *f*

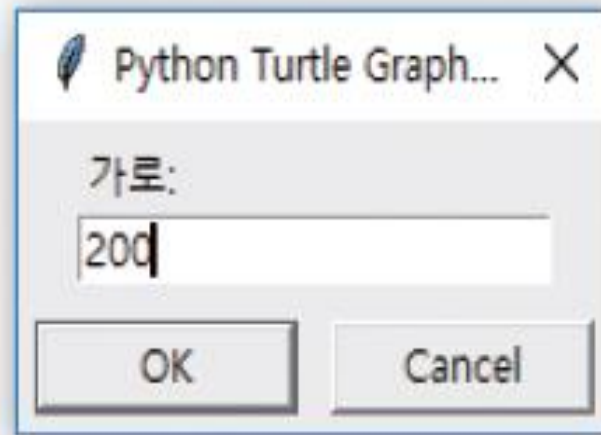
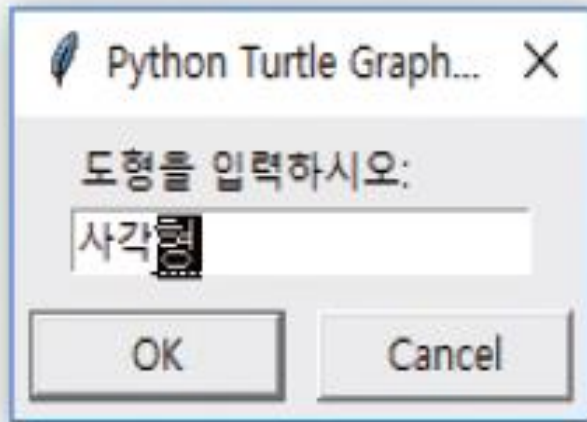
```
while True:
    command = input("명령을 입력하시오: ")
    if command == "l":
        t.color("gold")
        t.lt(90)
        t.fd(100)
    if command == 'r':
        t.color("blue")
        t.rt(90)
        t.fd(100)

    if command == 'f':
        t.color("red")
        t.fd(100)

    if command == 's':
        t.color("green")
        t.circle(30)
        break
```

도전 - 도형 그리기

터틀 그래픽을 이용하여 사용자가 선택하는 도형을 화면에 그리는 프로그램을 작성해보자. 도형은 **"사각형"**, **"삼각형"**, **"원"** 중의 하나이다. 각 도형의 치수는 사용자에게 물어보도록 하자.



```
import turtle
```

```
t = turtle.Turtle()
```

```
t.shape("turtle")
```

```
s = turtle.textinput( " 도형종류", "도형입력: ")
```

```
size = turtle.textinput( " 한변길이", "길이: ")
```

```
w = int(size)
```

```
if s == "사각형" :  
    이부분을 완성하세요
```

```
if s == "삼각형" :  
    이부분을 완성하세요
```

```
if s == "원" :  
    이부분을 완성하세요
```

```
turtle.done()
```

```
if s == "사각형" :  
    t.forward(w)  
    t.left(90)  
    t.forward(w)  
    t.left(90)  
    t.forward(w)  
    t.left(90)  
    t.forward(w)
```

```
if s == "삼각형" :  
    t.forward(w)  
    t.left(120)  
    t.forward(w)  
    t.left(120)  
    t.forward(w)  
    t.left(120)
```

```
if s == "원" :  
    t.circle(w)
```

```
turtle.done()
```