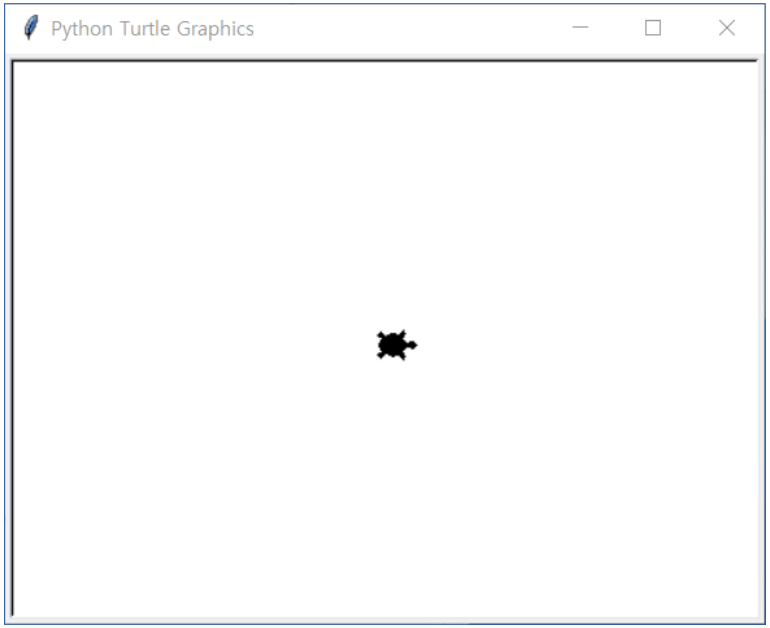
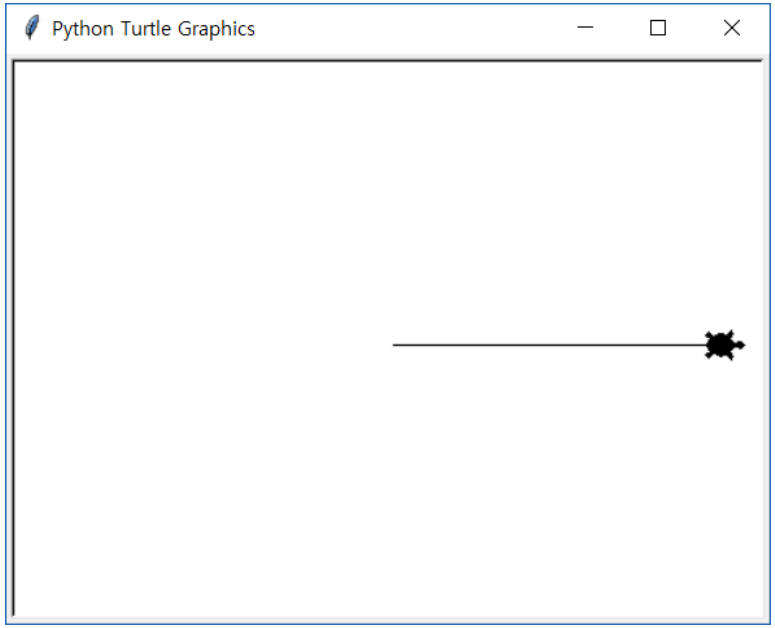
■ 터틀 그래픽

- Logo라는 언어의 그래픽 라이브러리
- 종이에 펜으로 그림을 그리는 것처럼 거북이 모양의 이미지를 이동시켜 도형을 그리는 방식
- 쉽고 직관적이어서 교육용으로 주로 활용
- 명령창에서는 실행되지 않고 그래픽으로 구성된 창이 열림





■ 터틀 그래픽

● 거북이 이미지 : shape('문자열')

문자열	모양	문자열	모양	문자열	모양
turtle		arrow		square	
classic (기본값)		circle		triangle	

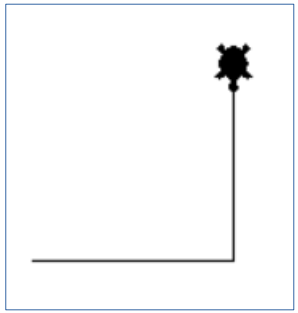
● 거북이 이동

메소드	단축명	설명	메소드	단축명	설명
forwared(픽셀)	rd	앞으로 이동	left(각도)	lt	왼쪽으로 회전
backward(픽셀)	back	뒤로 이동	right(각도)	rt	오른쪽으로 회전

■ 터틀 그래픽

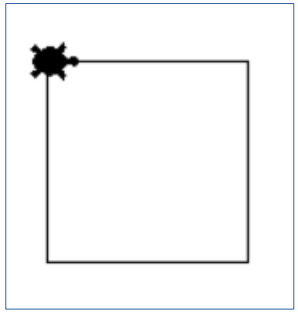
● 앞으로 100픽셀 이동 / 오른쪽으로 90도 회전 / 뒤로 100픽셀 이동

```
import turtle as t

t.shape('turtle')

t.speed(1) # 1 느리게, 10 빠르게

t.forward(100)
t.right(90)
t.backward(100)
```



● 반복문을 이용하여 네모 그리기 (앞으로 100픽셀 / 오른쪽 90도)

```
import turtle as t

t.shape('turtle')

t.speed(1) # 1 느리게, 10 빠르게

for i in range(4):
    t.forward(100)
    t.right(90)
```



■ 터틀 그래픽

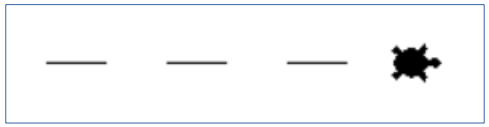
● 거북이 조정 – 1

메소드	단축명	설명	메소드	단축명	설명
pendown	down	꼬리 내림 (그림그리기)	showturtle	st	거북이 보이기
penup	up	꼬리 올림	hideturtle	ht	거북이 숨기기
stamp		도장찍기 id 이용 개별 제어 가능	speed(속도)		1 느리게 10 빠르게 0 최고속도
clear		도형 지우기	reset		도형 지우기 거북이 중앙 배치

■ 터틀 그래픽

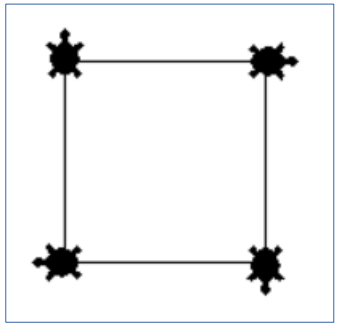
● 꼬리 올림 / 내림

```
import turtle as t

t.shape('turtle')

t.forward(30)
t.up()
t.forward(30)
t.down()
t.forward(30)
t.up()
t.forward(30)
t.down()
t.forward(30)
t.up()
t.forward(30)
```



● 도장 / 거북이 숨기기

```
import turtle as t

t.shape('turtle')

for i in range(4):
    t.forward(100)
    t.stamp()
    t.right(90)

t.hideturtle()
```



■ 터틀 그래픽

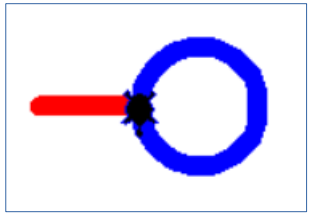
● 거북이 조정 – 2

메소드	단축명	설명	메소드	단축명	설명
pensize	width	펜 굵기	bgcolor		(캔버스) 배경 색상
circle		원 그리기 다각형 가능	pencolor		펜 색상
begin_fill		도형 내부 색칠 준비	fillcolor		채우기 색상
end_fill		도형 내부 색칠	color		펜 / 채우기 색상

■ 터틀 그래픽

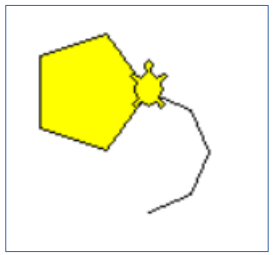
● 펜 굵기 / 색상 / 원 그리기

```
import turtle as t

t.shape('turtle')

t.pensize(10)
t.pencolor('red')
t.forward(50)

t.right(90)
t.circle(30)
```



● 180도 4조각 도형 / 360도 4조각 도형 / 색 채우기

```
import turtle as t

t.shape('turtle')

t.circle(30, 180, 4)

t.right(90)
t.begin_fill()
t.circle(30, 360, 5)
t.fillcolor('yellow')
t.end_fill()
```



■ 터틀 그래픽

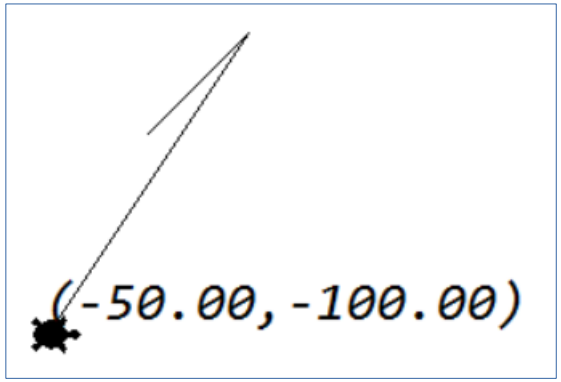
● 거북이 조정 – 3

메소드	단축명	설명	메소드	단축명	설명
xcor / ycor pos		x좌표/y좌표 x, y 좌표	distance		현 좌표와의 거리
setpos	goto	좌표 이동	towards (x, y)		(x, y) 방향의 각도
heading		현재 각도	home		초기 위치로 이동
setheading	seth	각도 변경 우 0도 상 90도	write		문자열 출력

■ 터틀 그래픽

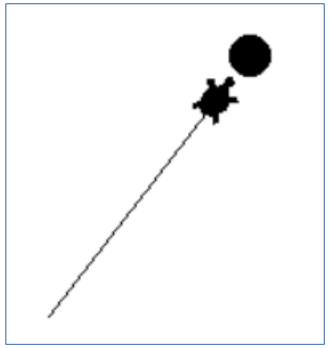
● (50, 50) 위치 이동 / (-50, -100) 위치 이동 / 현재 좌표 출력

```
import turtle as t

t.shape('turtle')

t.goto(50, 50)
t.goto(-50, -100)

t.write(str(t.pos()), font=('Consolas', 20, 'italic'))
```



● 거북이 및 원 생성 / 원 위치 이동 / 거북이 방향 전환 및 위치 이동

```
import turtle as t

t1 = t.Turtle() # 거북이
t1.shape('turtle')

t2 = t.Turtle() # 원
t2.shape('circle')
t2.up()
t2.goto(100, 130) # 원 위치 이동

# 원 방향으로 거북이 각도 변경
t1.setheading(t1.towards(t2.pos()))

# 원 좌표 - 30 좌표로 거북이 이동
t1.forward(t1.distance(t2.pos()) - 30)
```



■ 터틀 그래픽

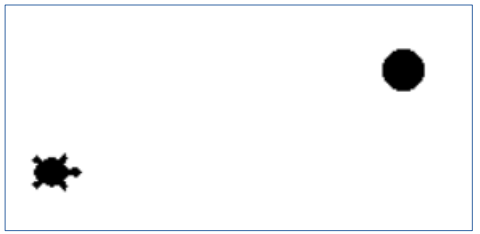
● 이벤트 처리

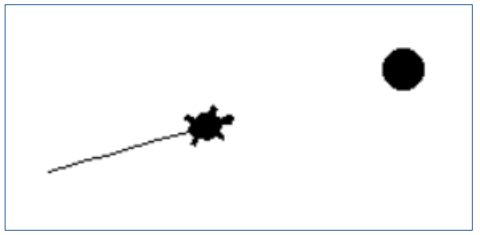
메소드	설명
onkeypress	키를 누르면 지정한 함수 호출
onkeyrealse	키를 놓으면 지정한 함수 호출
onscreenclick	마우스 버튼을 누르면 지어한 함수 호출
ontimer	일정 시간이 지난 후 함수 호출 (1 / 1000 초)
listen()	키 입력 모드 실행

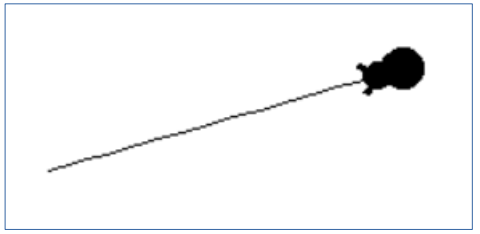
■ 터틀 그래픽

● 랜덤 좌표에 원을 생성한 후 Enter 키를 입력해서 원 방향으로 거북이 이동

```
import turtle as t
from random import randint

circle = t.Turtle('circle')
circle.speed(0)
circle.hideturtle()
circle.up()
circle.goto(randint(-300, 300), randint(-300, 300))
circle.showturtle()

my = t.Turtle('turtle')

def enter():
    circle_pos = circle.pos()
    my.setheading(my.towards(circle_pos))
    my.forward(10)


t.onkeypress(enter, 'Return')
t.listen()
```







■ 터틀 그래픽

● 따라오기 게임 (1 / 2)

```
import turtle as t
from random import randint
import threading

enemy = t.Turtle('turtle')
enemy.color('red')
enemy.up()
enemy.goto(-300, 300) # 좌상단

me = t.Turtle('arrow')
me.up()

def turn_right():
    me.setheading(0)
    me.forward(10)

def turn_up():
    me.setheading(90)
    me.forward(10)

def turn_left():
    me.setheading(180)
    me.forward(10)

def turn_down():
    me.setheading(270)
    me.forward(10)
```

■ 터틀 그래픽

● 따라오기 게임 (2 / 2)

```
def run():
    enemy.setheading(enemy.towards(me.pos()))
    enemy.forward(5)
    if enemy.distance(me.pos()) <= 7:
        me.write('Game Over')
    else:
        t.ontimer(run, 100)

t.onkey(turn_right, 'Right')
t.onkey(turn_up, 'Up')
t.onkey(turn_left, 'Left')
t.onkey(turn_down, 'Down')

t.listen()

run()
```