



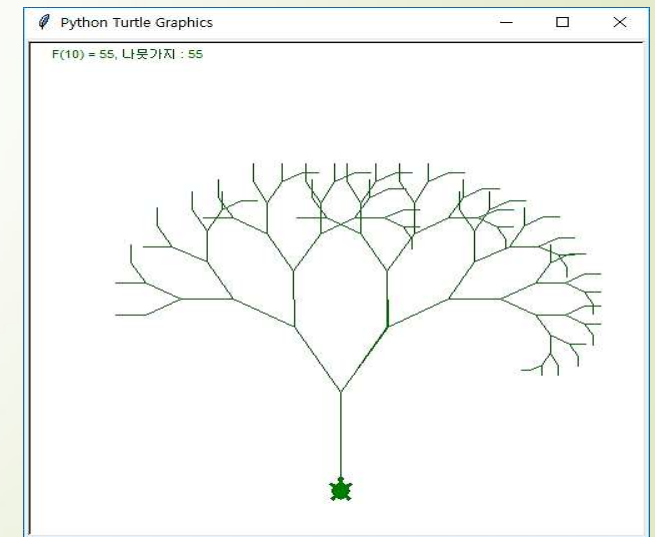
터틀 그래픽 프로젝트

컴퓨팅사고와 코딩기초

1. 터틀 그래픽이란?

▶ 터틀 그래픽 (turtle graphic)

- ▶ 1966, 교육용 프로그래밍 언어인 Logo에서 처음 소개
- ▶ 꼬리에 잉크가 묻은 거북이를 종이에 올려놓고 리모컨으로 조작하는 방식으로 동작
- ▶ 화면에서 거북이를 이용하여 지나간 흔적으로 만들어지는 그림
- ▶ 거북이가 펜을 가지고 있고 프로그래머가 명령을 이용하여 거북이를 움직이면 그림이 그려짐



2. 터틀 그래픽을 이용한 도형 그리기

■ 터틀 그래픽의 사용







- `import` 예약어로 `turtle` 모듈을 불러와 사용
- `turtle.shape("turtle")`에 의해 거북이가 캔버스에 나타남

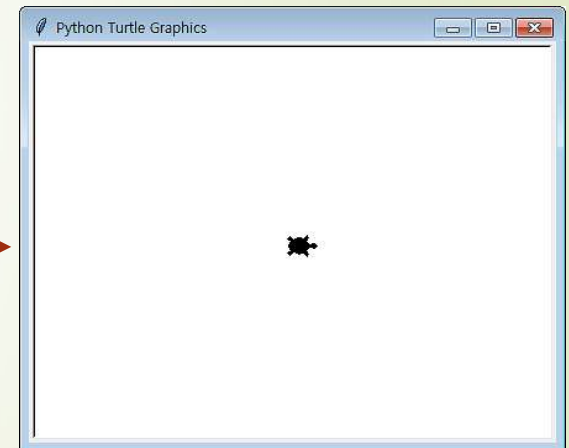
```
>>> import turtle  
>>> turtle.shape("turtle")
```

터틀 모듈을 임포트 할 때
① `import turtle as t`
- Turtle 모듈명 대신 `t`를 사용
② `from turtle import *`
- Turtle 모듈명을 생략하고 바로 메서드 사용

- [Python Turtle Graphics] 화면의 중앙(x:0, y:0)에 거북이가 나타남
- `turtle.shape("turtle")`에 의해 거북이 모양 변경가능

Turtle 모듈의 메서드 `shape`의 문자열 인수 값을 변경하여 그래픽 포인트 모양을 바꿀 수 있다.

classic	arrow	turtle	circle	square	triangle
					





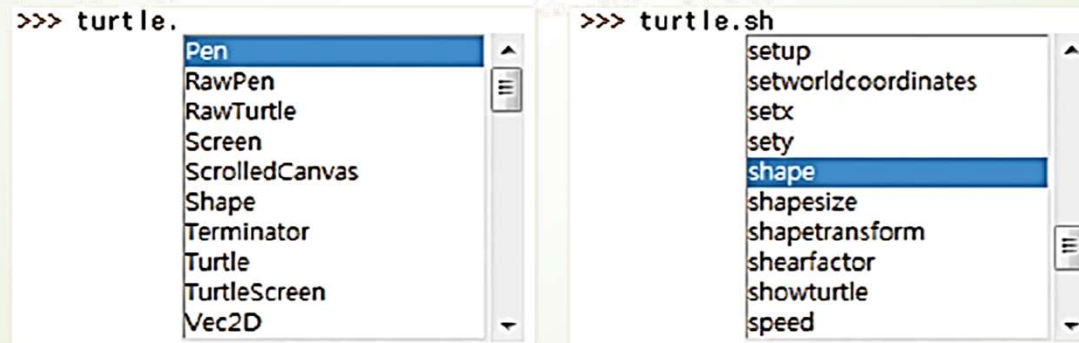
TIP

모듈의 메서드 실행 및 메서드 종류 확인

turtle 모듈 내에는 다양한 기능을 담당하는 메서드(method)가 제공되며, 다음 형태로 해당 메서드를 실행할 수 있다.

```
turtle.shape("turtle")    # 모듈.메서드명(인수)
```

문장을 입력하는 도중 모듈 내의 메서드를 확인하기 위하여 모듈명 및 .(점)을 입력하고 Tab키를 누르면 모듈 내의 메서드가 나타난다. 방향키로 위/아래로 이동하거나 메서드를 입력하면서 원하는 메서드를 찾아 스페이스키를 누르면 메서드를 입력할 수 있다.



※ 터틀 모듈 공식문서 참조:
<https://docs.python.org/3/library/turtle.html>

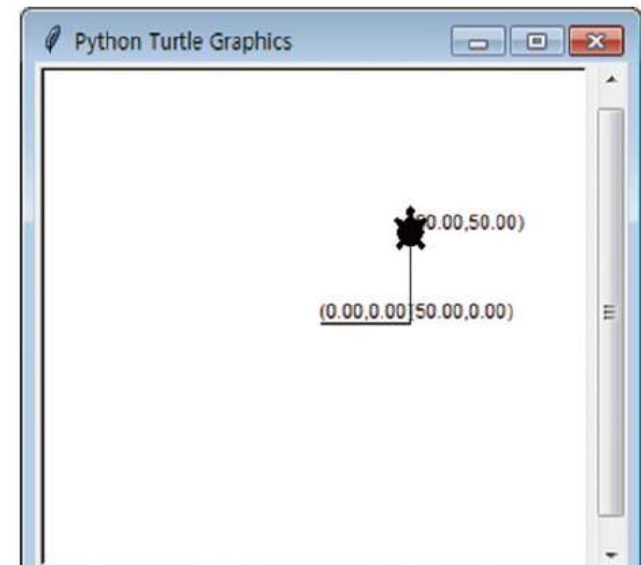
* 모듈 내의 메서드는 일반적으로 함수 형태이므로 '모듈의 메서드'를 '모듈의 메서드 함수' 또는 '모듈의 함수'로 표현하기도 한다.

2.1 거북이를 이동하여 선 그리기

거북이를 앞으로 50이동하고, 왼쪽으로 90도 회전한 후, 앞으로 50이동하면서 선을 그려본다. 이동하는 과정에서 거북이의 현재 위치를 출력해보자.

```
>>> import turtle
① >>> turtle.shape("turtle")
② >>> turtle.write(turtle.position())
③ >>> turtle.forward(50)
>>> turtle.write(turtle.position())
④ >>> turtle.left(90)
⑤ >>> turtle.forward(50)
>>> turtle.write(turtle.position())
```

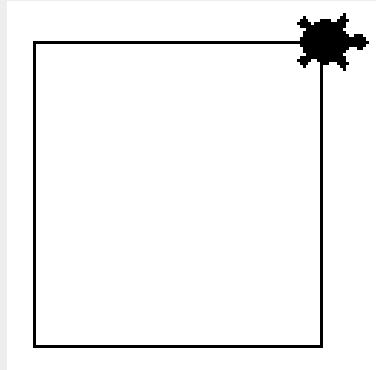
거북이 방향과 회전
메서드 종류:
앞(forward),
뒤(backward),
왼쪽(left),
오른쪽(right)



2.2 터틀 그래픽을 이용한 도형 그리기(1)

거북이를 이용하여 한 변의 크기가 100픽셀인 정사각형을 그려보자.

```
>>> import turtle
>>> turtle.shape("turtle")
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
```



[참고]

turtle.clear()

- 화면을 지우고 그 자리에 그대로 있음.

turtle.reset()

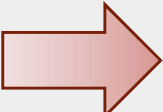
- 화면을 지우고 초기 위치로 돌아옴.

- 펜 색깔, 굵기 등 기본 설정까지 모두 초기화 된다.

2.2 터틀 그래픽을 이용한 도형 그리기(2)

100픽셀인 정사각형 그리기의 반복코드를 반복문으로 표현해보자.

```
>>> import turtle
>>> turtle.shape("turtle")
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
>>> turtle.right(90)
>>> turtle.forward(100)
```

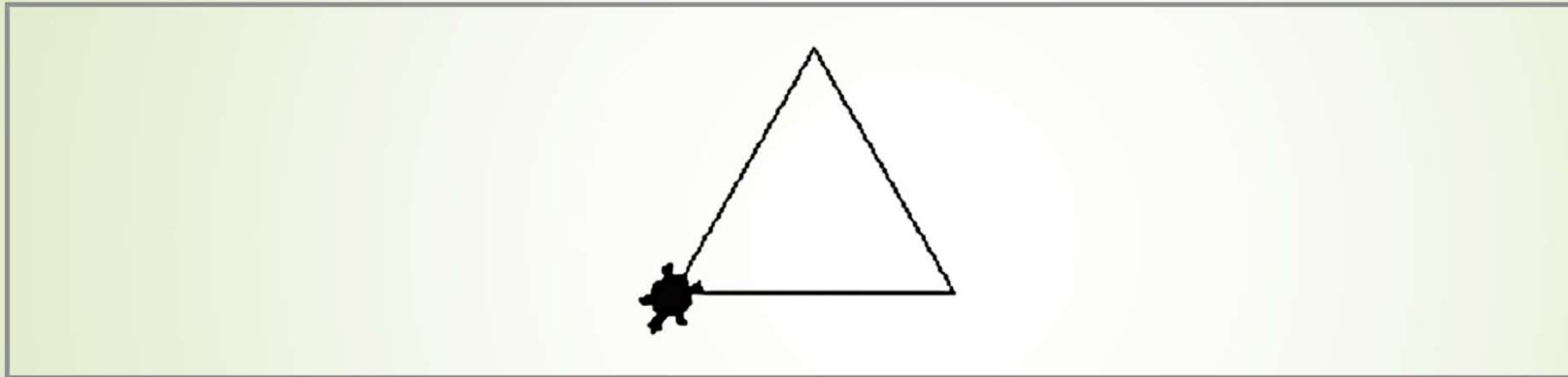


```
>>> for i in range(4):
>>>     t.right(90)
>>>     t.forward(100)
```

```
>>>
```

2.3 터틀 그래픽을 이용한 다양한 도형(1)

터틀 그래픽을 이용하여 삼각형을 그려보자.

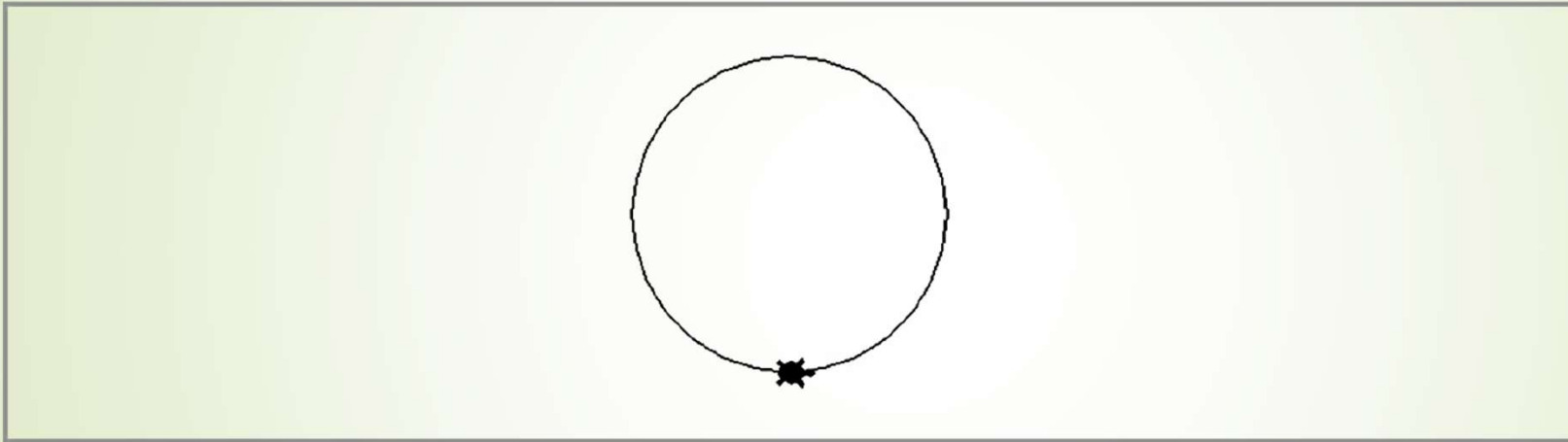


turtle.forward(100)과 turtle.left(120)을 반복하여 삼각형을 그릴 수 있다.

```
>>> t.reset()
>>> for i in range(3):
    t.forward(100)
    t.left(120)
```


2.3 터틀 그래픽을 이용한 다양한 도형(2)

앞서 작업한 터틀 그래픽을 초기화하고, 반지름이 100인 원을 그려보자.

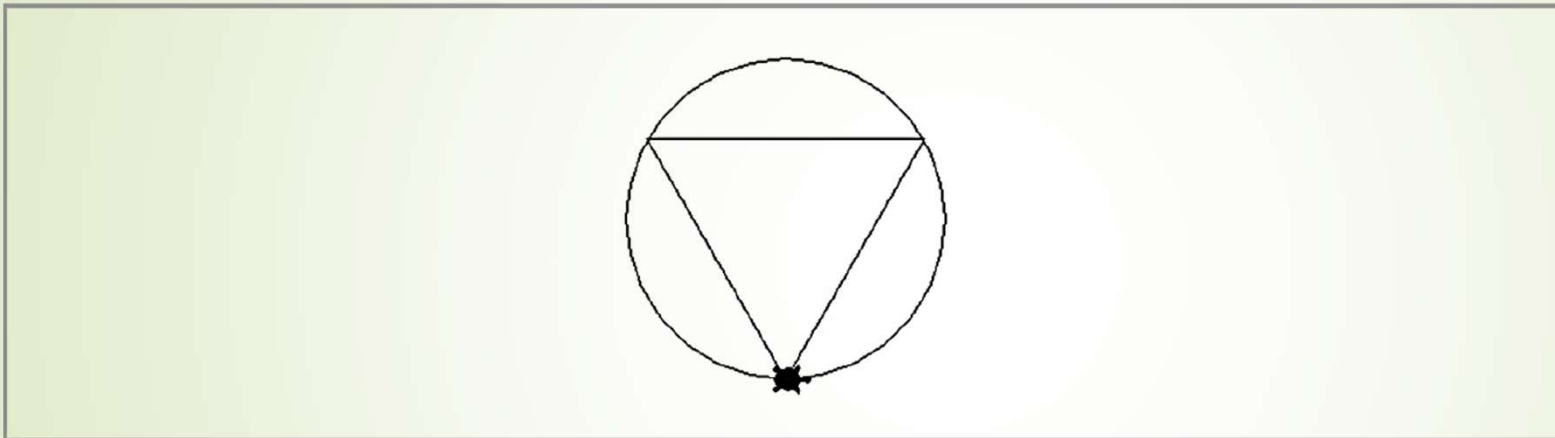



터틀 그래픽의 초기화는 `turtle.reset()` 함수, 원을 그리기 위해서는 `turtle.circle()` 함수를 사용한다.

```
>>> t.reset()
>>> t.circle(100)
```

2.3 터틀 그래픽을 이용한 다양한 도형(3)

앞서 작업한 반지름이 100인 원에 내접하는 역삼각형을 그려보자.



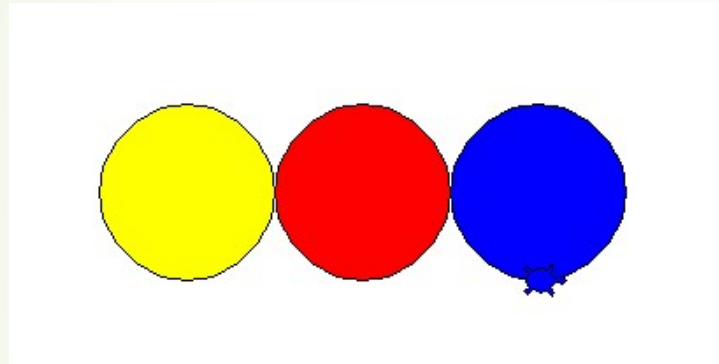
 turtle.circle() 함수의 인수로 반지름 값 100과, steps=3을 지정하면 반지름 100에 내접하는 역삼각형을 그릴 수 있다.

```
>>> t.reset()
>>> t.circle(100)
>>> t.circle(100, steps=3)
```

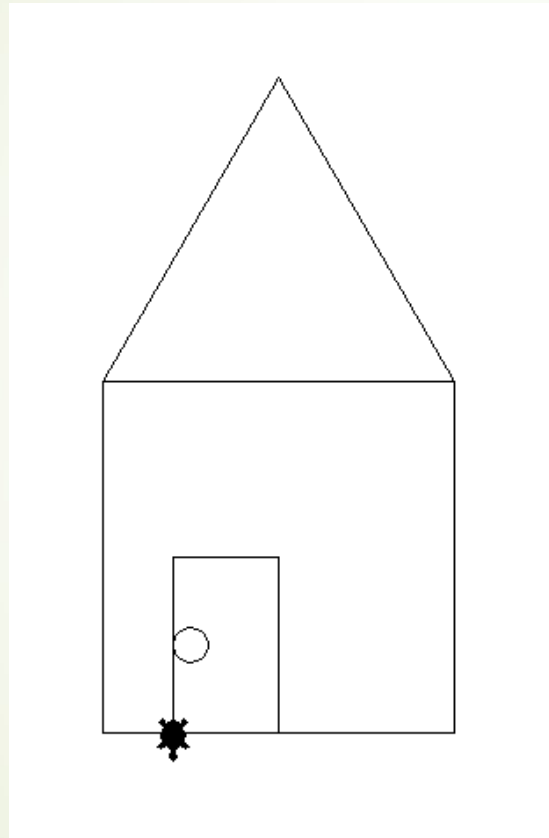
주요 명령어

➤ <code>import turtle</code>	# 터틀 그래픽을 사용하기 위해 터틀 모듈을 불러온다.
➤ <code>turtle.shape("turtle")</code>	# 거북이 모양을 결정한다. 기본 모양은 classic 이며 turtle, circle, square, arrow, triangle 모양이 있다.
➤ <code>turtle.bgcolor("red")</code>	# bgcolor를 이용하여 배경색을 red로 결정한다.
➤ <code>turtle.color("yellow")</code>	# color를 이용하여 선 색을 yellow로 결정한다.
➤ <code>turtle.forward(100)</code>	# forward를 이용하여 거북이를 앞으로 100픽셀 만큼 이동한다.
➤ <code>turtle.right(90)</code>	# right를 이용하여 거북이를 오른쪽으로 90도 회전한다. 왼쪽으로 회전할 경우 left로 수정한다.
➤ <code>turtle.pensize(10)</code>	# pensize를 이용하여 선의 굵기를 10으로 변경한다.
➤ <code>turtle.circle(100)</code>	# circle을 이용하여 반지름이 100인 원을 그린다.
➤ <code>turtle.up()</code>	#up을 이용하여 펜을 올려 거북이가 이동할 때 그림이 그려지지 않게 설정한다.
➤ <code>turtle.goto(-200, 0)</code>	# goto 를 이용하여 좌표로 이동한다. 화면 중앙 좌표는 (0,0)이다.
➤ <code>turtle.down()</code>	# down을 이용하여 펜을 내려 거북이가 이동할 때 그림이 그려지게 설정한다.
➤ <code>turtle.fillcolor("blue")</code>	# fillcolor를 이용하여 도형 내부의 색을 결정한다.
➤ <code>turtle.begin_fill()</code>	# begin_fill을 이용하여 도형 내부 채우기를 시작한다. 이때 거북이 내부의 색이 도형 내부의 색으로 변한다.
➤ <code>turtle.circle(100)</code>	# circle을 이용하여 반지름이 100인 원을 그린다.
➤ <code>turtle.end_fill()</code>	# end_fill을 이용하여 도형 내부 채우기를 종료한다.
➤ <code>turtle.clear()</code>	# clear를 이용하여 화면의 그래픽을 지운다.

Ex. 삼색 원 그리기

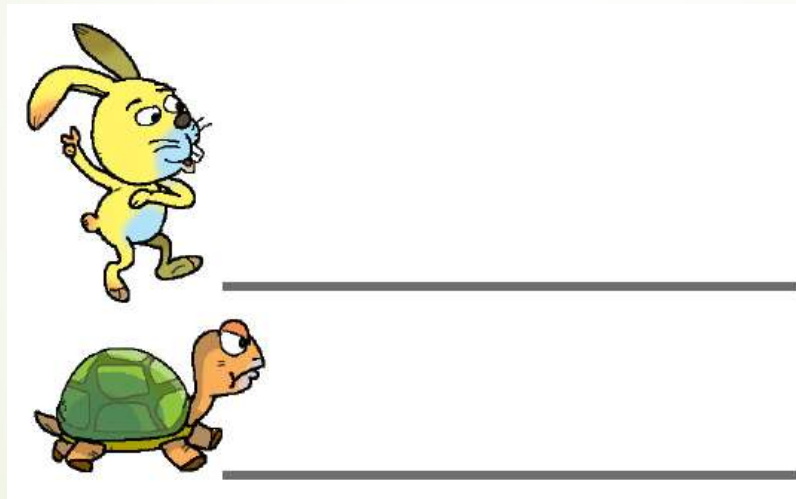


Ex. 집 그리기



3. 팀프로젝트에서 만들 프로그램

- ▶ 거북이 경주 게임 작성
- ▶ 난수 이용



3.1 난수 발생하기

- ▶ 난수(random number)는 게임과 시뮬레이션에 필수적
- ▶ random 모듈을 통하여 난수 발생을 지원

```
>>> import random
```

```
>>> random.random() # 0.0부터 1.0보다 작은 실수 난수  
0.8345121533431609
```

```
>>> random.randint(1, 100) # 1부터 100 사이의 정수 난수  
49
```

```
>>> random.choice('abcdefghij') # 리스트에서 하나의 랜덤한 항목  
'c'
```

3.2 터틀 그래픽 함수들

함수	인수	설명
forward()	픽셀값	거북이를 지정된 거리만큼 앞으로 이동한다.
backward()	픽셀값	거북이를 지정된 거리만큼 뒤로 이동한다.
right()	각도	거북이를 시계방향으로 회전시킨다.
left()	각도	거북이를 반시계방향으로 회전시킨다.
penup()	None	거북이의 펜을 올린다. 그림이 그려지지 않는다.
pendown()	None	거북이의 펜을 내린다. 그림이 그려진다.
up()	None	거북이의 펜을 올린다. 그림이 그려지지 않는다.
down()	None	거북이의 펜을 내린다. 그림이 그려진다.
color()	색상 이름	거북이 펜의 색상을 변경한다.
fillcolor()	색상 이름	다각형을 채우는 색상을 변경한다.
heading()	None	현재의 방향을 반환한다.
position()	None	현재 위치를 반환한다.
goto()	x, y	거북이를 (x, y) 위치로 이동시킨다.
begin_fill()	None	채워진 다각형을 시작한다.
end_fill()	None	채워진 다각형을 닫는다.
dot()	None	현재 위치에 점을 남긴다.
stamp()	None	현재 위치에 거북이 모양을 남긴다.
shape()	모양 이름	거북이의 모양을 'arrow', 'classic', 'turtle', 'circle' 중의 하나로 변경한다.

거북이 경주 게임 예 #1

➡ 거북이 2마리 생성

```
import turtle
```

```
t1 = turtle.Turtle() # 첫 번째 거북이
```

```
t2 = turtle.Turtle() # 두 번째 거북이
```

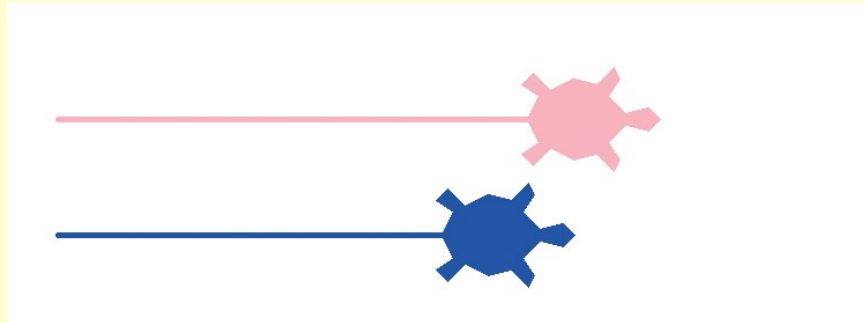


거북이 경주 게임 예 #2

➡ 거북이들 구별을 위해 색상, 모양 다르게 설정

```
t1.color("pink")  
t1.shape("turtle")  
t1.shapesize(5)  
t1.pensize(5)
```

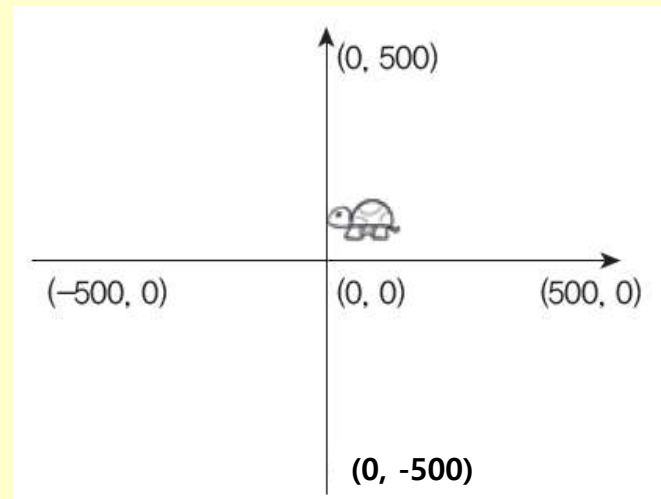
```
t2.color("blue")  
t2.shape("turtle")  
t2.shapesize(5)  
t2.pensize(5)
```



거북이 경주 게임 예 #3

➡ 출발점에 세우기

```
t1.penup()  
t1.goto(-400, 0)  
  
t2.penup()  
t2.goto(-400, -100)
```



거북이 경주 게임 #4

➡ 30번 정도 반복하면서 한 번 반복할 때마다 난수만큼 이동

```
for i in range(30):  
    d1 = random.randint(1, 50)  
    t1.forward(d1)  
    d2 = random.randint(1, 50)  
    t2.forward(d2)
```

10번 반복한다.
1부터 50 사이의 난수를 발생한다.
난수만큼 이동한다.
1부터 50 사이의 난수를 발생한다.
난수만큼 이동한다.

