

turtle

학습목표

- turtle 모듈을 이용하여 정보를 그래픽으로 표현한다.

import ~ as : 긴 모듈 이름을 간략하게 부르는 별명을 붙여주는 방법

NOTE : import ~ as 문법

모듈 내에 있는 클래스나 메소드를 활용할 때 점으로 연결해주는데, 모듈 이름이 너무 긴 경우 계속해서 이름을 써주는 것이 번거로울 수 있다. 이 때, as를 활용하여 모듈의 새 이름을 지정할 수 있다. 보통 math 모듈은 m, datetime은 dt, random은 rd, turtle은 t와 같은 짧은 이름을 널리 사용한다.

ex1) import datetime as dt

ex2) import random as rd

ex3) import math as m

ex3) import turtle as t

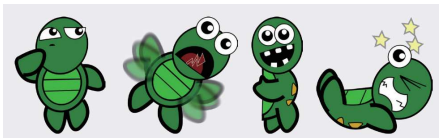
as t를 붙이지 않았을 때	as t를 붙였을 때
import turtle	import turtle as t
turtle.forward(100) turtle.right(100) turtle.forward(100)	t.forward(100) t.right(100) t.forward(100)



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.
shapsize(s), shapsize(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)

함수	설명	사용 예
forward(거리) / fd(거리)	거북이가 앞으로 이동합니다.	t.forward(100)
backward(거리) / back(거리)	거북이가 뒤로 이동합니다.	t.back(50)
left(각도) / lt(각도)	거북이가 왼쪽으로 회전합니다.	t.left(45)
right(각도) / rt(각도)	거북이가 오른쪽으로 회전합니다.	t.right(45)
circle(반지름)	현재 위치에서 원을 그립니다.	t.circle(50)
down() / pendown()	펜(잉크 묻힌 꼬리)을 내립니다.	t.down()
up() / penup()	펜(잉크 묻힌 꼬리)을 올립니다.	t.up()
shape("모양")	거북이 모양을 바꿉니다.	t.shape("turtle") t.shape("arrow") ※ 거북이 모양으로 "circle", "square", "triangle"을 사용
speed(속도)	거북이 속도를 바꿉니다.	t.speed(1) # 가장 느린 속도 t.speed(10) # 빠른 속도 t.speed(0) # 최고 속도

pensize(굵기) / width	펜 굵기를 바꿉니다.	t.pensize(3)
color("색 이름")	펜 색을 바꿉니다.	t.color("red")
bgcolor("색 이름")	화면의 배경색을 바꿉니다.	t.bgcolor("black")
fillcolor("색 이름")	도형 내부를 칠하는 색을 바꿉니다.	t.fillcolor("green") ※ 색상을 따로 지정하지 않으면 현재 색으로 칠합니다.
begin_fill()	도형 내부를 색칠할 준비를 합니다.	t.begin_fill()
end_fill()	도형 내부를 색칠합니다.	t.end_fill()
showturtle() / st()	거북이를 화면에 표시합니다.	t.st()
hideturtle() / ht()	거북이를 화면에서 가립니다.	t.ht()
clear()	거북이를 그대로 둔 채 화면을 지웁니다.	t.clear()
reset()	화면을 지우고 거북이도 원래 자리와 상태로 되돌립니다.	t.reset()

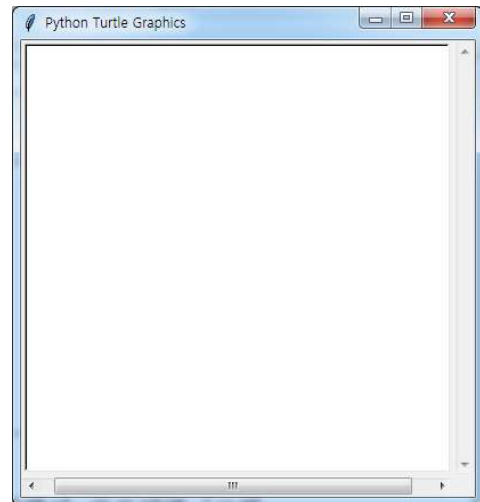
함수	설명	사용 예
pos() / position()	거북이의 현재 위치(좌표)를 구합니다 (x, y 둘 다).	t.pos()
xcor(), ycor()	거북이의 x 좌표나 y 좌표를 구합니다 (x, y 중 하나만).	a = t.ycor()
goto(x, y), setpos(x, y)	거북이를 특정 위치(좌표)로 보냅니다 (x, y 둘 다).	t.goto(100,50)
setx(x), sety(y)	거북이의 x 좌표나 y 좌표를 지정한 위치로 이동합니다 (x, y 중 하나만).	t.sety(50)
distance(x, y)	현재 거북이가 있는 위치에서 특정 위치까지의 거리를 구합니다.	d = t.distance(100,100)
heading()	거북이가 현재 바라보는 각도를 구합니다.	ang = t.heading()
towards(x, y)	현재 거북이가 있는 위치에서 특정 위치까지 바라보는 각도를 구합니다.	ang = t.towards(10,10)
setheading(각도)/ seth(각도)	거북이가 바라보는 방향을 바꿉니다.	t.setheading(90) # 거북이가 화면 위쪽을 바라봅니다. .
home()	거북이의 위치와 방향을 처음 상태로 돌립니다.	t.home()

함수	설명	사용 예
onkeypress(함수, "키 이름")	키보드를 눌렀을 때 실행할 함수를 정합니다.	def f(): t.forward(10) t.onkeypress(f, "Up") # 위쪽 방향키 ↑ 를 누르면 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다).
onscreenclick(함수)	마우스 버튼을 눌렀을 때 실행할 함수를 정합니다.	t.onscreenclick(t.goto) # 마우스 버튼을 누르면 앞에서 정의한 goto 함수를 호출합니다(goto 함수는 거북이를 마우스 버튼을 누른 위치로 이동시킵니다).
ontimer(함수, 시간)	일정한 시간이 지난 뒤 실행할 함수를 정합니다.	def f(): t.forward(10) t.ontimer(f, 1000) # 1000밀리초(1초) 후에 f 함수를 호출합니다(f 함수는 거북이를 10만큼 앞으로 이동시킵니다.)
listen()	사용자 입력이 잘 처리되도록 거북이 그래픽 창에 포커스를 줍니다	t.listen()
title("창 이름")	거북이 그래픽 창의 이름을 지정합니다.	t.title("welcome") # 거북이 그래픽 창의 이름이 Untitle에서 welcome으로 바뀝니다.
write("문자열")	현재 거북이 위치에 문자를 출력합니다.	t.write("Hello") # 현재 거북이 위치에 Hello를 출력합니다. t.write("Hello", False, "center", ("", 20)) # 현재 거북이 위치에 가운데 정렬로 크기가 20인 Hello를 출력합니다.

- 파이썬 대화창에서 다음과 같은 코드를 입력하면 오른쪽의 창이 뜬다.

대화창 실행 : 터틀 모듈의 활용

```
>>> import turtle as t
>>> t.setup(width = 400, height = 400)
```

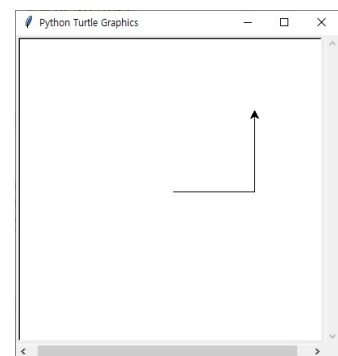


done() 사용

코드 7-13 : 터틀 그래픽의 setup(), forward(), left(), bye() 함수

turtle_show3.py

```
import turtle as t
t.setup(width = 400, height = 400)
t.forward(100)
t.left(90)
t.forward(100)
t.done()    # 마우스 이벤트를 받기 위해 이벤트 루프로 진입하는 기능
```









사용자 입력을 기다리게 됨

shape() 함수

- 터틀 그래픽의 디폴트 커서 모양은 화살표 모양
- shape() 함수를 이용하여 커서의 모양을 바꿀 수 있음

[표 7-4] 터틀 그래픽의 커서 이름과 모양

arrow	turtle	circle	square	triangle	classic
					

shapsize() 함수

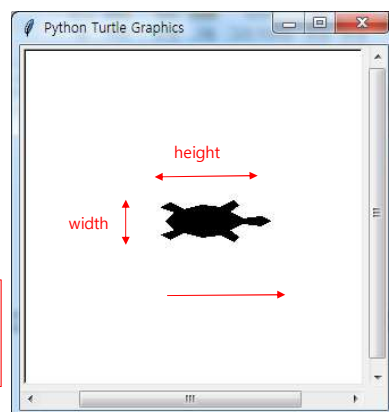
- shapsize() 함수를 이용하여 커서의 크기를 바꿀 수 있음

대화창 실습 : 터틀 커서의 크기 변경

```
>>> t.shapesize(2, 4)
```

- 너비 2, 높이 4로 변경

거북이의 진행방향을 기준으로
너비가 2
높이(키)가 4임



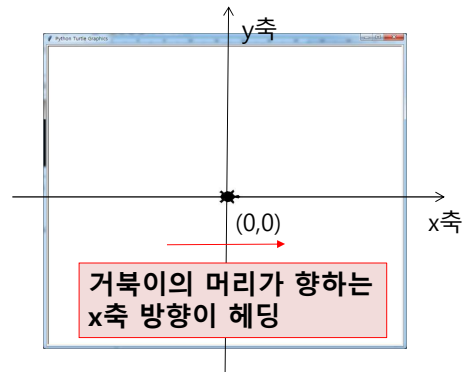
움직임과 좌표 변경, 그리기 메소드

메소드	하는 일
<code>forward(d) fd(d)</code>	터틀을 거리 d만큼 앞으로 이동한다.
<code>backward(d) bk(d) back(d)</code>	터틀을 거리 d만큼 뒤로 이동한다.
<code>left(x) lt(x)</code>	터틀을 각도만큼 왼쪽으로 회전한다.
<code>right(x) rt(x)</code>	터틀을 각도만큼 오른쪽으로 회전한다.
<code>circle(r)</code>	현재 위치에서 지정된 반지름 크기의 원을 그린다.
<code>goto(x, y) setpos(x,y) setposition(x,y)</code>	커서를 특정 위치(좌표)로 보낸다.
<code>setx(x)</code>	커서의 x 좌표를 지정한 위치로 이동한다.
<code>sety(y)</code>	커서의 y 좌표를 지정한 위치로 이동한다.
<code>setheading(x) seth(x)</code>	터틀이 바라보는 방향을 바꾼다.
<code>home()</code>	터틀의 위치와 방향을 초기화한다.
<code>speed(sp)</code>	터틀의 속도를 바꾼다. (0: 최고 속도, 1: 느린 속도, 10: 빠른 속도)

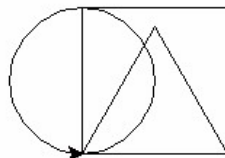
색상과 채우기, 상태 변경에 관련된 메소드

메소드	하는 일
<code>begin_fill() ... end_fill()</code>	<code>begin_fill()</code> 과 <code>end_fill()</code> 사이의 코드로 그린 그림을 색칠한다.
<code>color(c)</code>	터틀의 색깔을 변경한다.
<code>shape(s)</code>	터틀의 모양을 변경한다.
<code>shapsize(s)</code> <code> shapsize(w, h)</code>	터틀의 크기를 변경한다.
<code>pos() position()</code>	터틀의 현재 위치를 구한다.
<code>towards(x, y)</code>	현재 터틀이 있는 위치에서 특정 위치까지 바라보는 각도를 구한다.
<code>xcor()</code>	터틀의 x좌표를 구한다.
<code>ycor()</code>	터틀의 y좌표를 구한다.
<code>heading()</code>	터틀이 현재 바라보는 각도를 구한다.
<code>distance(x, y)</code>	현재 터틀이 있는 위치에서 특정 위치까지의 거리를 구한다.
<code>pendown() pd() down()</code>	펜을 내린다. (그릴 수 있는 상태)
<code>penup() pu() up()</code>	펜을 올린다.
<code>pensize(w) width(w)</code>	펜 굵기 변경
<code>isdown()</code>	펜이 내려져 있는 지 여부 확인

- 커서의 좌표는 기본적으로 원점 좌표인 (0, 0)
 - 머리가 향하는 진행방향, 헤딩heading
 - 초기 상태의 커서가 향하는 양의 x축 방향
- forward() 메소드를 만나면
머리가 향하는 방향으로 나아간다.



```
import turtle as t
t.setup(width = 400, height = 400)
# 삼각형 그리기
for x in range(3):
    t.forward(100)
    t.left(120)
# 사각형 그리기
for x in range(4):
    t.forward(100)
    t.left(90)
# 원 그리기
t.circle(50)
t.done()
```



• #변수를 사용

```
import turtle as t
t.setup(width = 400, height = 400)
d = 100
for x in range(3):
    t.forward(d)
    t.left(120)
for x in range(4):
    t.forward(d)
    t.left(90)
t.circle(d/2)
t.done()
```


for 문을 사용한 동일한 코드

for 문을 사용한 삼각형 그리기

turtle_triangle_with_for.py

```
import turtle as t
```

```
for _ in range(3): # 아래의 기능을 세 번 반복
    t.forward(100) #터틀을 헤딩 방향으로 100픽셀
    이동
    t.left(120) #터틀의 헤딩 방향을 왼쪽으로 120도
    회전
```

```
t.done()
```

– t.forward(100), t.left(90)을 4번 반복

for 반복문과 forward(), left() 메소드를 사용한 사각형

turtle_rect.py

```
import turtle as t
```

```
for _ in range(4):
    t.forward(100)
    t.left(90)
```

```
t.done()
```

원 그리기

- n을 100으로 초기화해서 for 루프의 반복 횟수를 100번으로 설정,
- $360/n$ 을 이용하여 3.6도씩 회전하고 전진

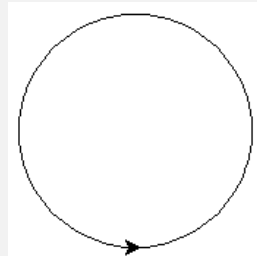
코드 7-19 : forward(), left() 명령을 사용한 원 그리기

turtle_circle_with_for.py

```
import turtle as t
```

```
n = 100
length = 5
for i in range(n):
    t.left(360/n)
    t.forward(length)

t.done()
```

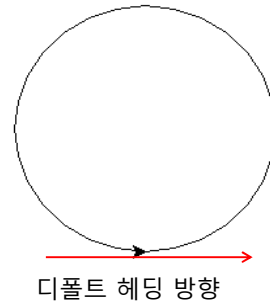


circle() 메소드를 이용한 원 그리기

- 인자로 원의 지름 입력
- 디폴트 헤딩 방향은 오른쪽

대화창 실습 : 원 그리기 메소드

```
>>> import turtle as t
>>> t.circle(100)
```



대화창 실습 : 크기가 다른 5개의 원 그리기

```
>>> import turtle as t
>>> t.circle(10)
    # 원 그리기는 디폴트 헤딩 방향을 기준으로 그린다
>>> t.circle(20)
>>> t.circle(30)
>>> t.circle(40)
>>> t.circle(50)
```

• 수행 결과



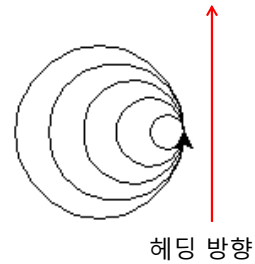
For문으로 변경해서 작성

setheading(90)을 추가

- 헤딩 방향이 위쪽으로 변경됨

대화창 실습 : 크기가 다른 5개의 원 그리기

```
>>> import turtle as t
>>> t.setheading(90)    # 헤딩 방향을 이동시킴 # 왼쪽 원
                        # 그리기
>>> t.circle(10)        # 왼쪽 원 그리기를 수행
>>> t.circle(20)
>>> t.circle(30)
>>> t.circle(40)
>>> t.circle(50)
```



For문으로 변경해서 작성

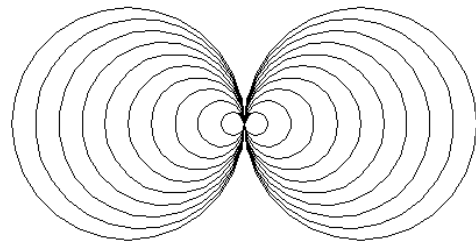
여러 개의 크기가 다른 원 그려보기

코드 7-20 : circle() 명령을 이용한 여러 개의 원 그리기

```
turtle_circle.py
import turtle as t

t.setheading(90)    # 왼쪽 원 그리기
for i in range(1, 11):
    t.circle(10*i)

t.setheading(270)   # 오른쪽 원 그리기
for i in range(1, 11):
    t.circle(10*i)
t.done()
```

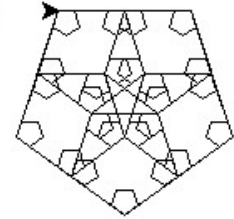


왼쪽 원은 진행방향을 90도 회전한 상태이며,
오른쪽 원은 진행방향을 270도 회전한 결과.
진행방향(setheading()) 이 중요한 역할을 한다)

정오각형을 그리는 프로그램

```
import turtle as t
t.setup(width = 400, height = 400)
n = 5
t.color("purple")
t.begin_fill()
for x in range(n):
    t.forward(50)
    t.left(360/n)
t.end_fill()
t.done()
```

```
nSides = 5
Size = 80
for step1s in range(nSides):
    turtle.forward(Size)
    turtle.right(360/nSide
```

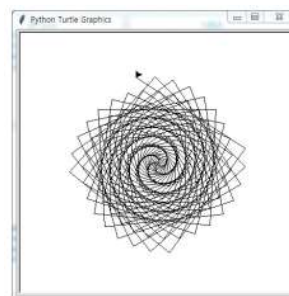


```
for Step2s in range(nSides):
    turtle.forward(Size/2)
    turtle.right(360/nSides)
```

```
for Step3s in range(nSides):
    turtle.forward(Size/8)
    turtle.right(360/nSides)
```

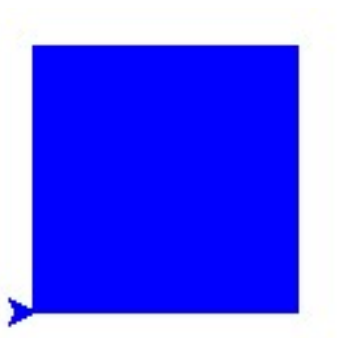
터틀 그래픽의 `setup()`, `forward()`, `left()`, `done()` 메소드

```
import turtle as t
t.setup(width = 400, height = 400)
for i in range(200):
    t.forward(i)
    t.left(93)
t.done()
```



터틀 그래픽을 이용한 색칠

- 한 변의 길이가 100 픽셀이고 내부가 칠해진 정사각형 그리기
- `begin_fill()`, `end_fill()` 메소드 이용



color() : 색상을 지정

코드 7-21 : `color()`, `begin_fill()`, `end_fill()`을 이용한 색상 사각형 그리기

```
turtle_fill_rect1.py
import turtle as t

t.color('blue')    # 파란색을 선택
t.begin_fill()    # 내부를 채움
for _ in range(4):
    t.forward(100)
    t.left(90)
t.end_fill()      # 사각형 내부를 파란색으로 채워서 그리기

t.done()
```

LAB 7-8 : 사각형과 원 그리기

1. 터틀 그래픽의 사각형 그리기 코드를 수정하여 다음과 같이 4개의 사각형을 만드시오. 이때 1,2,3,4 분면의 색상은 각각 파랑, 빨강, 노랑, 초록이 되도록 하시오



2. 위의 사각형 그리기를 위한 `draw_rect()` 함수를 정의하시오. 이 `draw_rect()` 함수를 `for` 문을 이용하여 4번 호출하여 동일한 결과가 나오도록 수정하시오(힌트 : `colors = ['blue', 'red', 'yellow', 'green']` 와 같은 색상 리스트를 만들고 해당 방향을 `90 * i` 연산을 통해 변경하여 0, 90, 180, 270도가 되도록 하시오).

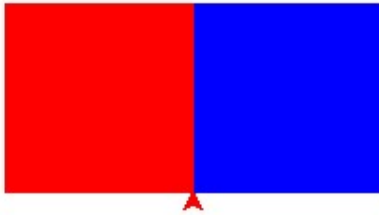
3. 터틀 그래픽의 사각형 그리기 코드를 수정하여 다음과 같이 4개의 원을 만드시오. 이때 원의 색상은 각각 파랑, 빨강, 주황, 초록이 되도록 하시오(힌트 : 사각형 그리기와 마찬가지로 `setheading()`을 이용하여 해당 방향이 0, 90, 180, 270도가 되도록 하시오).



여러 개의 다른색 사각형 그리기

코드 7-22 : 여러 개의 사각형 그리기

실행 결과



turtle_fill_rect2.py

import turtle as t

파란색 네모 그리기

t.color('blue')

t.begin_fill()

for _ in range(4):

 t.forward(100)

 t.left(90)

t.end_fill() # 사각형 내부를 파란색으로 채워서 그리기

t.setheading(90)

빨간색 네모 그리기

t.color('red')

t.begin_fill()

for _ in range(4):

 t.forward(100)

 t.left(90)

t.end_fill() # 사각형 내부를 빨간색으로 채워서 그리기

t.done()