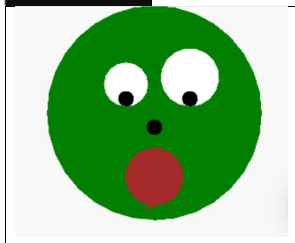
그래픽스 시작

1. 문제

다음 그림과 같이 Turtle 을 이용해서 그림을 그리시오

실행결과



조건

1) DrawCircle() 함수를 이용해서 7개의 원을 표현하시오

```
import turtle
def MyGoto(x_xx,x_yy):
    aaa.penup()
    aaa.goto(x_xx,x_yy)
    aaa.pendown()

def DrawCircle(x_xx, x_yy, x_radius, x_color):
    MyGoto(x_xx,x_yy)
```

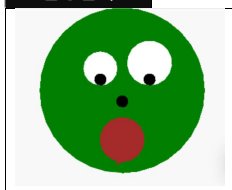


```
    aaa.color(x_color)
    aaa.fillcolor(x_color)
    aaa.begin_fill()
    aaa.circle(x_radius)
    aaa.end_fill()
aaa = turtle.Turtle()
aaa.speed(10)

DrawCircle(0,-100, 150,'green')
DrawCircle(-40, 60, 30,'white')
DrawCircle(-40, 60, 10,'black')
DrawCircle(50, 60, 40,'white')
DrawCircle(50, 60, 10,'black')
DrawCircle(0, 20, 10,'black')
DrawCircle(0, -80, 40,'brown')
```

2. 문제

다음 그림과 같이 Turtle 을 이용해서 그림을 그리시오

실행결과



조건

1) DrawCircle() 함수를 이용해서 7개의 원을 표현하시오
2) 관련된 함수는 mysmile.py 파일에 완성하시오

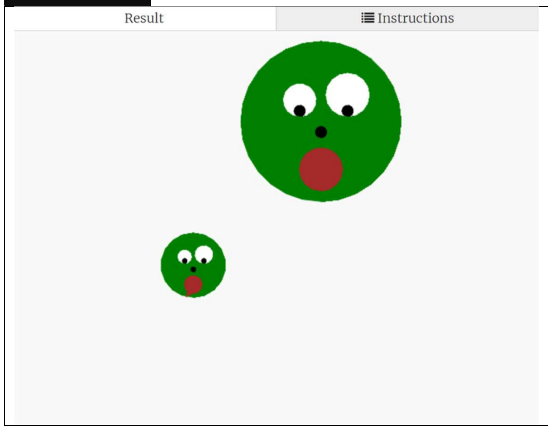
main.py

```
import turtle
from mysmile import *
aaa = turtle.Turtle()
aaa.speed(10)
DrawCircle(aaa,0,-100, 150,'green')
DrawCircle(aaa,-40, 60, 30,'white')
DrawCircle(aaa,-40, 60, 10,'black')
DrawCircle(aaa,50, 60, 40,'white')
DrawCircle(aaa,50, 60, 10,'black')
DrawCircle(aaa,0, 20, 10,'black')
DrawCircle(aaa,0, -80, 40,'brown')
```


3. 문제

다음 그림과 같이 smile 을 그리는 함수 MySmile() 을 mysmile.py에 정의하고 아래와 같이 크기와 위치가 다른 smile 2개를 그리시오.

실행결과



조건

1) DrawCircle() 함수를 포함하는 MySmile() 함수를 작성하시오.
2) 관련된 함수는 mysmile.py 파일에 완성하시오
3) Smile 의 크기와 위치를 조정할 수 있도록 파라메터를 추가 하시오.

main.py

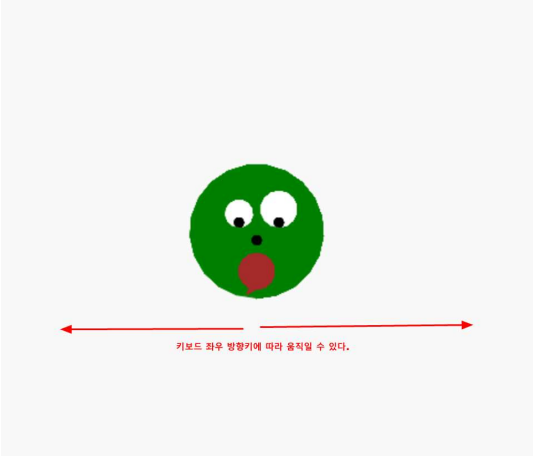
```
import turtle
from mysmile import * #case2
aaa = turtle.Turtle()
aaa.speed(0)
MySmile(aaa,0,0,0.9)
MySmile(aaa,-150,-150,0.3)
```

이벤트 등록

4. 문제

다음 그림과 같이 smile 이 오른쪽 왼쪽 방향키에 따라 움직이도록 main.py 파일을 작성하시오

실행결과



mysmile.py

```
def MyGoto(aaa,x_xx,x_yy):
    aaa.penup()
    aaa.goto(x_xx,x_yy)
    aaa.pendown()

def DrawCircle(aaa,x_xx, x_yy, x_radius, x_color):
    MyGoto(aaa,x_xx,x_yy)
```



```
    aaa.color(x_color)
    aaa.fillcolor(x_color)
    aaa.begin_fill()
    aaa.circle(x_radius)
    aaa.end_fill()

def MySmile(aaa,xx, yy, rw):
    DrawCircle(aaa,xx + 0*rw, yy -100*rw, 150*rw,'green')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 30*rw,'white')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 40*rw,'white')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy + 20*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy -80*rw, 40*rw,'brown')
```

main.py

```
import turtle
from mysmile import * #case2
aaa = turtle.Turtle()
sss = turtle.Screen() #####
aaa.speed(0)

xx = 0
yy = 0
```

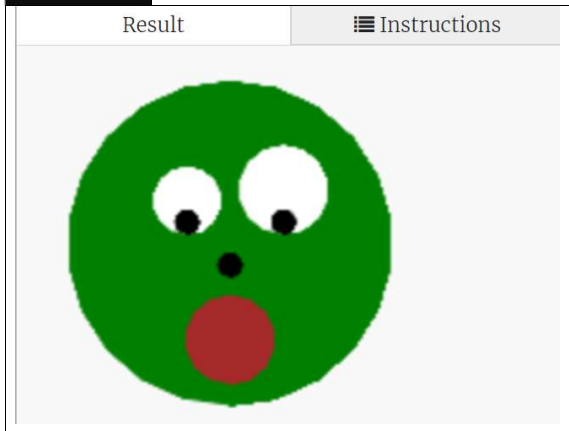
```
def MoveRight():
    aaa.clear()
    global xx
    xx = xx + 10
    MySmile(aaa,xx,yy,0.3)
    sss.update()

def MoveLeft():
    aaa.clear()
    global xx
    xx = xx - 10
    MySmile(aaa,xx,yy,0.3)
    sss.update()

MySmile(aaa,xx,yy,0.3)

####
sss.onkey(MoveRight,"Right")
sss.onkey(MoveLeft,"Left")
sss.listen()
sss.mainloop()
sss.tracer(0,0)
####
```


CAD 와 그래픽스의 결합

5. 문제

다음 그림과 같이 smile 이미지를 만들고, 이미지를 불러서 조건을 만족하는 프로그램을 작성하시오..

실행결과



조건

1) 4개 방향키의 모든 이벤트 함수를 만들고, 등록하시오

main.py

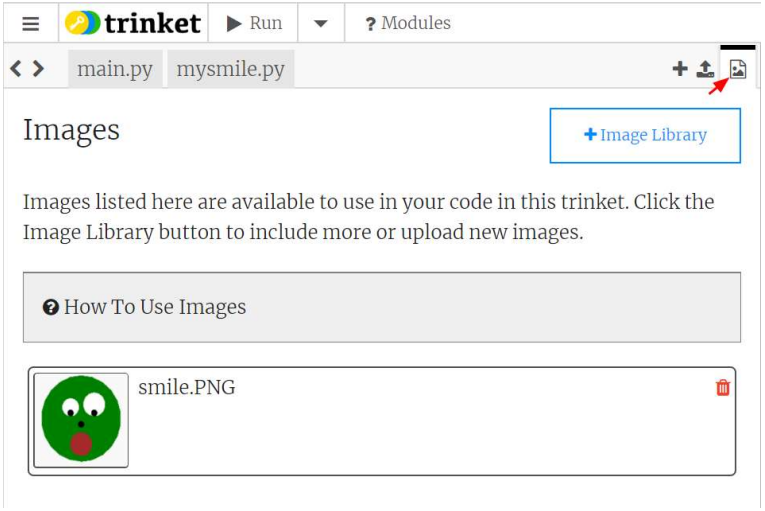
```
import turtle
aaa = turtle.Turtle()
sss = turtle.Screen()
aaa.speed(0)
```

```
sss.addshape("smile.PNG")
aaa = turtle.Turtle()
aaa.shape("smile.PNG")
aaa.left(90)

xx = 0
yy = 0
ss = 0.3

def MoveRight():
  global xx
  xx = xx + 10
  DrawFrame()

def MoveLeft():
  global xx
  xx = xx - 10
  DrawFrame()

def MoveUp():
  global yy
  yy = yy + 10
  DrawFrame()
```



```
def MoveDown():
  global yy
  yy = yy - 10
  DrawFrame()

def GrowUp():
  global ss
  ss = ss + 0.1
  DrawFrame()

def GrowDown():
  global ss
  ss = ss - 0.1
  DrawFrame()

def DrawFrame():
  global xx,yy,ss
  aaa.clear()
  aaa.goto(xx,yy)
  aaa.shape(10)



sss.onkey(MoveRight,"Right")
```

```
sss.onkey(MoveLeft,"Left")
sss.onkey(MoveUp,"Up")
sss.onkey(MoveDown,"Down")
sss.onkey(GrowUp,"a")
sss.onkey(GrowDown,"z")
sss.listen()
```



<그림> 이미지 등록 방법

6. 문제

다음 그림과 같이 오른쪽 왼쪽 방향키에 따라 smile이 움직이면서 오른쪽 상단에 smile의 중심 위치를 표시하도록 하시오.

실행결과

mysmile.py

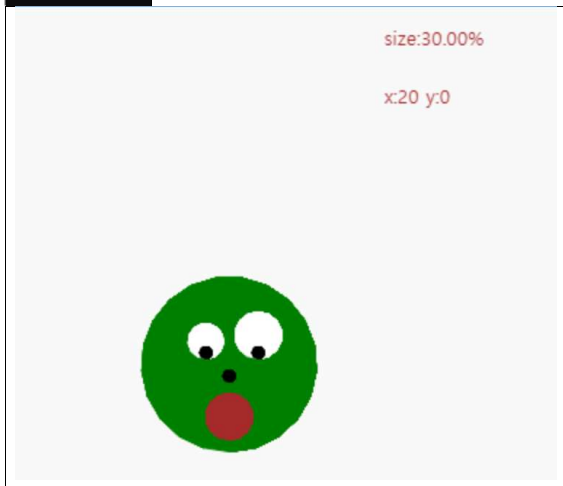
```
def MyGoto(aaa,x_xx,x_yy):
    aaa.penup()
    aaa.goto(x_xx,x_yy)
    aaa.pendown()

def DrawCircle(aaa,x_xx, x_yy, x_radius, x_color):
    MyGoto(aaa,x_xx,x_yy)
    aaa.color(x_color)
    aaa.fillcolor(x_color)
    aaa.begin_fill()
    aaa.circle(x_radius)
```

```
    aaa.end_fill()


def MySmile(aaa,xx, yy, rw):
    DrawCircle(aaa,xx + 0*rw, yy -100*rw, 150*rw,'green')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 30*rw,'white')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 40*rw,'white')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy + 20*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy -80*rw, 40*rw,'brown')
```


7. 문제

다음 그림과 같이. 'A', 'B' 버튼으로 smile 의 크기를 조정할 수 있도록 하시오.

실행결과



조건

1) 오른쪽 왼쪽 방향키에 따라 smile이 움직이면서 오른쪽 상단에 smile의 중심 위치를 표시하도록 하시오
2) 크기조정 함수 GrowUp() GrowDown() 정의

mysmile.py

```
def MyGoto(aaa,x_xx,x_yy):
    aaa.penup()
    aaa.goto(x_xx,x_yy)
    aaa.pendown()
```

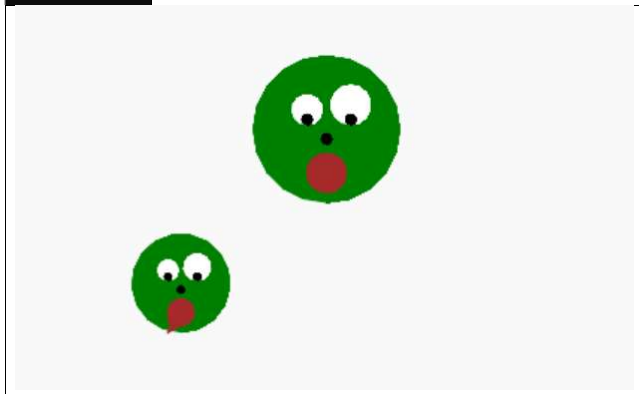
```
def DrawCircle(aaa,x_xx, x_yy, x_radius, x_color):
    MyGoto(aaa,x_xx,x_yy)
    aaa.color(x_color)
    aaa.fillcolor(x_color)
    aaa.begin_fill()
    aaa.circle(x_radius)
    aaa.end_fill()

def MySmile(aaa,xx, yy, rw):
    DrawCircle(aaa,xx + 0*rw, yy -100*rw, 150*rw,'green')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 30*rw,'white')
    DrawCircle(aaa,xx -40*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 40*rw,'white')
    DrawCircle(aaa,xx +50*rw, yy +60*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy + 20*rw, 10*rw,'black')
    DrawCircle(aaa,xx +0*rw, yy -80*rw, 40*rw,'brown')
```


Smile 클래스 작성

8. 문제

다음 그림과 같이 smile 이 오른쪽 왼쪽 방향키에 따라 움직이도록 main.py 파일을 작성하시오

실행결과



myclass.py

```
class CMySmile:
    def __init__(self,aaa,sss):
        self.aaa = aaa
        self.sss = sss

    def MyGoto(self,x_xx,x_yy):
        self.aaa.penup()
        self.aaa.goto(x_xx,x_yy)
```

```
        self.aaa.pendown()

    def DrawCircle(self, x_xx, x_yy, x_radius, x_color):
        self.MyGoto(x_xx,x_yy)
        self.aaa.color(x_color)
        self.aaa.fillcolor(x_color)
        self.aaa.begin_fill()
        self.aaa.circle(x_radius)
        self.aaa.end_fill()

    def MySmile(self, xx, yy, rw):
        self.xx = xx
        self.yy = yy
        self.rw = rw

        self.DrawCircle(self.xx + 0*self.rw, self.yy -100*self.rw, 150*self.rw,'green')
        self.DrawCircle(self.xx -40*self.rw, self.yy +60*self.rw, 30*self.rw,'white')
        self.DrawCircle(self.xx -40*self.rw, self.yy +60*self.rw, 10*self.rw,'black')
        self.DrawCircle(self.xx +50*self.rw, self.yy +60*self.rw, 40*self.rw,'white')
        self.DrawCircle(self.xx +50*self.rw, self.yy +60*self.rw, 10*self.rw,'black')
        self.DrawCircle(self.xx +0*self.rw, self.yy + 20*self.rw, 10*self.rw,'black')
        self.DrawCircle(self.xx +0*self.rw, self.yy -80*self.rw, 40*self.rw,'brown')

    def MyWrite(self,xx,yy,str):
```



```
        self.MyGoto(xx,yy)
        self.aaa.write(str)
        self.aaa.hideturtle()

    def DrawFrame(self):
        self.aaa.clear()
        self.MySmile(self.xx,self.yy,self.rw)
        self.MyWrite(100,150,'x:{0} y:{1}'.format(self.xx,self.yy))
        self.MyWrite(100,180,'size:{0:3.2}%'.format(self.rw*100))
        self.sss.update()

    def MoveRight(self):
        self.xx = self.xx + 10
        self.DrawFrame()

    def MoveLeft(self):
        self.xx = self.xx - 10
        self.DrawFrame()

    def MoveUp(self):
        self.yy = self.yy + 10
        self.DrawFrame()

    def MoveDown(self):
```

```
        self.yy = self.yy - 10
        self.DrawFrame()


    def GrowUp(self):
        self.rw = self.rw + 0.1
        self.DrawFrame()


    def GrowDown(self):
        self.rw = self.rw - 0.1
        self.DrawFrame()
```

main.py

```
import turtle


from myclass import CMySmile #case2


aaa = turtle.Turtle()
sss = turtle.Screen()
aaa.speed(0)


sm1 = CMySmile(aaa,sss)
sm1.MySmile(0,0,0.3)
```



```
sss.onkey(sm1.MoveRight,"Right")
sss.onkey(sm1.MoveLeft,"Left")
sss.onkey(sm1.MoveUp,"Up")
sss.onkey(sm1.MoveDown,"Down")
sss.onkey(sm1.GrowUp,"a")
sss.onkey(sm1.GrowDown,"z")
sss.listen()
sss.tracer(0,0)
sss.mainloop()
```