*HW7 과제 – 5/ 23*

**주제 :**

**그림 그리는 함수를 정의하고 좌우로 움직이기**

**이름 : 김민석**

**학번 : 20195125**

**메일 : planetside2@naver.com**

**소속 : 소프트웨어융합**

**과목 : 컴퓨터사고와 문제해결**

**목차**

1. **첫번째 코드**
2. **실행화면 사진**
3. **내용 설명**
4. **두번째 코드**
5. **실행화면 영상, 사진**
6. **내용 설명**
7. **첫 번째 코드**
8. **Main.py**

import turtle

from mysmile import \*

mycircle = turtle.Turtle()

mycircle.speed(10)

CpyP(mycircle, -150, -150, 0.2)

CpyP(mycircle, 0, 0, 0.8)

1. **mysmile.py**

**def MyGoto(mycircle, x\_xx, x\_yy):**

mycircle.penup()

mycircle.goto(x\_xx, x\_yy)

mycircle.pendown()

**def DrawCircle(mycircle, x\_xx, x\_yy, x\_radius, x\_color):**

MyGoto(mycircle, x\_xx, x\_yy)

mycircle.color(x\_color)

mycircle.fillcolor(x\_color)

mycircle.begin\_fill()

mycircle.circle(x\_radius)

mycircle.end\_fill()

**def CpyP(mycircle, x\_xx, x\_yy, x\_rw):**

DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy -100\*x\_rw, 150\*x\_rw, 'green')

DrawCircle(mycircle, x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 30\*x\_rw, 'white')

DrawCircle(mycircle, x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')

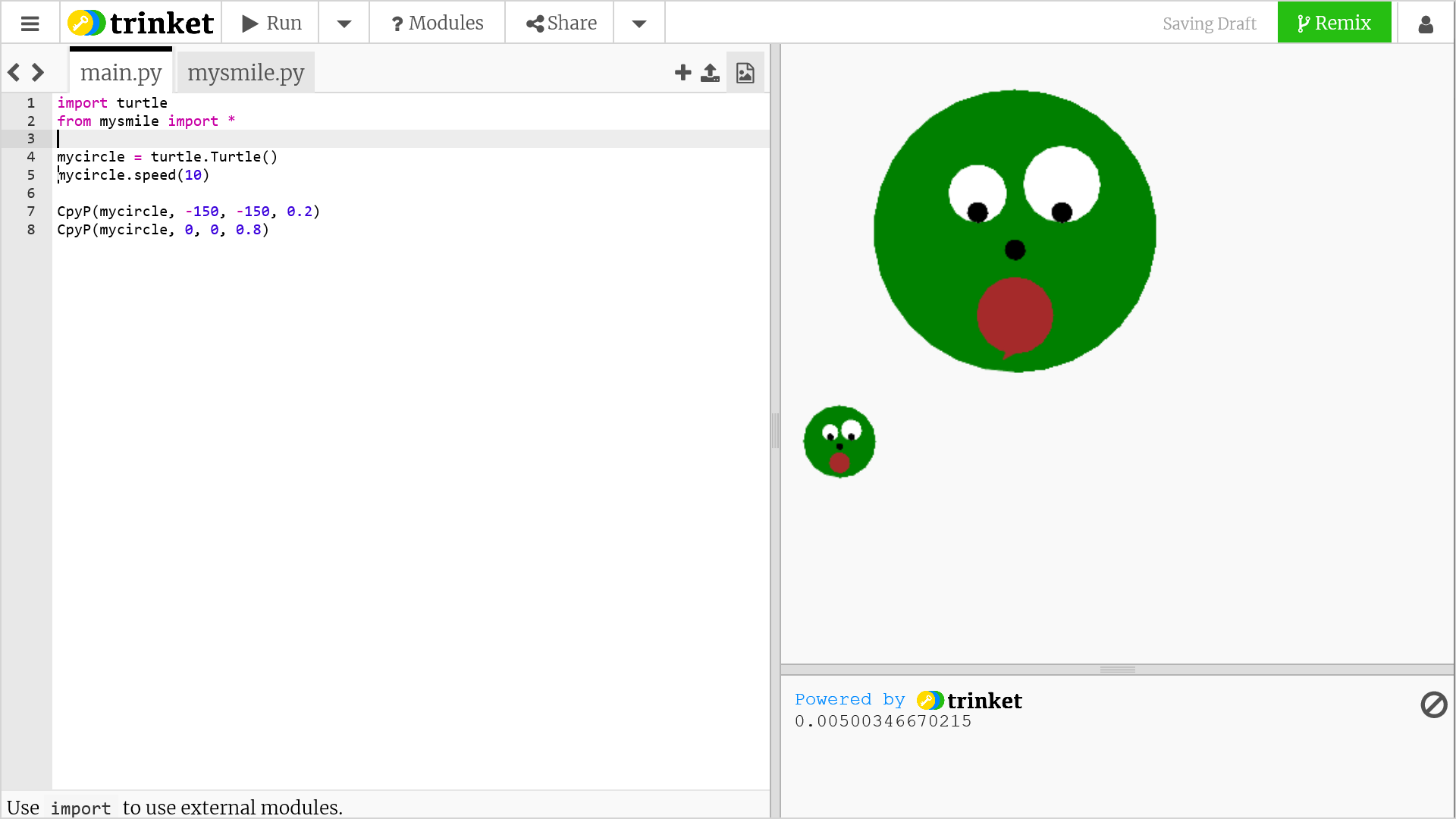
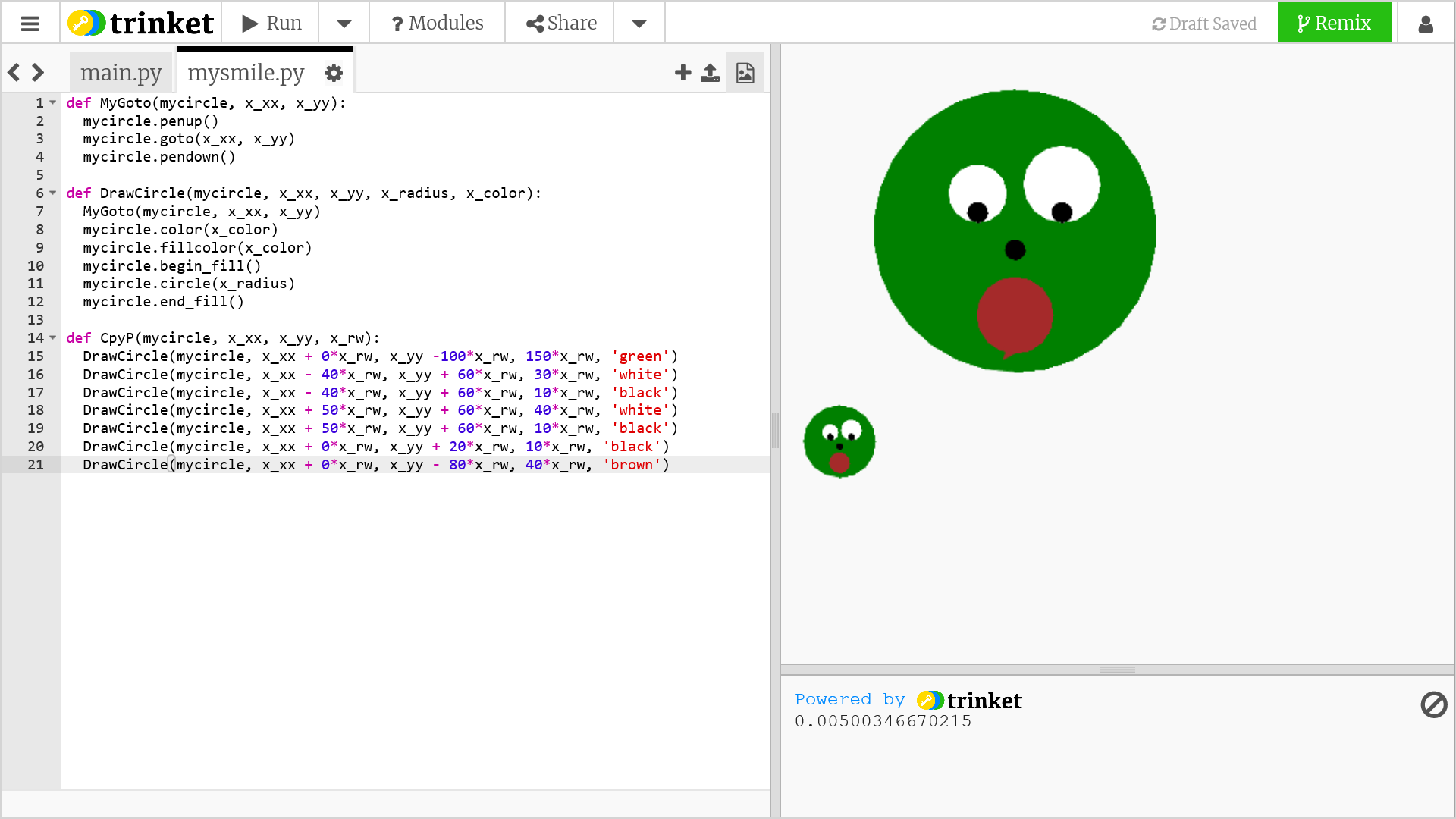
DrawCircle(mycircle, x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 40\*x\_rw, 'white')

DrawCircle(mycircle, x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')

DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy + 20\*x\_rw, 10\*x\_rw, 'black')

DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy - 80\*x\_rw, 40\*x\_rw, 'brown')

1. **실행화면 사진**

1. **구현내용 설명**

**1. import turtle**로 **turtle** 상자를 불러온다.

**2.** **mycircle = turtle.Turtle()**로 원을 그릴 수 있게 준비하고**mycircle.speed(10)**으로 **속도를 10으로 해서 빠르게 원을 그린다.**

**3.** 우선 **mysmile.py**라는 파일을 추가한 후, 원을 그릴 때 좌표가 바뀌므로

**def MyGoto(mycircle, x\_xx, x\_yy):**

**mycircle.penup()**

**mycircle.goto(x\_xx, x\_yy)**

**mycircle.pendown()** 로 **mysmile.py**에 넣어둔다.

**4.** 다음으로

**def DrawCircle(mycircle, x\_xx, x\_yy, x\_radius, x\_color):**

**MyGoto(mycircle, x\_xx, x\_yy)**

**mycircle.color(x\_color)**

**mycircle.fillcolor(x\_color)**

**mycircle.begin\_fill()**

**mycircle.circle(x\_radius)**

**mycircle.end\_fill()** 로 **MyGoto** 함수를 불러서 사용자가 원하는 좌표로 이동해서 지정된 크기로 원을 그릴 수 있게 **DrawCircle**을 정의한다.

**5.** 그리고 마지막 함수로써 앞서 쓴 **MyGoto**와 **DrawCircle**함수를 이용해서 사용자가 원하는 좌표와 반지름에 알맞은 그림을 언제든지 그릴 수 있게 함수를 정의한다.

**def CpyP(mycircle, x\_xx, x\_yy, x\_rw):**

**DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy -100\*x\_rw, 150\*x\_rw, 'green')**

**DrawCircle(mycircle, x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 30\*x\_rw, 'white')**

**DrawCircle(mycircle, x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 40\*x\_rw, 'white')**

**DrawCircle(mycircle, x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy + 20\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_xx + 0\*x\_rw, x\_yy - 80\*x\_rw, 40\*x\_rw, 'brown')**

**6.** 함수를 **mysmile.py**에 저장해둔 후 **main.py**에서

**from mysmile import \*** 로 **mysmile.py**에서 만든 **모든 함수를 main.py**에서 쓸 수 있게 만든다. 그리고 함수를 쓸 때마다 원을 그릴 변수(?)를 같이 보냄으로써 **smile.py에서 mycircle을 인식할 수 있게 해준다**.

**7.** 마지막으로

**CpyP(mycircle, -150, -150, 0.2)**

**CpyP(mycircle, 0, 0, 0.8)** 로 첫번째는 **x(-150), y(-150)**에 가서 기본 원 그림 크기의 **20퍼센트만큼의 크기**로 그려주고 두번째로 **호출해서 바이어스(0, 0좌표)에서 기본 원그림의 80퍼센트만큼**으로 나올 수 있게 선언한다.

**8.** 이와 같은 활용으로 **함수 CpyP(copy painting)을 쓸 때 x값, y값, 그림의 전체 크기들을 보냄으로써** 사용자가 어디든지 자유롭게 그림을 볼 수 있도록 할 수 있다.

1. **두 번째 코드**
2. **Main.py**

from turtle import Screen, Turtle

from mysmile import \*

screen = Screen()

screen.setup(width = 500, height = 500, startx = 0, starty = 0)

mycircle = Turtle()

CpyP(mycircle, 0, 0, 0.5, 0)

mov = 0

def left():

screen.reset()

global mov

mov -= 1

CpyP(mycircle, 0, 0, 0.5, mov\*(10))

def right():

screen.reset()

global mov

mov += 1

CpyP(mycircle, 0, 0, 0.5, mov\*(10))

screen.onkey(left, "Left")

screen.onkey(right, "Right")

screen.listen()

screen.exitonclick()

1. **smile.py**

def MyGoto(mycircle, x\_xx, x\_yy):

mycircle.penup()

mycircle.goto(x\_xx, x\_yy)

mycircle.pendown()

def DrawCircle(mycircle, x\_xx, x\_yy, x\_radius, x\_color):

MyGoto(mycircle, x\_xx, x\_yy)

mycircle.color(x\_color)

mycircle.fillcolor(x\_color)

mycircle.begin\_fill()

mycircle.circle(x\_radius)

mycircle.end\_fill()

def CpyP(mycircle, x\_xx, x\_yy, x\_rw, x):

mycircle.speed(0)

DrawCircle(mycircle, x + x\_xx + 0\*x\_rw, x\_yy -100\*x\_rw, 150\*x\_rw, 'green')

DrawCircle(mycircle, x + x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 30\*x\_rw, 'white')

DrawCircle(mycircle, x + x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')

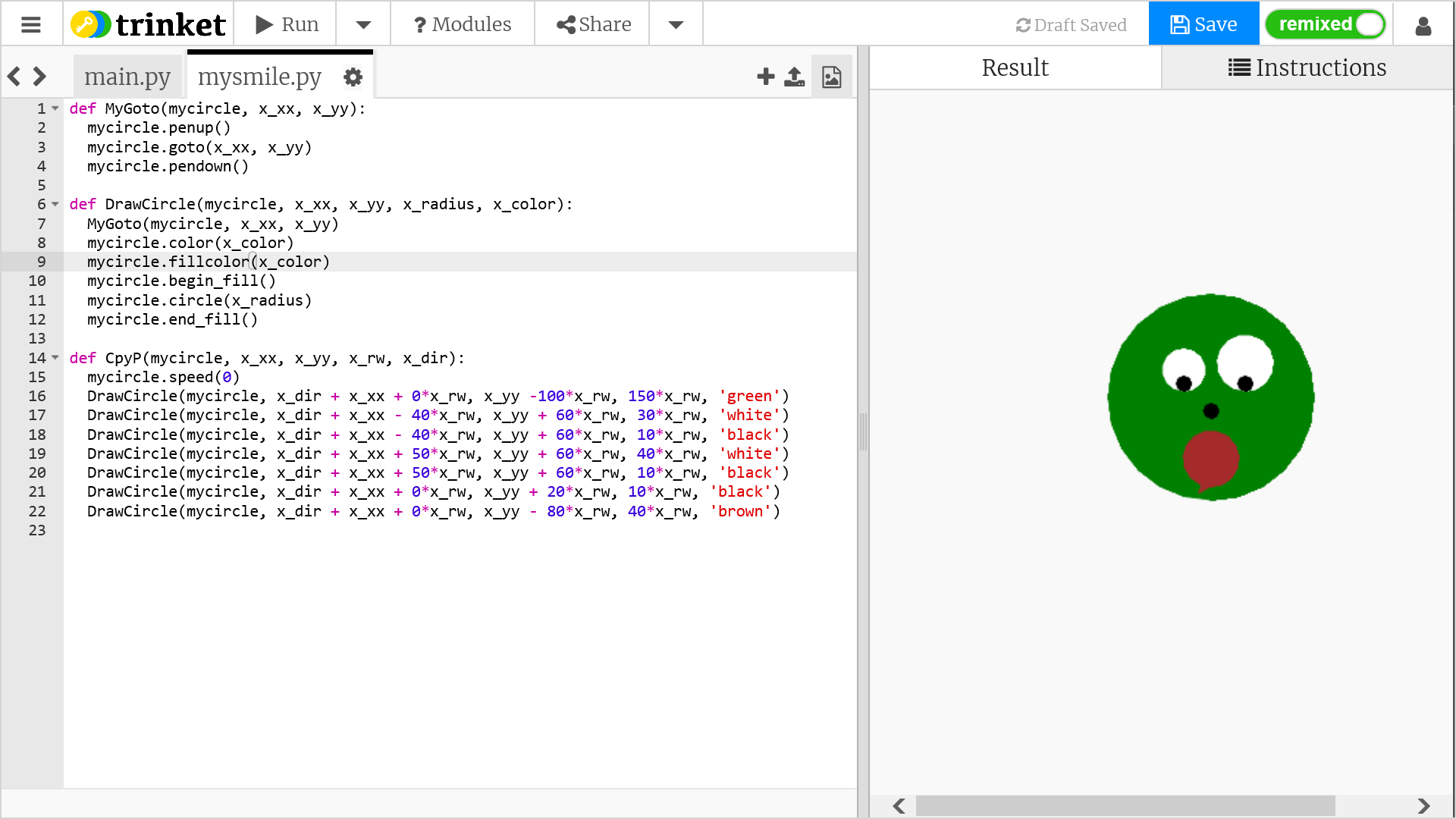
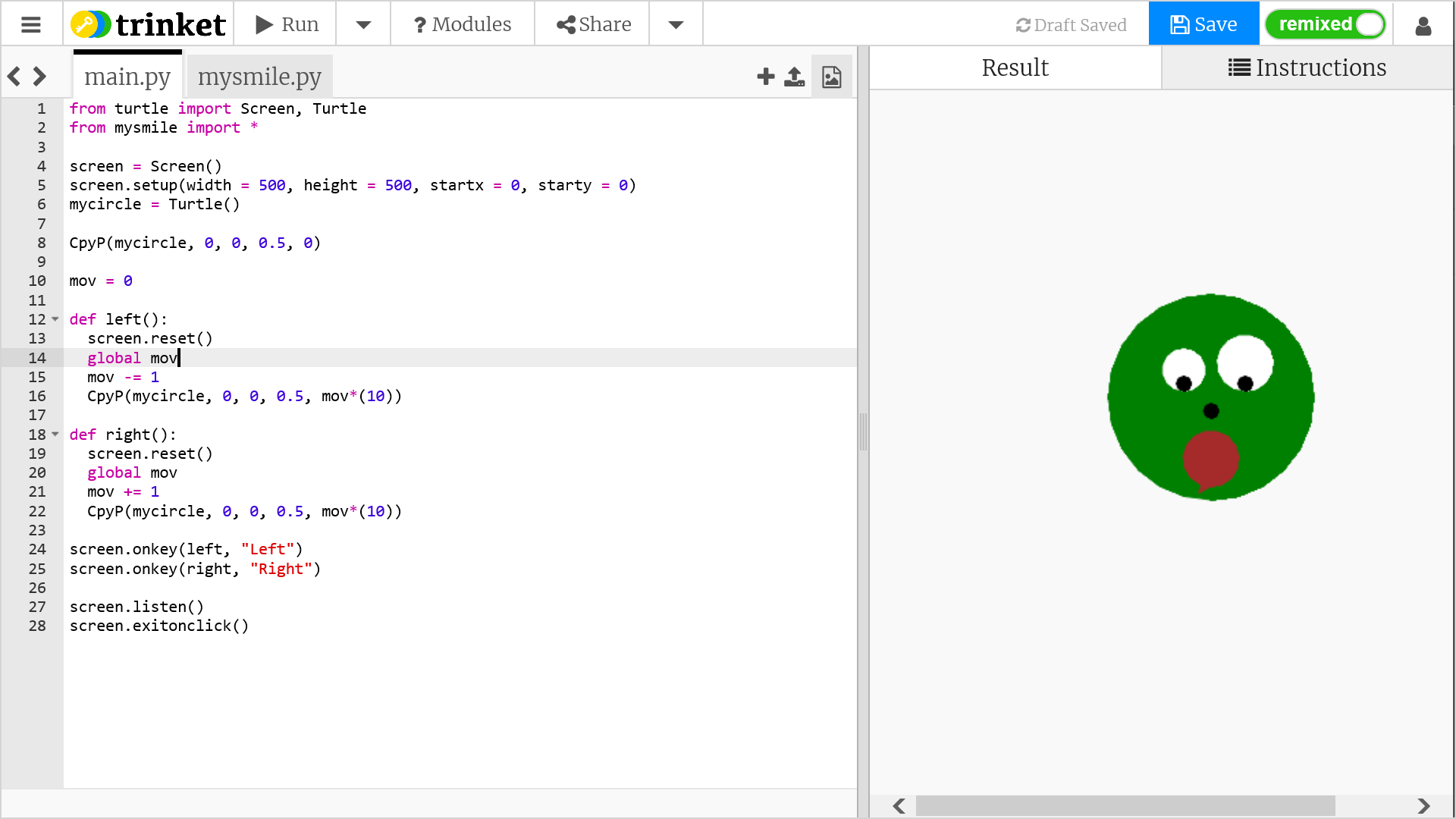
DrawCircle(mycircle, x + x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 40\*x\_rw, 'white')

DrawCircle(mycircle, x + x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')

DrawCircle(mycircle, x + x\_xx + 0\*x\_rw, x\_yy + 20\*x\_rw, 10\*x\_rw, 'black')

DrawCircle(mycircle, x + x\_xx + 0\*x\_rw, x\_yy - 80\*x\_rw, 40\*x\_rw, 'brown')

1. **실행화면 영상, 사진**





1. **구현내용 설명**

**1**. **from turtle import Screen, Turtle**로 **Screen과 Turtle**을 불러온다.

**2**. **mycircle = Turtle()**로 그림을 그릴 수 있게 하고 screen = Screen()으로 화면을 만든다. 그다음 **Screen.setup(width = 500, height = 500, startx = 0, starty = 0)**로 **폭과 너비를 500, 옵셋을 (0, 0)**으로 지정한다.

**3.** 새로운 파일 **mysmile.py**를 만들고 **main.py**에서 **smile.py**에 있는 함수를 모두 쓸 수 있게 **from mysmile import \***을 선언한다. 그리고 **전역변수 mov**를 **0으로 초기화한다**.

**4.** 원을 그릴 때 좌표가 바뀌므로

**def MyGoto(mycircle, x\_xx, x\_yy):**

**mycircle.penup()**

**mycircle.goto(x\_xx, x\_yy)**

**mycircle.pendown()** 로 **mysmile.py**에 넣어둔다.

**5.** 다음으로

**def DrawCircle(mycircle, x\_xx, x\_yy, x\_radius, x\_color):**

**MyGoto(mycircle, x\_xx, x\_yy)**

**mycircle.color(x\_color)**

**mycircle.fillcolor(x\_color)**

**mycircle.begin\_fill()**

**mycircle.circle(x\_radius)**

**mycircle.end\_fill()** 로 **MyGoto** 함수를 불러서 사용자가 원하는 좌표로 이동해서 지정된 크기로 원을 그릴 수 있게 **DrawCircle**을 정의한다.

**6.** 그리고 마지막 함수로써 앞서 쓴 **MyGoto**와 **DrawCircle**함수를 이용해서 사용자가 원하는 좌표와 반지름에 알맞은 그림을 언제든지 그릴 수 있게 함수를 정의한다. 그리고 **mycircle.speed(0)**으로 나중에 키를 눌렀을 때 마다 빠른 속도로 그릴 수 있게 한다.

**def CpyP(mycircle, x\_xx, x\_yy, x\_rw, x\_dir):**

**mycircle.speed(0)**

**DrawCircle(mycircle, x\_dir + x\_xx + 0\*x\_rw, x\_yy -100\*x\_rw, 150\*x\_rw, 'green')**

**DrawCircle(mycircle, x\_dir + x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 30\*x\_rw, 'white')**

**DrawCircle(mycircle, x\_dir + x\_xx - 40\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_dir + x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 40\*x\_rw, 'white')**

**DrawCircle(mycircle, x\_dir + x\_xx + 50\*x\_rw, x\_yy + 60\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_dir + x\_xx + 0\*x\_rw, x\_yy + 20\*x\_rw, 10\*x\_rw, 'black')**

**DrawCircle(mycircle, x\_dir + x\_xx + 0\*x\_rw, x\_yy - 80\*x\_rw, 40\*x\_rw, 'brown')**

**7.** screen.onkey는 함수와 ‘키’ 를 받는데 이때 키보드에서 정해진 ‘키’ 가 입력되면 그때 함수가 작동하는 식이므로 왼쪽과 오른쪽으로 이동해서 그림을 그릴 수 있도록

**def left():**

**screen.reset()**

**global mov**

**mov -= 1**

**CpyP(mycircle, 0, 0, 0.5, mov\*(10))**

**def right():**

**screen.reset()**

**global mov**

**mov += 1**

**CpyP(mycircle, 0, 0, 0.5, mov\*(10))로 시작 x, y는 (0, 0), 크기는 절반으로 왼쪽과 오른쪽에 대한 함수를 정의한다.**

**이때 매 키를 눌러 이동할 때마다 그전 그림을 지울 수 있도록 screen.reset()을 쓰고 전역변수를 각 방향함수 안에서 수정할 수 있게 해주는 global mov와 mov += 1 로 점점 멀리 갈 수 있게 설정한다.**

**8.** 마지막으로

**screen.onkey(left, "Left")**

**screen.onkey(right, "Right")**

**screen.listen()**

**screen.exitonclick()로 키보드에서 각각 “Left”, “Right”를 입력 받았을 때 각각 방향 함수 left와 right를 실행하도록 해준다. 이때 키보드에서 입력 받을 수 있는 screen.listen()을 넣고 다른 작업을 하거나 자유롭게 왔다 갔다 할 수 있게 screen.exitonclick()도 넣어주면서 마무리한다.**

(+)다른 방법으로는 **Image Library** 추가로 다음 이미지를 **mycircle.shape(“python\_pict.png”)**로 모양을 바꿔서 그 이미지를 가진 **mycircle**만 자유롭게 좌우로 왔다 갔다 할 수 있게 해주는 방법도 있었다.

