객체지향프로그래밍 이해하기 2

13주차_02

한동대학교
김경미 교수

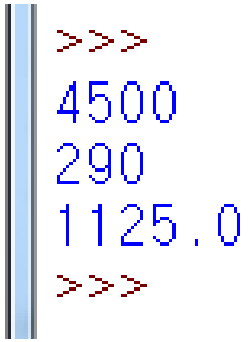
Methods 정의

- **Method는 두 가지 면에서 function과 다르다**
  - method는 한 class에 속하며 class 안에서 정의된다
  - method를 정의하는 첫 번째 매개변수(parameter)는 class의 instance를 위해 reference "self"가 있어야 한다
  - method를 호출 할 때는 이 매개변수 "self" 없이 사용한다

```
class Shape:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def area(self):
        return self.x * self.y

print(rectangle.area())
```

Methods

```
class Shape:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def area(self):
        return self.x * self.y
    def perimeter(self):
        return 2 * self.x + 2 * self.y
    def describe(self,text):
        self.description = text
    def authorName(self,text):
        self.author = text
    def scaleSize(self,scale):
        self.x = self.x * scale
        self.y = self.y * scale

rectangle = Shape(100, 45)
print(rectangle.area())
print(rectangle.perimeter())
rectangle.describe("A wide rectangle, more than twice as wide as it is tall")
rectangle.scaleSize(0.5)
print(rectangle.area())
```



연산자 중복정의(Operator Overloading)

- **사용자가 정의하는 객체 내에서 필요한 연산자를 정의할 때**
  - 기존에 사용하는 내장형 연산자의 데이터 타입과 형태와 동작이 유사하도록 재정의 하는 것
  - 연산자 중복 선언은 벡터, 행렬 등 수치연산에서 자주 사용

수치연산자 1

- **수치연산을 위해 이미 정의된 method**

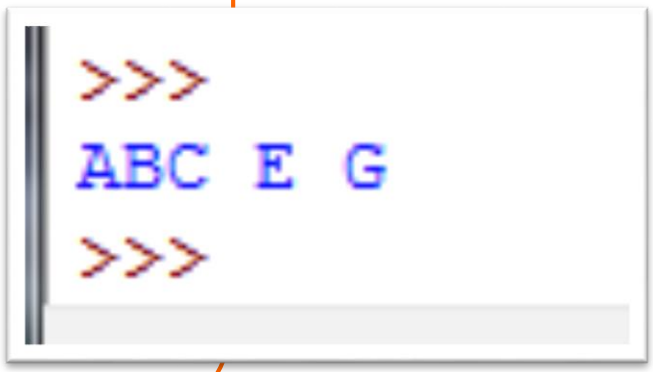
Method	Operator	사용 예
__add__(self, other)	+ (이항)	A + B, A += B
__sub__(self, other)	- (이항)	A – B, A -= B
__mul__(self, other)	*	A * B, A *= B
__truediv__(self, other)	/	A / B, A /= B
__floordiv__(self, other)	//	A // B, A //= B
__mod__(self, other)	%	A % B, A %= B
__divmod__(self, other)	divmod()	divmod(A, B)
__pow__(self, other[,modulo])	pow(), **	Pow(A, B), A ** B
__lshift__(self, other)	<<	A<<B, A<<=B
__rshift__(self, other)	>>	A>>B, A>>=B

수치연산자 2

Method	Operator	사용 예
__and__(self, other)	&	A & B, A &= B
__xor__(self, other)	^	A ^ B, A ^= B
__or__(self, other)	|	A | B, A |= B
__abs__(self, other)	abs()	abs(A)
__pos__(self, other)	+ (단항)	+A
__neg__(self, other)	- (단항)	-A
__invert__(self, other)	~	~A(비트 연산자)

연산자 중복정의 예제

```
class GString:
    def __init__(self, init=None):
        self.content = init

    def __sub__(self, str):                    # '-' 연산자 중복정의
        for i in str:
            self.content = self.content.replace(i, ' ')
        return GString(self.content)

    def __abs__(self):                         # 'abs' 연산자 중복정의
        return GString(self.content.upper())

    def  Print(self):
        print(self.content)

g = GString("ABcdefg")
g = g - "df"
g = abs(g)
g.Print()
```



연습문제 1

- **이전 슬라이드 예제에서 '*' 연산자를 사용하면**
  - 문자열 전체가 곱하는 정수배가 되도록 정의하고
  - 실행해 보시오

연습문제 1 코드 (1)

```
class GString:
    def __init__(self, init=None):
        self.content = init

    def __sub__(self, str):
        for i in str:
            self.content = self.content.replace(i, ' ')
        return GString(self.content)

    def __abs__(self):
        return GString(self.content.upper())

    def __mul__(self, int):
        return GString(self.content * int)

    def Print(self):
        print(self.content)
```

연습문제 1 코드 (2)

```
g = GString('ABcdefg')
g = g * 3
g.Print()

g = GString('KMooC')
g = g * 5
g.Print()
```

```
>>>
ABcdefgABcdefgABcdefg
KMooCKMooCKMooCKMooCKMooC
>>>
```

특별한 Class 속성들 (Attributes)

- Self 정의된 class 속성(attribute)들을 제외하고, class는 몇 가지 특별한 속성들을 가진다
- 이 속성들은 객체 모듈(object module)에 의해 제공된다

Attributes Name	Description
__init__	Class의 객체(object)가 예시될 때 바로 동작한다. 객체(object)를 초기화하는 것이 목적이다.
__dict__	Instance에서 공간 생성
__doc__	Document reference
__name__	Class 이름
__module__	Class를 구성하는 module 이름
__bases__	모든 superclass를 포함하는 집합

상속 (Inheritance) 이란?

- **Class가 여러 개 정의 되었을 때, 부모클래스가 자식 클래스로 부모클래스의 모든 속성(데이터, method)를 물려주는 것**
  - 공통 속성은 부모클래스에 정의하고
  - 하위클래스에서는 자신만이 사용 가능한 것들을 정의한다

상속 (Inheritance)

Inherit Syntax:
class subclass(superclass):

```
class Person:
    def speak(self):
        print("I can speak")

class Man(Person):
    def wear(self):
        print("I wear pants")

class Woman(Person):
    def wear(self):
        print("I wear skirt")

man = Man()
man.speak()                    # Person.speak()
```

```
>>>
I can speak
>>>
```

상속, method 재정의

- **부모클래스에서 상속받은 method를 수정할 필요가 있을 때**
  - 부모클래스 내의 method를 고치면, 상속받은 다른 클래스에 문제가 생길 수도 있다
  - 그래서 자식클래스에서 동일한 이름의 method를 일부 수정하여 사용 가능하다

상속, method 재정의

```
class Person:
    def speak(self):
        print("I can speak")
class Man(Person):
    def wear(self):
        print("I wear pants")
class Woman(Person):
    def wear(self):
        print("I wear skirt")
    def speak(self):
        print("I can speak and dance")

m = Man()
w = Woman()
m.speak()
m.wear()
w.speak()
w.wear()
```

```
>>>
I can speak
I wear pants
I can speak and dance
I wear skirt
>>>
```

연습문제 2

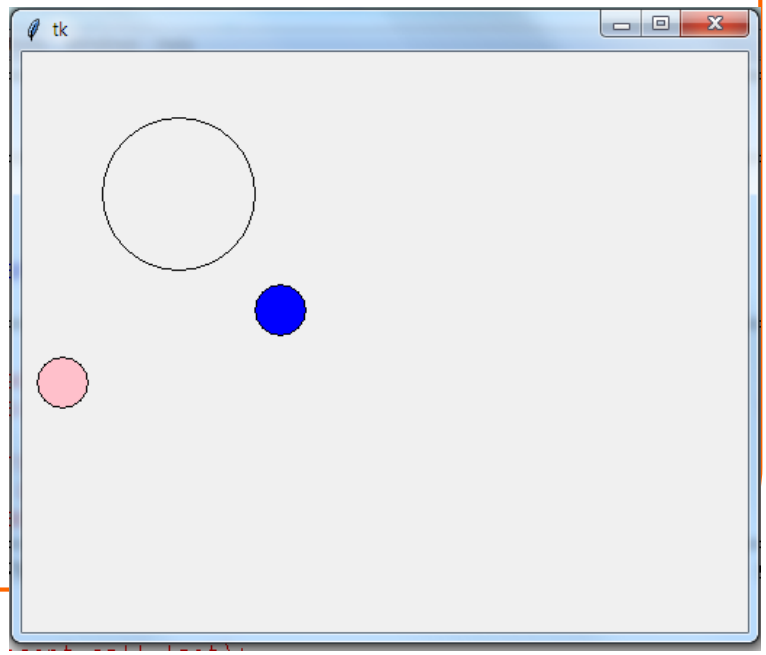
- **원을 그리기 위해 “class Circle”을 디자인하라**
  - Parameter: size
  - methods;: move_x, move_y, move_xy
- **색이 있는 원을 위해 “class f_Circle”을 디자인하라**
  - Parameters: size, color
  - Class circle로부터 상속
- **원 한 개를 그리고 그것을 (x, 0)으로 옮겨라**
- **한 개의 파랑 원을 그리고, 그것을 (x, y)로 옮겨라**
- **한 개의 분홍 원을 그리고, 그것을 (0, y)로 옮겨라**

연습문제 2 코드 (1)

```
from tkinter import *
import time

class Circle:
    def __init__(self, canvas):
        self.canvas = canvas
        self.id = canvas.create_oval(150,150,45,45)
    def move_x(self, x):
        self.x = x
        self.canvas.move(self.id, self.x, 0)
    def move_y(self, y):
        self.y = y
        self.canvas.move(self.id, 0, self.y)
    def move_xy(self,x, y):
        self.x = x
        self.y = y
        self.canvas.move(self.id, self.x, self.y)

#continue to..
```

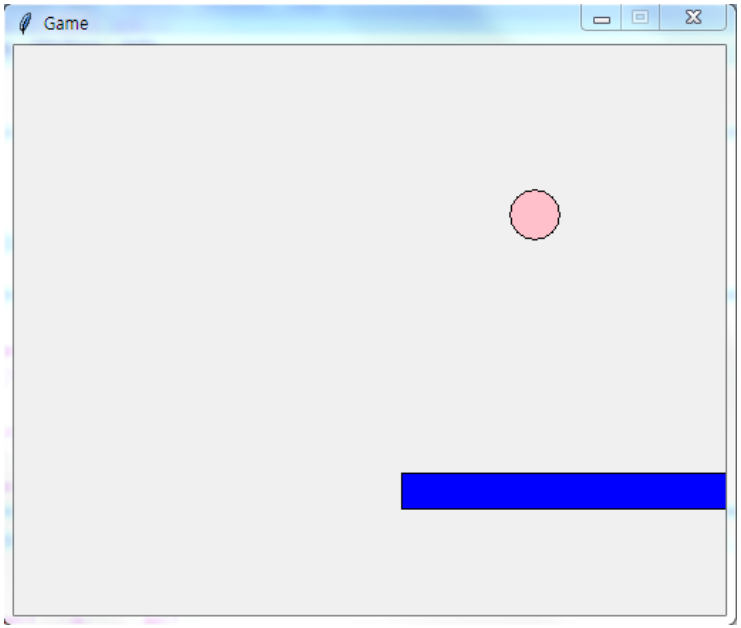
연습문제 2 코드 (2)

```
class f_Circle(Circle):
    def __init__(self, canvas, color):
        self.canvas = canvas
        self.id = canvas.create_oval(10,10,45,45, fill=color)

tk = Tk()
canvas = Canvas(tk, width=500, height=400, bd=0, highlightthickness=0)
canvas.pack()

circ = Circle(canvas)
bcirc = f_Circle(canvas, "blue")
rcirc = f_Circle(canvas, "pink")
canvas.pack()

circ.move_x(10)
bcirc.move_xy(150,150)
rcirc.move_y(200)
canvas.pack()
```



Game Bounce

- **간단한 게임 만들기**
  - 패들과 공이 존재
  - 공이 움직인다
  - 공이 떨어지지 않도록 패들을 움직인다



Step 0. 전체 구성

```
# import modules
from tkinter import *
import random
import time

# create ball
....
# create paddle
.....

# main process
...
```

Step 1. Create Ball 1

```
# create class Ball
class Ball:
    def __init__(self, canvas, paddle, color):
        self.canvas = canvas
        self.paddle = paddle
        self.id = canvas.create_oval(10,10,45,45, fill=color)     #ball size
        self.canvas.move(self.id,245,100)
        starts=[-3,-2,-1,1,2,3]
        random.shuffle(starts)
        self.x=starts[0]
        self.y=-3
        self.canvas_height=self.canvas.winfo_height()
        self.canvas_width=self.canvas.winfo_width()
        self.hit_bottom = False

    def hit_paddle(self,pos):
        paddle_pos=self.canvas.coords(self.paddle.id)
        if pos[2] >= paddle_pos[0] and pos[0] <= paddle_pos[2]:
            if pos[3] >= paddle_pos[1] and pos[3] <= paddle_pos[3]:
                return True
        return False
```

Step 1. Create Ball 2

```
def draw(self):
        self.canvas.move(self.id,self.x,self.y)
        pos=self.canvas.coords(self.id)
        if pos[1] <= 0:
            self.y=3
        if pos[3] >= self.canvas_height:
            self.hit_bottom = True
        if self.hit_paddle(pos) == True:
            self.y=-3
        if pos[0] <= 0:
            self.x=3
        if pos[2] >= self.canvas_width:
            self.x=-3
```

Step 2. Create Paddle

```
# create class paddle
class Paddle:
    def __init__(self, canvas, color):
        self.canvas = canvas
        self.id = canvas.create_rectangle(0,0,230,25,fill=color)          # paddle size
        self.canvas.move(self.id,200,300)
        self.x=0
        self.canvas_width=self.canvas.winfo_width()
        self.canvas.bind_all('<KeyPress-Left>', self.turn_left)
        self.canvas.bind_all('<KeyPress-Right>', self.turn_right)

    def turn_left(self,evt):
        self.x=-3

    def turn_right(self,evt):
        self.x=3

    def draw(self):
        self.canvas.move(self.id, self.x, 0)
        pos=self.canvas.coords(self.id)
        if pos[0] <= 0:
            self.x=0
        elif pos[2] >= self.canvas_width:
            self.x=0
```

Step 3. Main Process

```
#main process
tk = Tk()
tk.title("Game")
tk.resizable(0,0)
tk.wm_attributes("-topmost", 1)

canvas = Canvas(tk, width=500, height=400, bd=0, highlightthickness=0)
canvas.pack()
tk.update()

paddle = Paddle(canvas, 'blue')                #show paddle
ball = Ball(canvas, paddle, 'pink')            #show ball

while 1:
    if ball.hit_bottom == False:               # until ball touch bottom line
        ball.draw()
        paddle.draw()
    tk.update_idletasks()
    tk.update()
    time.sleep(0.07)                           # moving speed of ball
```

숙제

- 연습문제 1, 2를 입력한 코드와
- 실행 결과를 캡쳐하여 게시판에 올리시오

요약

- Method를 다양하게 활용한다
- 연산자 중복 정의를 이해한다
- 상속을 이해한다
- 간단한 게임을 만든다

감사합니다

13주차_02 객체지향프로그래밍 이해하기 2