새내기 파이썬

12장 객체와 클래스

학습 목표

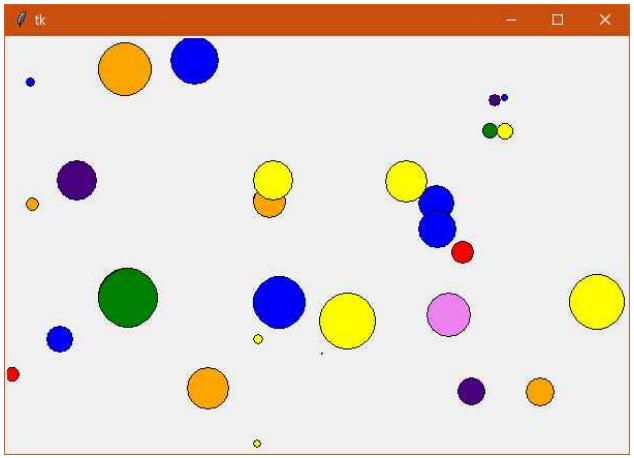
- 변수과 함수를 모아서 클래스로 작성할 수 있나요?
- 클래스에서 객체를 생성하는 방법을 알고 있나요?
- 생성자를 자유롭게 사용할 수 있나요?
- 인스턴스 변수와 클래스 변수의 차이점을 알고 있나요?
- 객체 지향을 응용하여서 실제 프로그램을 작성할 수 있나요?

이번 장에서 만들 프로그램

- 원을 클래스로 표현해보자. 원을 초기화하는 생성자는 만들어야 한다. 원은 반지름을 속성으로 가진다. 메소드로는 원의 넓이와 둘레를 반환하는 getArea()와 getPrimeter()를 정의한다.

```
원의 면적 314.1592653589793
원의 면적 62.83185307179586
```

- 공을 나타내는 Ball 클래스를 정의하고 애니메이션을 만들어보자.

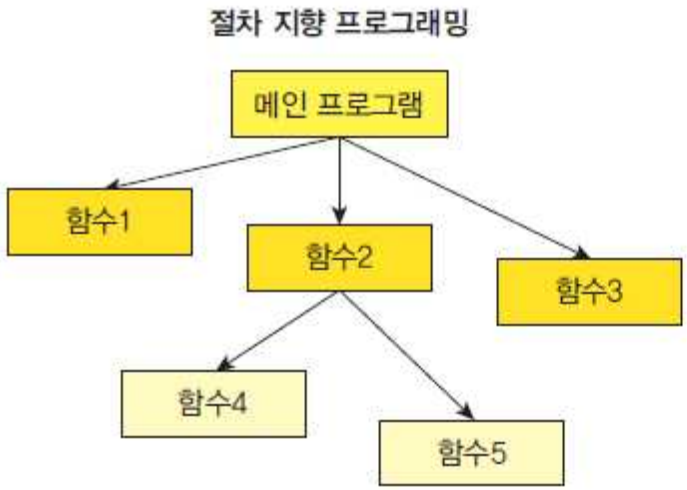


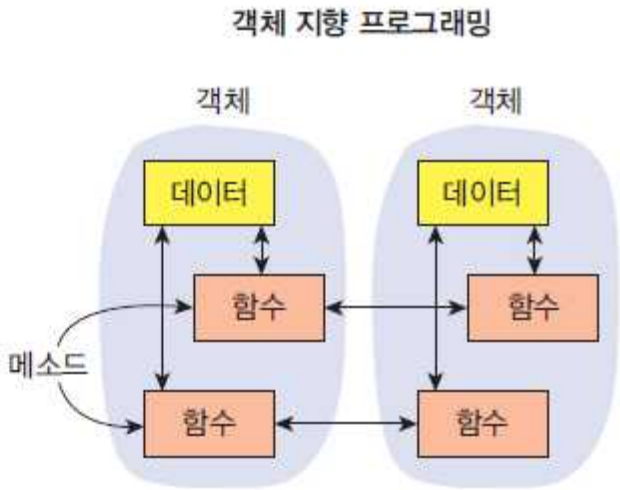
객체 지향 프로그래밍

- 객체 지향 프로그래밍(OOP: object-oriented programming)은 우리가 사는 실제 세계가 객체(object)들로 구성된 것과 비슷하게, 소프트웨어도 객체로 구성하는 방법이다. 실제 세계에는 사람, 자동차, 텔레비전, 세탁기, 냉장고 등의 많은 객체가 존재한다. 객체들은 객체 나름대로 고유한 기능을 수행하면서 다른 객체들과 메시지를 통하여 상호 작용한다.

절차 지향과 객체 지향

- 절차 지향 프로그래밍(procedural programming)은 프로시저(procedure)를 기반으로 하는 프로그래밍 방법이다.
- 객체 지향 프로그래밍(object-oriented programming)은 데이터와 함수를 하나의 덩어리로 묶어서 생각하는 방법이다.





절차 지향의 문제점

- 가장 중요한 단점은, 서로 관련된 데이터와 함수를 묶을 수가 없다는 점이다. 따라서 절차 지향 방법에서는 데이터가 프로그램의 중요한 부분임에도 불구하고 프로그래머들은 함수 작성에만 신경을 쓰게 된다.

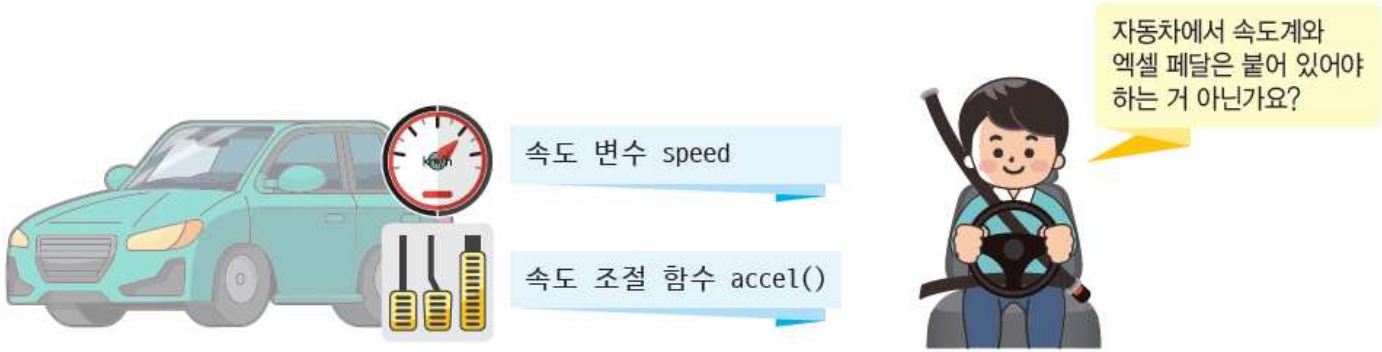
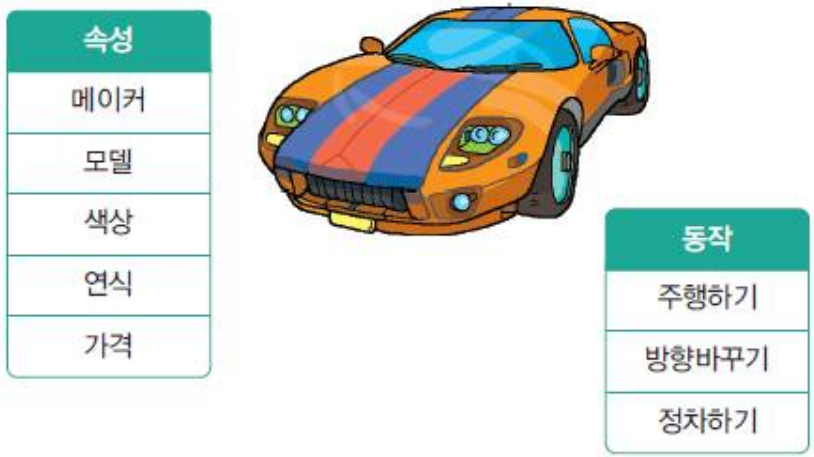


그림 12.3 절차 지향 프로그래밍의 문제점

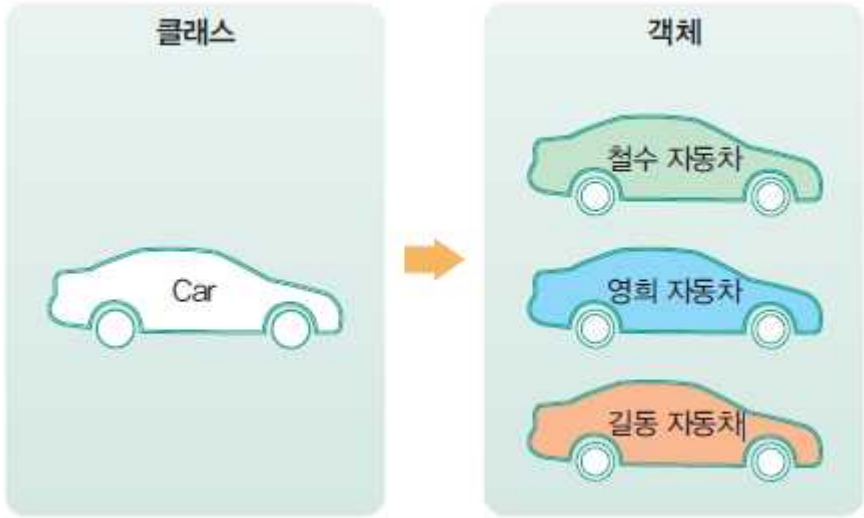
객체란?

- 객체(object)는 속성과 동작을 가진다.
- 자동차는 메이커나 모델, 색상, 연식, 가격 같은 속성(attribute)을 가지고 있다. 또 자동차는 주행할수 있고, 방향을 전환하거나 정지할 수 있다. 이러한 것을 객체의 동작(action)이라고 한다.



클래스

- 객체에 대한 설계도를 클래스(class)라고 한다. 클래스란 특정한 종류의 객체들을 찍어내는 형틀(template) 또는 청사진(blueprint)이라고도 할 수 있다.
- 클래스로부터 만들어지는 객체를 그 클래스의 인스턴스(instance)라고 한다.



파이썬에서는 모든 것이 객체이다.

- 파이썬에서는 모든 것이 객체로 구현된다. 정수도 객체이고 문자열도 객체이며 리스트도 객체이다.

```
>>> "Everything in Python is an object".upper()
'EVERYTHING IN PYTHON IS AN OBJECT'
```

캡슐화

- 개발자로서 클래스의 객체를 가지고 작업할 때는 객체가 어떻게 내부적으로 속성을 저장하고 어떻게 메소드들이 구현되는지 알 필요가 없다.
- 공용 인터페이스만 제공하고 구현 세부 사항을 감추는 것은 **캡슐화(encapsulation)**이라고 한다.

캡슐화

- 캡술화를 이렇게 이해하여보자. 우리가 자동차를 구입하면 자동차가 어떻게 동작되는지 알 필요가 있을까? 우리에게 중요한 것은 자동차를 사용하는 방법이다.

중간점검

1. 객체 지향 프로그래밍은 _____들을 조합하여서 프로그램을 작성하는 기법이다.
2. 절차 지향 프로그래밍의 가장 큰 단점은 무엇인가?
3. 캡슐화를 다시 설명해보자.
4. 클래스는 무엇인가?
5. 인스턴스는 무엇인가?
6. 클래스와 인스턴스의 관계는 무엇인가?

클래스 작성하기

Syntax 클래스 정의

형식

```
class 클래스이름 :
    def __init__(self, ...) :
        ...
    def 메소드1(self, ...) :
        ...
    def 메소드2(self, ...) :
        ...
```

예

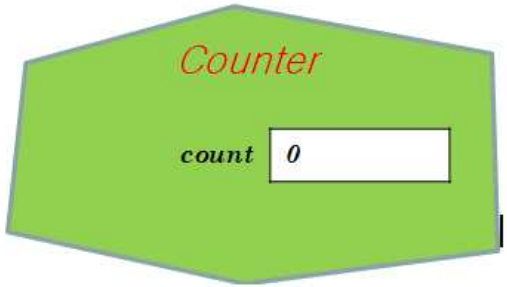
```
class Counter:
    def __init__(self):
        self.count = 0
    def increment(self):
        self.count += 1
```

생성자를 정의한다.

메소드를 정의한다.

Counter 클래스

-->



```
class Counter:
    def __init__(self):
        self.count = 0
    def increment(self):
        self.count += 1
```

생성자 정의

인스턴스 변수 생성

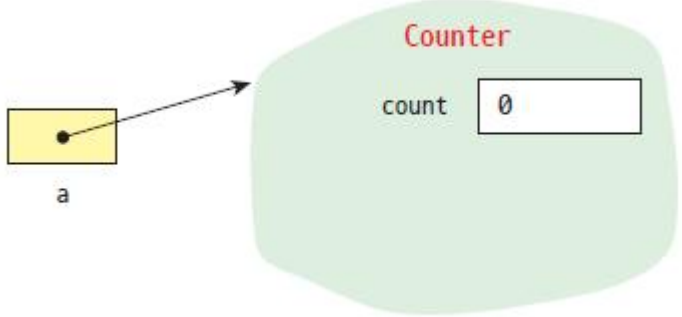
객체 생성

```
class Counter:
    def __init__(self):
        self.count = 0
    def increment(self):
        self.count += 1
```

생성자 정의

인스턴스 변수 생성

```
a = Counter()
```



객체의 멤버 접근

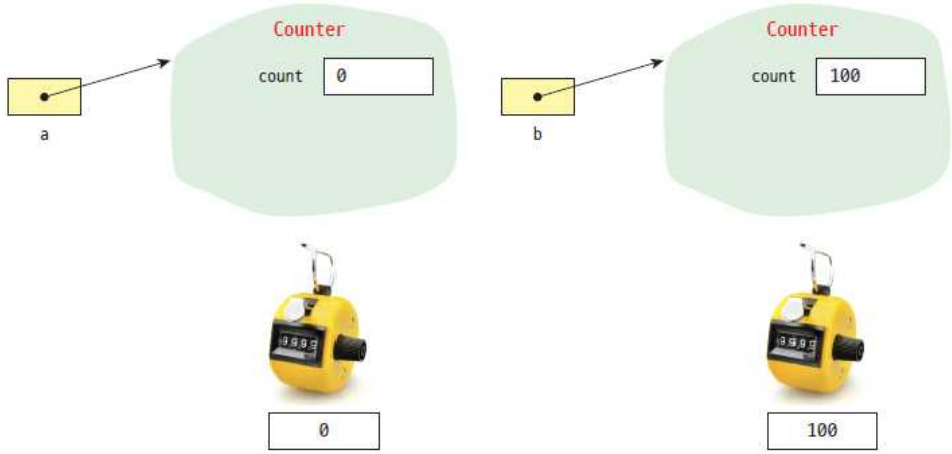
```
class Counter:
    def __init__(self):
        self.count = 0
    def increment(self):
        self.count += 1

a = Counter()
a.increment()
print("카운터의 값=", a.count)
```

객체 동작

```
카운터의 값= 1
```

하나의 클래스로 객체는 많이 만들 수 있다.

```
class Counter:
    def __init__(self,  initValue=0) :
        self.count = initValue

    def increment(self) :
        self.count += 1

a = Counter(0)              # 계수기를 0으로 초기화한다.
b = Counter(100)            # 계수기를 100으로 초기화한다.
```



중간점검

1. 클래스를 정의하는 데 사용되는 구문은 무엇인가?
2. 클래스 이름은 일반적으로 어떻게 작명하는가?
3. 클래스의 인스턴스를 어떻게 생성하는가?
4 인스턴스의 속성과 메소드에 어떻게 접근하는가?
5. 메소드는 무엇인가?
6. self의 목적은 무엇인가?
7. __init__() 메소드의 목적은 무엇인가?

Lab:TV 클래스 정의

- TV 클래스를 작성해보자.

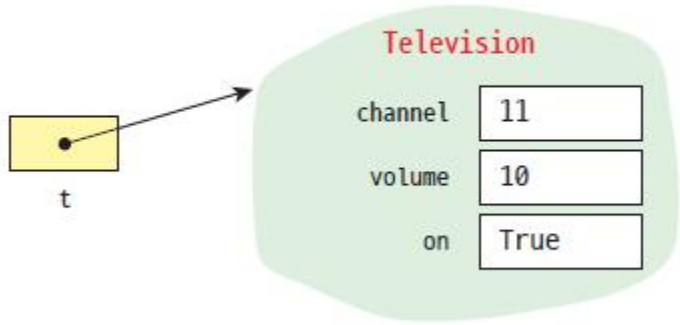
Solution:

```
class Television:
        def __init__(self, channel, volume, on):
                self.channel = channel
                self.volume = volume
                self.on = on

        def show(self):
                print(self.channel, self.volume, self.on)

        def setChannel(self, channel):
                self.channel = channel

        def getChannel(self):
                return self.channel
```

Solution:

```
t = Television(9, 10, True)
t.show()

t.setChannel(11)
t.show()
```

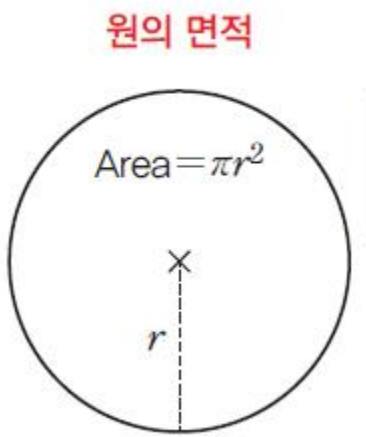
```
9 10 True
11 10 True
```



Lab: 원 클래스 정의

- 원을 클래스로 표현해보자. 클래스 이름은 Circle로 하자. 원을 초기화하는 생성자는 만들어야 한다. 원은 반지름을 속성으로 가진다. 메소드로는 원의 넓이와 둘레를 반환하는 getArea()와 getPrimeter()를 정의한다.

```
원의 면적 314.1592653589793
원의 둘레 62.83185307179586
```

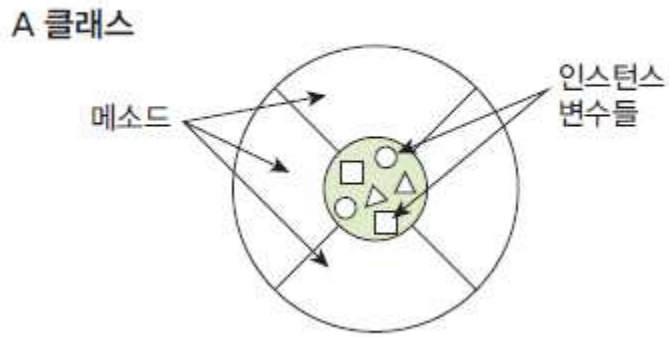


Solution:

```
import math

# Circle 클래스를 정의한다.
class Circle:
    def __init__(self, radius = 0):
        self.radius = radius

    def getArea(self):
        return  math.pi * self.radius * self.radius

    def getPerimeter(self):
        return 2 * math.pi * self.radius

# Circle 객체를 생성한다.
c = Circle(10)
print("원의 면적", c.getArea())
print("원의 면적", c.getPerimeter())
```

정보 은닉

- 파이썬에서 인스턴스 변수를 private으로 정의하려면 변수 이름 앞에 __을 붙이면 된다.
- private이 붙은 인스턴스 변수는 클래스 내부에서만 접근될 수 있다.



정보 은닉

```
class Student:
        def __init__(self, name=None, age=0):
                self.__name = name          # __가 변수 앞에 붙으면 외부에서 변경 금지
                self.__age = age            # __가 변수 앞에 붙으면 외부에서 변경 금지

obj=Student()
print(obj.__age)
```

```
...
AttributeError: 'Student' object has no attribute '__age'
```

접근자와 설정자

- 인스턴스 변수값을 반환하는 **접근자(getters)**
- 인스턴스 변수값을 설정하는 **설정자(setters)**

접근자와 설정자

```
class Student:
        def __init__(self, name=None, age=0):
                self.__name = name
                self.__age = age

        def getAge(self):
                return self.__age

        def getName(self):
                return self.__name

        def setAge(self, age):
                self.__age=age

        def setName(self, name):
                self.__name=name

obj=Student("Hong", 20)
obj.getName()
```

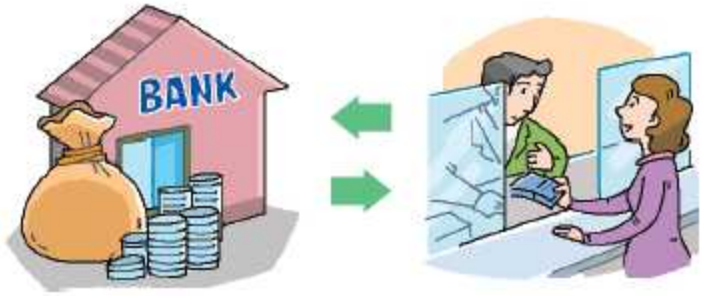
중간점검

1. 정보은닉은 왜 필요한가?
2. 접근자와 설정자의 이름은 어떻게 정하는가?
3. 클래스 안에서 어떤 변수를 private로 만들려면 어떻게 하면 되는가?

Lab:은행 계좌

- 우리는 은행 계좌에 돈을 저금할 수 있고 인출할 수도 있다. 은행 계좌를 클래스로 모델링하여 보자. 은행 계좌는 현재 잔액(balance)만을 인스턴스 변수로 가진다. 생성자와 인출 메소드 withdraw()와 저축 메소드 deposit() 만을 가정하자. 은행 계좌의 잔액은 외부에서 직접 접근하지 못하도록 하라.

```
통장에서 100 가 출금되었음
통장에 10 가 입금되었음
```

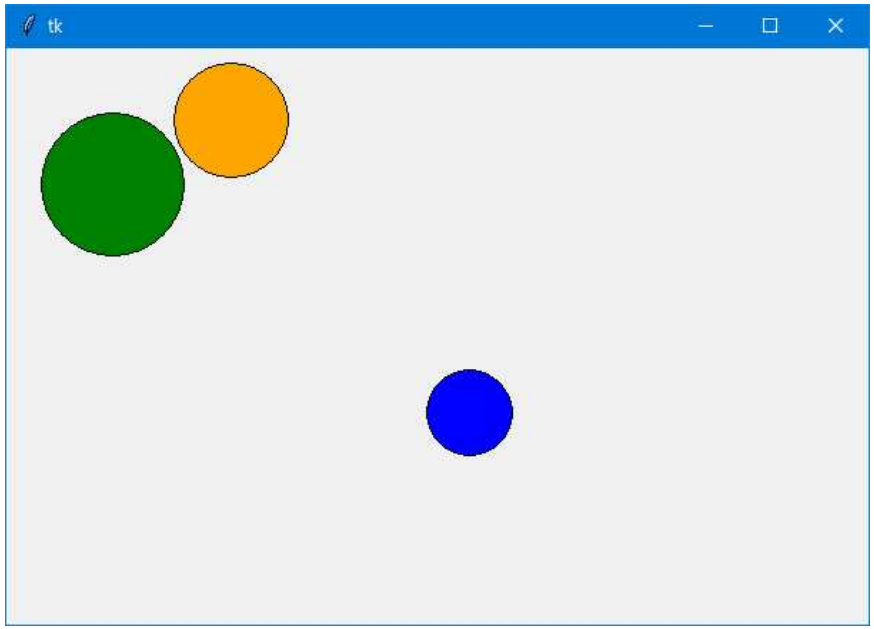


Solution:

```
class BankAccount:
    def __init__(self):
        self.__balance = 0

    def withdraw(self, amount):
        self.__balance -= amount
        print("통장에 ", amount, "가 인출되었음")
        return self.__balance

    def deposit(self, amount):
        self.__balance += amount
        print("통장에서 ", amount, "가 출금되었음")
        return self.__balance

a = BankAccount()
a.deposit(100)
a.withdraw(10)
```

Lab: 공 애니메이션 1

- 실제로 객체 지향 기법이 많이 사용되는 곳이 GUI 응용 프로그램이다. 이번 실습에서는 클래스를 사용하여 공이 움직이는 애니메이션을 작성해보자. 공의 개수를 마음대로 늘릴 수 있으면 공의 다양한 속성(크기, 색상, 속도)등도 자유롭게 표현할 수 있다.



```
from tkinter import *
import time
import random

window = Tk()

canvas=Canvas(window, width=600,height=400)
canvas.pack()

class Ball():
    def __init__(self,color,size):
        self.id=canvas.create_oval(0, 0, size, size,fill=color)
        self.dx=random.randint(1,10)
        self.dy=random.randint(1,10)

    def move(self):
        canvas.move(self.id,self.dx,self.dy)
        x0, y0, x1, y1 = canvas.coords(self.id)
        if y1 > canvas.winfo_height() or y0 < 0: # 공이 위쪽이나 아래쪽으로 벗어나면
            self.dy = -self.dy                                  # dy의 부호를 반전시킨다.
        if x1 > canvas.winfo_width() or x0 < 0:    # 공이 왼쪽이나 오른쪽으로 벗어나면
            self.dx = -self.dx                                  # dx의 부호를 반전시킨다.
```

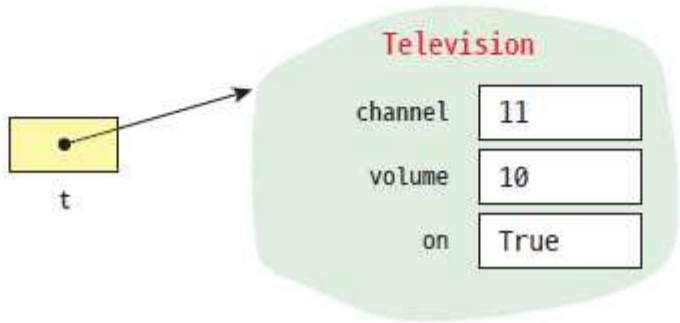
Solution:

```
ball1=Ball("blue", 60)
ball2=Ball("green",100)
ball3=Ball("orange",80)

while True:
    ball1.move()
    ball2.move()
    ball3.move()
    window.update()
    time.sleep(0.05)

window.mainloop()
```

객체 참조

- 파이썬에서 변수는 실제로 객체를 저장하지 않는다. 변수는 단지 객체의 메모리 주소를 저장한다. 객체 자체는 메모리의 다른 곳에 생성된다.

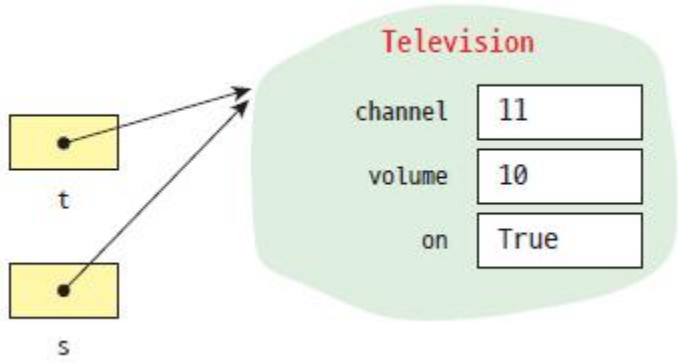
```
class Television:
    def __init__(self, channel, volume, on):
        self.channel = channel
        self.volume = volume
        self.on = on
    def setChannel(self, channel):
        self.channel = channel

t = Television(11, 10, True)
```



참조 공유

- 객체의 참조값을 저장하고 있는 변수를 다른 변수로 복사하면 어떻게 될까? 즉 다음과 같은 문장을 생각해보자.

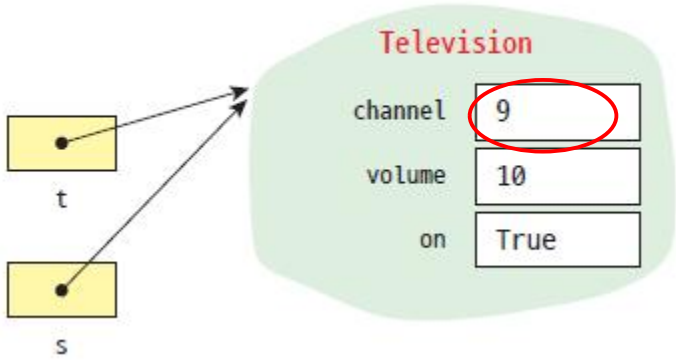
```
t = Television(11, 10, True)
s = t
```



참조 공유

- s를 통하여 객체를 수정하면 어떻게 될까? t가 가리키는 객체의 값도 변경된다.

```
t = Television(11, 10, True)
s = t
s.channel = 9
```



is, is not

- 2개의 변수가 동일한 객체를 참조하고 있는지를 검사하는 연산자가 있다. 바로 is와 is not 이다.

```
if s is t :
        print("2개의 변수는 동일한 객체를 참조하고 있습니다.")

if s is not t :
        print("2개의 변수는 다른 객체를 참조하고 있습니다.")
```

None 참조값

- 변수가 현재 아무것도 가리키고 있지 않다면 None으로 설정하는 것이 좋다.
- None은 아무것도 참조하고 있지 않다는 것을 나타내는 특별한 값이다.

```
myTV = None

if myTV is None :
        print("현재 TV가 없습니다. ")
```

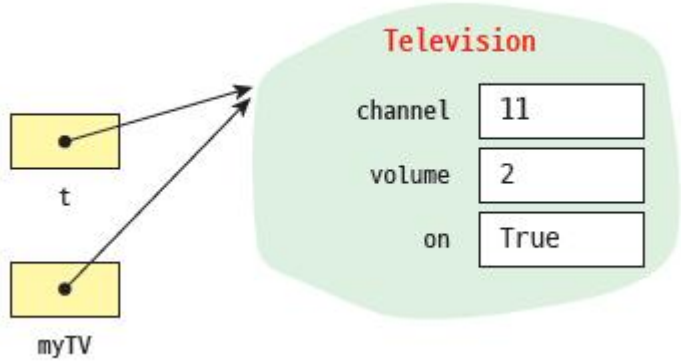
객체를 함수로 전달할 때

```
# 텔레비전을 클래스로 정의한다.
class Television:
        def __init__(self, channel, volume, on):
                self.channel = channel
                self.volume = volume
                self.on = on

        def show(self):
                print(self.channel, self.volume, self.on)

# 전달받은 텔레비전의 음량을 줄인다.
def setSilentMode(t):
        t.volume = 2

# setSilentMode()을 호출하여서 객체의 내용이 변경되는지를 확인한다.
myTV = Television(11, 10, True);
setSilentMode(myTV)
myTV.show()
```

11 2 True

객체를 함수로 전달할 때

- setSilentMode(t)에서는 Television 객체를 받아서 객체 안의 변수 volume을 무조건 2로 설정한다. 원본 객체가 변경될까? 아니면 원본 객체에는 영향이 없을까?



중간점검

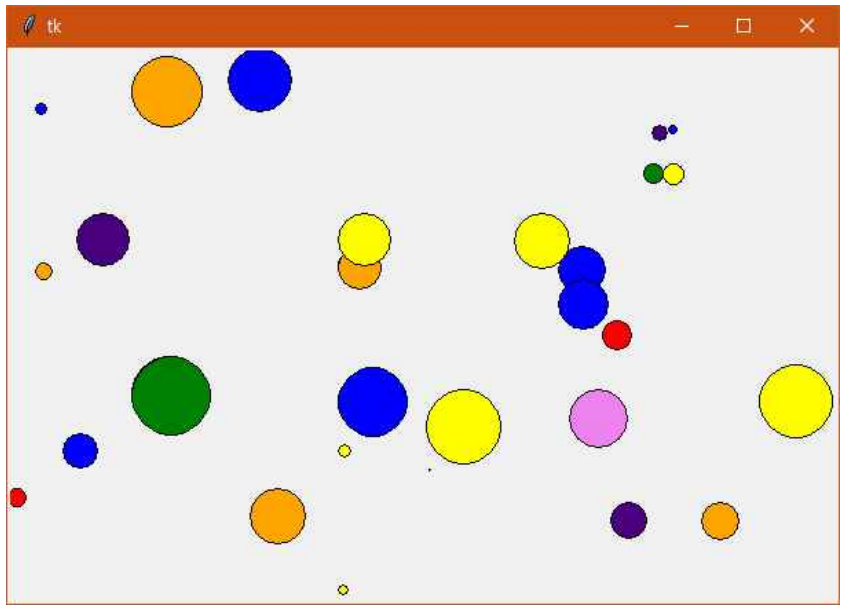
1. 객체를 함수로 전달하면 원본이 전달되는가? 아니면 복사본이 전달되는가?

2 다음과 같은 문장이 실행되면 내부적으로 어떤 일이 일어나는가?

```
t = Television(11, 10, True)
s = t
```

Lab: 공 애니메이션 2

- 앞 절의 실습에서 공을 3개 생성하여서 화면에서 움직이게 하였다. 그런데 공을 30개 정도 만들어서 움직이려면 어떻게 해야 할까? 이럴 때는 개별 변수를 사용하여 각 객체를 참조하는 것은 거의 불가능하다. 이런 경우에는 리스트를 생성하고 리스트에 객체를 저장하여야 한다.



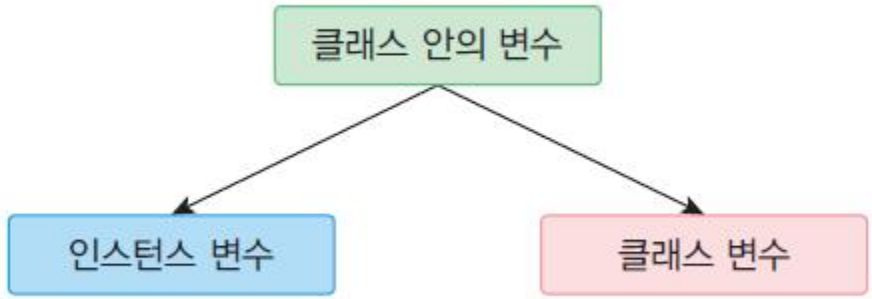
```
from tkinter import *
import time
import random

window = Tk()

canvas=Canvas(window, width=600,height=400)
canvas.pack()

class Ball():
    def __init__(self,color,size):
        self.id=canvas.create_oval(0, 0, size, size,fill=color)
        self.dx=random.randint(1,10)
        self.dy=random.randint(1,10)

    def move(self):
        canvas.move(self.id,self.dx,self.dy)
        x0, y0, x1, y1 = canvas.coords(self.id)
        if y1 > canvas.winfo_height() or y0 < 0: # 공이 위쪽이나 아래쪽으로 벗어나으면
            self.dy = -self.dy                                  # dy의 부호를 반전시킨다.
        if x1 > canvas.winfo_width() or x0 < 0:    # 공이 왼쪽이나 오른쪽으로 벗어나으면
            self.dx = -self.dx                                  # dx의 부호를 반전시킨다.
```

Sol.

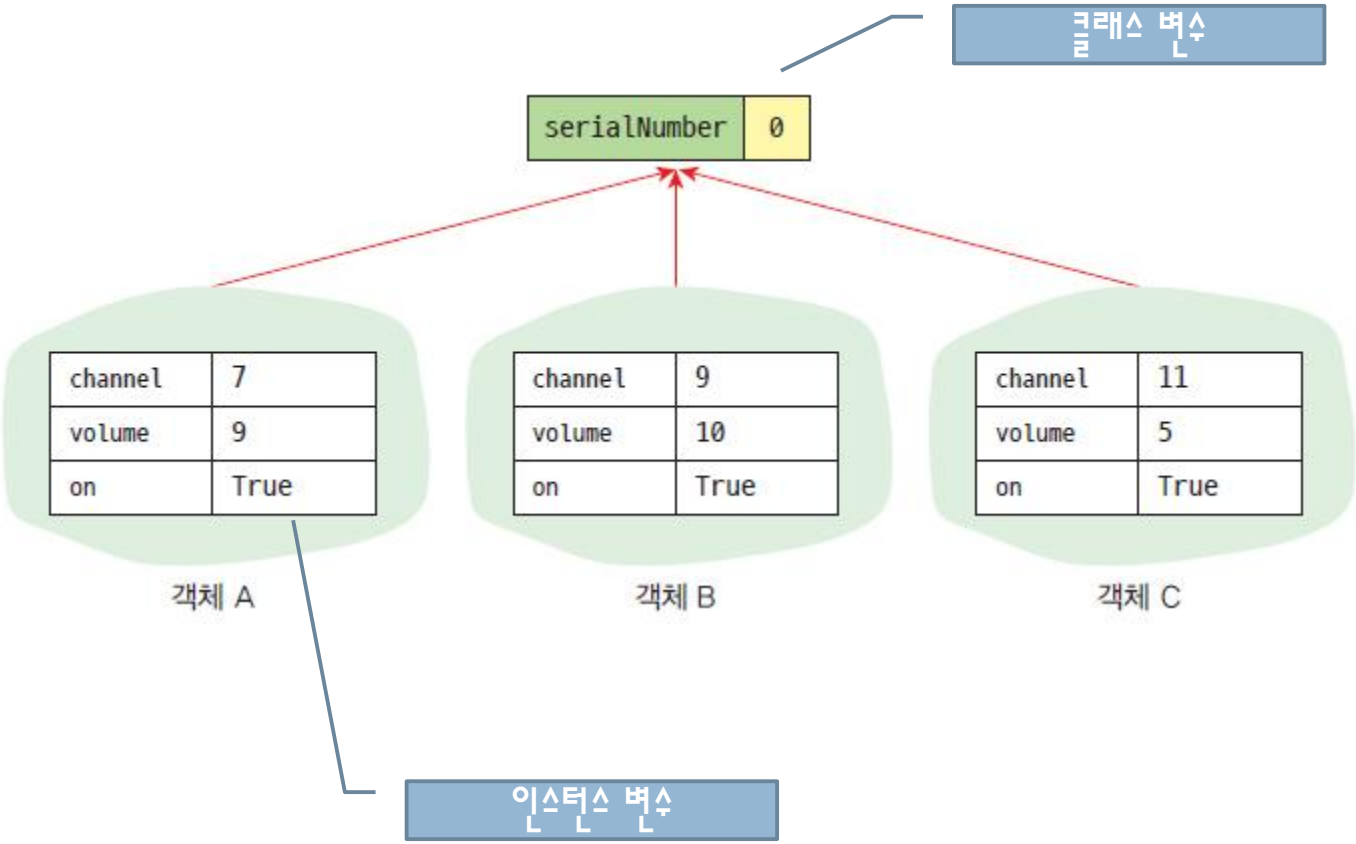
```
colors = ["red", "orange", "yellow", "green", "blue", "indigo", "violet"]
ballList = []
for i in range(30):
    ballList.append(Ball(random.choice(colors), random.randint(1, 60)))

while True:
    for i in range(30):
        ballList[i].move()
    window.update()
    time.sleep(0.05)

window.mainloop()
```

클래스 변수

- 모든 객체를 통틀어서 하나만 생성되고 모든 객체가 이것을 공유하는 변수를 클래스 멤버(class member)라고 한다.



인스턴스 변수 vs 클래스 변수



클래스 변수

```
# 텔레비전을 클래스로 정의한다.
class Television:
    serialNumber = 0                    # 이것이 클래스 변수이다.

    def __init__(self, channel, volume, on):
        self.channel = channel
        self.volume = volume
        self.on = on
        Television.serialNumber += 1            # 클래스 변수를 하나 증가한다.
                    # 클래스 변수의 값을 객체의 시리얼 번호로 한다.
        self.number = Television.serialNumber

    def show(self):
        print(self.channel, self.volume, self.on, self.number)

myTV = Television(11, 10, True);
myTV.show()
```

```
11 10 True 1
```

상수 정의

- 상수들은 흔히 클래스 변수로 정의된다.

```
class Monster :
   # 상수 값 정의
   WEAK = 0
   NORMAL = 10
   STRONG = 20
   VERY STRONG = 30

   def __init__(self) :
      self._health = Monster.NORMAL

   def eat(self) :
      self._health = Monster.STRONG

   def attack(self) :
      self._health = Monster.WEAK
```

중간점검

1. 인스턴스 변수와 클래스 변수의 차이점은 무엇인가?
2. 파이썬에서 클래스 변수를 생성하려면 어떻게 하면 되는가?

특수 메소드

- 파이썬에는 연산자(+, -, *, /)에 관련된 **특수 메소드(special method)**가 있다.
- 이들 메소드는 우리가 객체에 대하여 +, -, *, /와 같은 연산을 적용하면 자동으로 호출된다.

```
class Circle:
        ...
        def __eq__(self, other):
                return self.radius == other.radius
c1 = Circle(10)
c2 = Circle(10)
if c1 == c2:
        print("원의 반지름은 동일합니다. ")
```

특수 메소드

연산자	메소드	설명
x + y	__add__(self, y)	덧셈
x - y	__sub__(self, y)	뺄셈
x * y	__mul__(self, y)	곱셈
x / y	__truediv__(self, y)	실수나눗셈
x // y	__floordiv__(self, y)	정수나눗셈
x % y	__mod__(self, y)	나머지
divmod(x, y)	__divmod__(self, y)	실수나눗셈과 나머지
x ** y	__pow__(self, y)	지수
x << y	__lshift__(self, y)	왼쪽 비트 이동
x >> y	__rshift__(self, y)	오른쪽 비트 이동
x <= y	__le__(self, y)	less than or equal(작거나 같다)
x < y	__lt__(self, y)	less than(작다)
x >= y	__ge__(self, y)	greater than or equal(크거나 같다)
x > y	__gt__(self, y)	greater than(크다)
x == y	__eq__(self, y)	같다
x != y	__neq__(self, y)	같지않다

__str__() 메소드

```
class Counter:
    def __init__(self) :
        self.count = 0
    def increment(self):
        self.count += 1
    def __str__(self):
        msg = "카운트값:"+ str(self.count)
        return msg

a = Counter(100)
print(a)
```

```
카운트값: 100
```

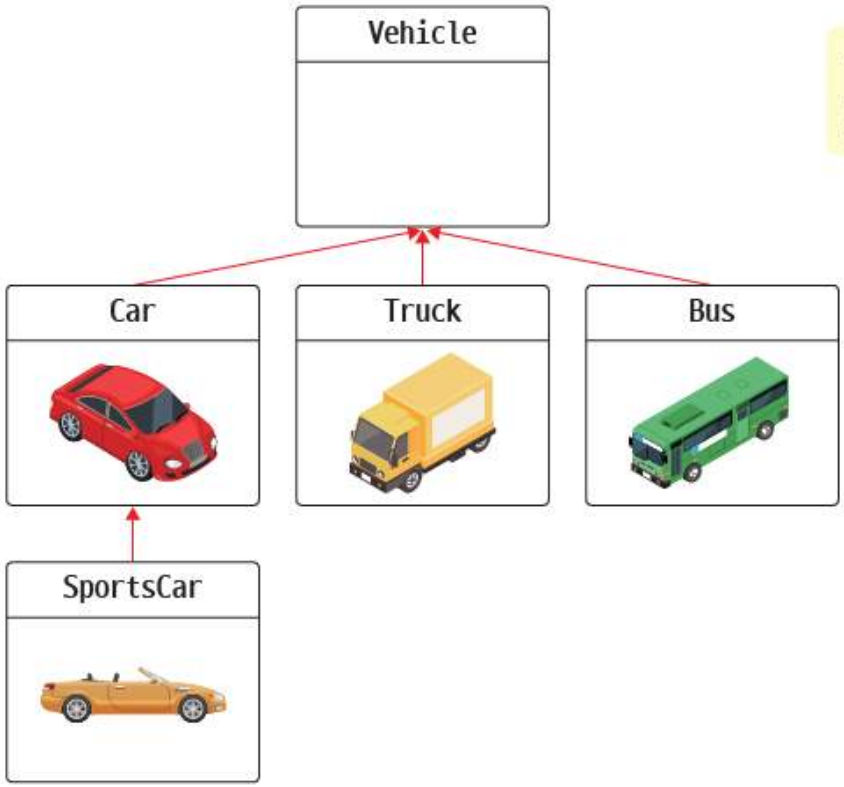
중간점검

1. __str__() 메소드는 어떤 경우에 호출되는가?
2. if obj1 == obj2 문장이 가능하게 하려면 클래스 안에 어떤 메소드를 정의해야 하는가?

상속

- 상속(inheritance)은 기존에 존 재하는 클래스로부터 코드와 데이터를 이어받고 자신이 필요한 기능을 추가하는 기 법이다.

상속의 예


Vehicle
Car
Truck
Bus
SportsCar


상속에서는 자식 클래스에서
부모 클래스로 화살표를
그립니다.

상속의 예

```
inheri.py
class Person():
    def __init__(self, name):
        self.name = name
    def getName(self):
        return self.name
    def isStudent(self):
        return False

class Student(Person):
   def __init__(self, name, gpa):
       super().__init__(name)
       self.gpa = gpa

   def isStudent(self):
        return True

obj1 = Person("Kim")
print(obj1.getName(), obj1.isStudent())

obj2 = Student("Park", 4.3)
print(obj2.getName(), obj2.isStudent())
```

```
Kim False
Park True
```

Lab: 특수 메소드

- 2차원 공간에서 벡터(vector)는 (a, b)와 같이 2개의 실수로 표현될 수 있다. 벡터 간에는 덧셈이나 뺄셈이 정의된다. 특수 메소드를 이용하여서 + 연산과 – 연산, str() 메소드를 구현해보자.

```
(0, 1) + (1, 0) = (1, 1)
```

Sol:

```
class Vector2D :
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Vector2D(self.x + other.x, self.y + other.y)

    def __sub__(self, other):
        return Vector2D(self.x - other.x, self.y - other.y)

    def __eq__(self, other):
        return self.x == other.x and self.y == other.y

    def __str__(self):
        return '(%g, %g)' % (self.x, self.y)

u = Vector2D(0,1)
v = Vector2D(1,0)
w = Vector2D(1,1)
a = u + v
print( a)
```

Mini Project: 주사위 클래스 만들기

- 먼저 데이터에 대하여 생각해보자. 가장 중요한 데이터는 주사위의 값이다. 주사위를 굴렸을 때 1에서 6까지의 숫자를 가질 수 있다. 만약 주사위를 화면에 그린다고 생각하면 주사위가 그려지는 위치와 주사위의 크기도 데이터로 가지고 있어야 한다. 일단 최소한도로 값, 위치, 크기만을 생각하자.
  - 주사위의 값(value)
  - 주사위의 위치(x, y)
  - 주사위의 크기(size)

이번 장에서 배운 것

- 클래스는 속성과 동작으로 이루어진다. 속성은 인스턴스 변수로 표현되고 동작은 메소드로 표현된다.
- 객체를 생성하려면 생성자 메소드를 호출한다. 생성자 메소드는 __init__() 이름의 메소드이다.
- 인스턴스 변수를 정의하려면 생성자 메소드 안에서 self.변수이름 과 같이 생성한다.

Q & A