

캡슐화와 상속

12주차

객체지향과 클래스

MEMO

캡슐화

- 캡슐로 객체를 감싸서 내부를 보호하고 볼 수 없게 하는 것
- 객체의 가장 본질적인 특징

- ⚙️ 중요한 정보들은 비공개로 둬
- ⚙️ 클래스 안의 메서드에서만 접근 가능



MEMO



- ◆ 파이썬의 기본 접근 권한
- ◆ 누구에게나 접근 가능

⚙️ __를 속성 앞에 붙여서 비공개로 사용

- ◆ 클래스 바깥쪽에서는 접근할 수 없음
- ◆ 클래스 안에서만 접근 가능

MEMO

```
class cellphone:
    def __init__(self, model, factory, price):
        self.model = model
        self.factory = factory
        self.__price = price

    def info(self):
        print("모델은", self.model, "입니다")
        print(self.factory, "에서 만들었습니다")
        print("가격은", self.__price, "입니다")

S1 = cellphone("M1", "Samsung", 1000)
s1.__price = 12000
s1.price = 14000
s1.info()
```


모델은 M1입니다
Samsung에서 만들었습니다
가격은 1000 입니다

```
class cellphone:
    def __init__(self, model, factory, price):
        self.model = model
        self.factory = factory
        self.__price = price

    def info(self):
        print("모델은", self.model, "입니다")
        print(self.factory, "에서 만들었습니다")
        print("가격은", self.__price, "입니다")

    def setprice (self, price):
        self.__price = price

s1 = cellphone("M1", "Apple", 1000)
s1.setprice(1200)
s1.info()
```



```
모델은 M1입니다
Samsung에서 만들었습니다.
가격은 1200 입니다
```

- ⚙ 부모클래스의 모든 속성(데이터, 메서드)을 물려줄 수 있음
- ⚙ 각 클래스마다 특화된 메서드와 멤버변수를 정의
 - ◆ 중복코드작성을 방지
 - ◆ 유지보수가 용이

MEMO

추상화

- 여러 클래스들의 공통된 특징 등을 추출해 기본 클래스로 작성하는 것
- 추상화의 결과물이 부모 클래스임

issubclass

상속관계에 있는 두 클래스의 관계를
알 수 있는 함수

MEMO

⚙ SaveCellPhone 클래스 정의

```
class cellphone :  
    def __init__(self, model, factory, color) :  
        self.model = model  
        self.factory = factory  
        self.color = color  
  
    def info(self) :  
        print("모델은", self.model, "입니다")  
        print(self.factory, "에서 만들었습니다")  
        print("색깔은", self.color, "입니다")  
  
class savecellphone :  
    def __init__(self, model, factory, color, store) :  
        self.model = model  
        self.factory = factory  
        self.color = color  
        self.store = store  
  
    def info(self) :  
        print("모델은", self.model, "입니다")  
        print(self.factory, "에서 만들었습니다")  
        print("색깔은", self.color, "입니다")  
        print("판매처는", self.store, "입니다")  
  
s1 = cellphone("S1", "Sample", "pink")  
s1.info()  
s2 = savecellphone ("S2", "Apple", "black", "우체국")  
s2.info()  
print(issubclass(savecellphone, cellphone))
```

모델은 S1 입니다
Sample 에서 만들었습니다
색깔은 pink 입니다
모델은 S2 입니다
Apple 에서 만들었습니다
색깔은 black 입니다
판매처는 우체국 입니다
False

⚙ SaveCellPhone 클래스 정의

```
class cellphone :
```

```
    def __init__(self, model, factory, color) :  
        self.model = model  
        self.factory = factory  
        self.color = color
```

부모클래스

```
    def info(self) :  
        print("모델은", self.model, "입니다")  
        print(self.factory, "에서 만들었습니다")  
        print("색깔은", self.color, "입니다")
```

```
class savecellphone :
```

```
    def __init__(self, model, factory, color, store) :  
        self.model = model  
        self.factory = factory  
        self.color = color  
        self.store = store
```

부모클래스 상속

```
    def info(self) :  
        print("모델은", self.model, "입니다")  
        print(self.factory, "에서 만들었습니다")  
        print("색깔은", self.color, "입니다")  
        print("판매처는", self.store, "입니다")
```

```
s1 = cellphone("S1", "Sample", "pink")  
s1.info()  
s2 = savecellphone ("S2", "Apple", "black", "우체국")  
s2.info()  
print(issubclass(savecellphone, cellphone))
```

⚙ SaveCellPhone 클래스 정의

```
class savecellphone(cellphone) :  
    def __init__(self, model, factory, color, store) :  
        cellphone.__init__(self, model, factory, color)  
        self.store = store  
  
    def info(self):  
        cellPhone.info(self)  
        print("판매처는", self.store, "입니다.")
```

다중 상속은 콤마(,)로 구분

⚙ SaveCellPhone 클래스 정의

```
class savecellphone(cellphone) :  
    def __init__(self, model, factory, color, store):  
        cellphone.__init__(self, model, factory, color)  
        self.store = store  
  
    def info(self):  
        cellphone.info(self)  
        print("판매처는", self.store, "입니다.")  
  
s1 = cellphone("S1", "Samsung", "Pink")  
s1.info()  
s2 = savecellphone("S2", "LG", "White", "우체국")  
s2.info()  
print(issubclass(savecellphone, cellphone))
```

```
모델은 S1입니다  
Samsung에서 만들었습니다.  
색깔은 Pink 입니다.  
모델은 S2입니다  
LG에서 만들었습니다.  
색깔은 White 입니다.  
판매처는 우체국입니다.  
True
```

상속관계가 있기 때문에 True값이 나옴

1

객체(object)

- 우리 주변의 모든 것들이 객체임
- 자신만의 고유한 특성(property)와
고유의 행동(behavior)를 가지고 있음

2

속성(property)

- 객체(object)가 가지는 자신만의 고유한
특성(property)

MEMO

3

메서드(method)

- 객체(object)가 가지는 행동(behavior)

4

클래스(class)

- 객체를 만들어내기 위한 설계도 또는 틀

MEMO

5

인스턴스(instance)

- 클래스를 통해서 만들어진 객체
- 보통 객체(Object)와 분리하기 위해서 인스턴스라는 용어를 사용

```
class 클래스명 :  
    속성(property)정의  
    행동(method)정의
```

MEMO

6

클래스 정의

- 클래스 객체가 생성됨
- 독립적인 이름공간이 생성됨

7

인스턴스 객체

- 클래스와 동일하게 생성됨
- 독립적인 이름 공간이 생성됨

MEMO

8

인스턴스 객체 변경

- 인스턴스 객체의 이름공간에 값이 변경되지 않으면 클래스 객체의 멤버변수와 메서드를 공유함
- 클래스 및 인스턴스 객체에 동적으로 멤버 변수를 추가 가능
- `self()`를 통해서 멤버변수에 접근

MEMO

9

캡슐화(정보은닉)

- 객체의 내부 속성이나 메서드를 외부에서 접근하지 못하도록 함으로 객체를 보호

10

상속

- 객체의 공통된 정보를 추출(추상화)하여 코드의 재사용성을 높임
- 유지보수 용이

MEMO

⚙️ 다음의 조건을 만족하는 책 클래스(class Book)를 작성하는 프로그램

◆ 멤버변수

- 책제목을 나타내는 title
- 책의 저자를 나타내는 author
- 책의 가격을 나타내는 price

◆ 메서드

- 책의 정보를 출력 – info()
- 책의 가격을 인하 – downPrice(price)

MEMO

⚙️ 다음의 조건을 만족하는 책 클래스(class Book)를 작성하는 프로그램

... 결과물 ...

```
책의 제목은 칼의 노래 입니다  
책의 작가는 김훈 입니다  
책의 가격은 12000 입니다  
책의 가격이 10000 로 인하되었습니다
```

MEMO

다음 시간에는

13주차. 모듈과 패키지

MEMO