

❖ 학습 내용

- 객체와 클래스
- 상속

❖ 객체와 클래스

❖ 객체

◆ 정의

- 객체(Object) = 속성(Attribute) + 기능(Method)

◆ 속성 - 사물의 특징 (변수)

- 예) 자동차의 속성. : 바디의 색, 바퀴의 크기, 엔진의 배기량

◆ 기능 - 특징적인 동작 또는 기능 (함수)

- 예) 자동차의 기능. : 전진, 후진, 좌회전, 우회전

❖ 객 체

- “18인치의 바퀴를 가진 2,000cc의 빨간 차는 전진, 후진, 좌회전, 우회전의 기능이 있다.”

→ ex 1]

```
class Car:
    def __init__(self):
        self.color = 0xFF0000 # 바디의 색
        self.wheel_size = 16 # 바퀴의 크기
        self.displacement = 2000 # 엔진 배기량

    def forward(self): # 전진
        pass

    def backward(self): # 후진
        pass

    def turn_left(self): # 좌회전
        pass

    def turn_right(self): # 우회전
        pass
```

Car 클래스의 정의 시작

Car 클래스 안에 차의 색, 바퀴 크기, 배기량을 나타내는 변수를 정의합니다.

Car 클래스 안에 전진, 후진, 좌회전, 우회전 함수를 정의합니다.

❖ 객 체

- 전 페이지의 Car 클래스는 자료형으로 사용된다.
- 객체 대신 인스턴스(Instance)라는 용어를 사용하기도 함.
 - 클래스가 설계도, 객체는 그 설계를 바탕으로 실체화한 것이라는 뜻에서 유래한 용어
 - 객체뿐 아니라 변수도 인스턴스라고 부름.
자료형을 메모리에 실체화한 것이 변수이기 때문임.

→ ex 1]

```
num = 123
```

자료형:int, 변수: num

```
my_car = Car()
```

자료형: Car 클래스, 객체: my_car

❖ 클래스(Class)

◆ 클래스의 정의

- 객체지향언어의 개념
- 객체를 만드는 틀과 같다.

◆ 클래스 정의 형식

클래스 정의 시 사용하는 키워드: class

```
class 클래스이름[ (상속받을클래스이름) ]:  
    클래스내용
```

❖ 클래스(Class)

→ ex 1]

```
class temp:
    a = None

    def __init__(self, dosi): # 객체가 만들어질 때 반드시 실행되는 메서드
        self.a = dosi

    def driving(self, dosi): # 메서드이며 반드시 첫 번째 인자에 self를 넣어야 한다
        print("%s로 운전 중입니다" % (dosi))

tory = temp("solid")
print(tory.a)
tory.driving("서울")
```

solid
서울로 운전 중입니다

→ 설명 1]

- `__init__()` : 객체가 만들어질 때 반드시 실행되는 메서드
- `driving()` : 메서드이며 반드시 첫 번째 인자에 `self`를 넣어야 한다
- `self` : 객체 자신을 가리키는 예약어

❖ 상속

❖ 상속

◆ 정의

- 기존 class를 그대로 물려받아 업그레이드시켜 사용하는 방법

◆ 상속 형식

```
class 클래스이름(상속받을클래스이름):  
    클래스내용
```

❖ 클래스(Class)

→ Car]

```
class Car:
    kind = "Sedan"
    countOfDoors = 5

    def __init__(self, name1):
        self.name = name1

    def sethorsepower(self, name1):
        self.horsepower = name1

    def driving(self, name1):
        print("%s로 운전 중입니다" % (name1))
```

→ Car_nextGeneration]

```
class Car_nextGeneration(Car):
    #Car클래스를 상속
    autoPilot = True
    #새롭게 추가된 속성
    countOfAirbag = 10; #새롭게 추가된 속성

    def setwheelSize(self, name1):
        #새롭게 추가된 setter
        self.wheelSize = name1

    def driving(self, name1):
        #메서드 오버라이딩
        print("%s(으)로 자동 운전 중입니다" % (name1))

    #새롭게 추가된 메서드
    def turnning(self, num):
        self.turnninghorsepower = self.horsepower + num
        return self.turnninghorsepower
```

→ ex 1]

```
toch = Car("touch")
toch.horsepower = 120
toch.driving("서울")
```

서울로 운전 중입니다

❖ 클래스(Class)

→ ex 2]

```
Mbbang = Car_nextGeneration("Mbbang")  
Mbbang.horsepower = 300  
Mbbang.wheelSize = 20  
Mbbang.turnning(200)  
Mbbang.driving("인천")
```

인천(으)로 자동 운전 중입니다

❖ 상속

◆ 정적메소드

- 인스턴스가 만들어지지 않아도 사용할 수 있는 메소드(2.x 버전에서...)
- 3.x 부터는 정적 메소드가 아니더라도 인스턴스 만들지 않아도 된다.
- 클래스와 연관성이 있는 함수를 클래스 안에 정의하여 클래스나 인스턴스를 통해서 호출

◆ 정적메소드 정의 형식

```
@staticmethod  
def 메소드이름([매개변수]):  
    메소드 실행내용
```

→ ex 1]

→ Car_nextGeneration]

```
class hello:  
    num = 10  
  
    @staticmethod  
    def calc(x):  
        return x + 10
```

```
print(hello.calc(10))
```

❖ 상속

◆ 클래스메소드

- 상속시켜준 클래스의 속성에 접근할 수 있는 메소드
- `cls` : '클래스'를 가리키는 예약어

◆ 클래스메소드 정의 형식

```
@classmethod  
def 메소드이름(cls [, 매개변수]):  
    메소드 실행내용
```

→ ex 1]

```
class hello:  
    t = '내가 상속해 줬어'  
  
    @classmethod  
    def calc(cls):  
        return cls.t  
  
class hello_2(hello):  
    t = '나는 상속 받았어'  
  
print(hello_2.calc())
```

나는 상속 받았어