DEEP LEARNING WITH PYTORCH

ABOUT CLASS

ABOUT CLASS

◆ CLASS란?

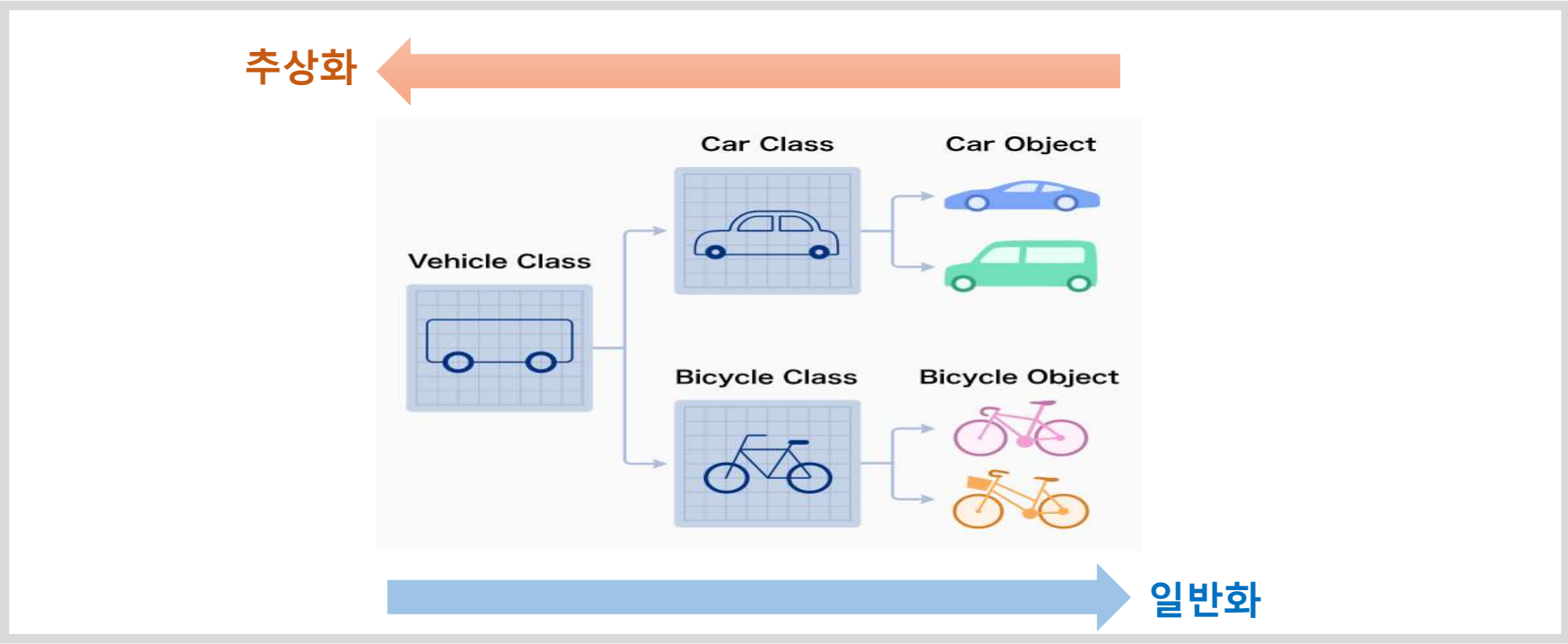
- 데이터에 대한 정의서
- 특징/외형/속성과 역할/행동/기능으로 데이터를 정의한 것
- 클래스(Class) 기반 프로그래밍 ➔ 객체지향프로그래밍(OOP Object Oriented Programming)

- **객체(Object)**
  → 메모리(힙 영역)에 클래스에 정의된 대로 데이터를 저장한 것

- **인스턴스(Instance)**
  → 메모리(힙 영역)에 저장된 데이터의 상세 클래스 정보 의미

ABOUT CLASS

◆ CLASS란?



ABOUT CLASS

◆ CLASS 구조

```
class 클래스이름:

    # 클래스 변수
    변수명='데이터'

    # 생성자 메서드
    def __init__(self, 매개변수1, 매개변수n):
        self.매개변수1=매개변수1

    # 인스턴스 메서드
    def 함수이름(self, 매개변수1, 매개변수n):
        print(self.매개변수1)
```

- 클래스 속성, 특징, 성질
- 생성된 인스턴스에 공유되는 변수

- 인스턴스 생성 시 초기화 되는 속성, 특징, 성질
- 인스턴스 마다 각자의 값을 저장

- 인스턴스의 행동/동작/기능
- 행동/동작/기능 코드 동일, 사용 데이터만 다름

ABOUT CLASS

◆ CLASS 구조

❖ self 키워드

- 메서드의 첫번째 매개변수
- **인스턴스/객체의 정보(주소)를 저장하고 있는 매개변수**
- 파이썬 시스템에서 자동으로 값을 입력하는 매개변수
- 메서드 코드를 공유하며 인스턴스 마다의 데이터를 사용할 수 있도록 하는 목적
- 메서드 호출 시에 self 매개변수는 무시

ABOUT CLASS

◆ CLASS 구조

❖ 다양한 클래스 형태

```
class Car:
    pass
```

- 속성 & 메서드 없는 클래스

```
class Car:

    def showInfo(self, name, price):
        print(f"{name} 자동차 ")
        print(f"{price}원입니다.")
```

- 메서드만 있는 클래스

ABOUT CLASS

◆ CLASS 구조

❖ 다양한 클래스 형태

```
class Car:

    def __init__(self, kind, user):
        self.kind = kind
        self.user=user

    def showInfo(self, name, price):
        print(f"{name} 자동차 ")
        print(f"{price}원입니다.")
```

- 속성 & 메서드 모두 존재하는 클래스

- 메모리(힙영역)에 데이터 저장하는 메서드
- **생성자 메서드(Constructor Method)**

ABOUT CLASS

◆ CLASS 구조

❖ 인스턴스/객체 생성

객체변수명 = 클래스명([인자, … , 인자])

```
class Car1:
    pass
```

생성

```
c1 = Car1()
c2 = Car1()
```

```
class Car2:
    def __init__(self, kind, user):
        self.kind = kind
        self.user=user
```

생성

```
c1 = Car('SUV',  1000)
c2 = Car('트럭', 55000)
```

ABOUT CLASS

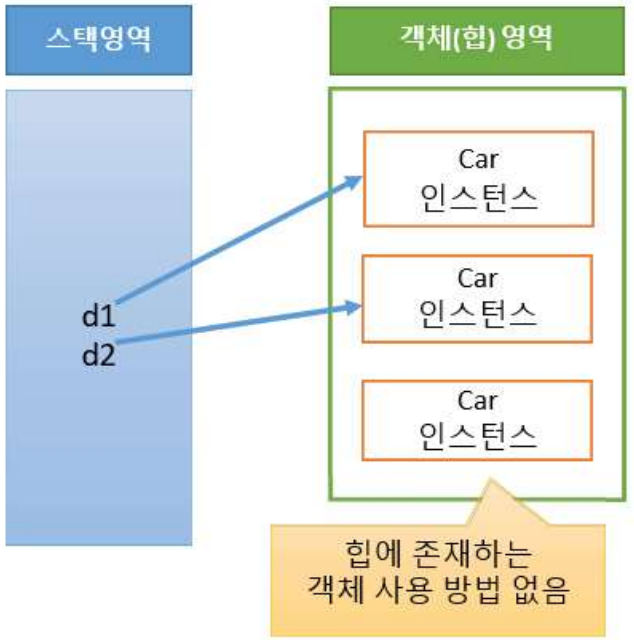
◆ CLASS 구조

❖ 인스턴스/객체 생성

```
class Car1:
    pass
```

생성

```
c1 = Car1()
c2 = Car1()
Car1()
```



ABOUT CLASS

◆ CLASS 구조

❖ 속성(Attribute)과 메서드(Method) 사용

- 속성값 읽기 : 객체변수명.속성명
- 속성값 변경 : 객체변수명.속성명 = 새로운 값

- 메서드 호출 : 객체변수명.메서드명( )

ABOUT CLASS

◆ CLASS 구조

❖ 예) 클래스 정의

```
class Car:
    # 클래스 변수
    brand = 'HYUDAI'

    # 생성자 : 인스턴스 생성 및 초기화 메서드
    def __init__(self, kind, user):
        self.kind = kind
        self.user=user

      # 인스턴스 메서드
      def showInfo(self, name):
          print(f"{name} 자동차 ")
          print(f"제조사:{Car.brand}")
          print(f"차주:{self.user}")
```

ABOUT CLASS

◆ CLASS 구조

❖ 예] Car 클래스 생성 및 속성/메서드 사용

```
# 인스턴스 생성
myCar = Car('SUV', '마징가')
yourCar = Car('Truck', '김')

# 인스턴스 메서드 호출
myCar.showInfo('펠리쉐이드')

# 속성 읽기
print(myCar.user)
print(yourCar.brand)

# 속성 변경
myCar.user='홍길동'
```

ABOUT CLASS

◆ CLASS 구조

❖ 인스턴스/객체의 속성(Attribute)과 메서드(Method) 확인

- 클래스명._ _dict_ _
- 인스턴스명._ _dict_ _

```
print(Car.__dict__)
```

{'__module__': '__main__', 'brand': 'HYUDAI', '__init__': <function Car.__init__ at 0x00000261910B8310>, 'showInfo': <function Car.showInfo at 0x00000261910B83A0>, '__dict__': <attribute '__dict__' of 'Car' objects>, '__weakref__': <attribute '__weakref__' of 'Car' objects>, '__doc__': None}

```
print(myCar.__dict__)
```

```
{'kind': 'SUV', 'user': '홍길동'}
```

ABOUT CLASS

◆ 스페셜/매직 변수 및 메서드

❖ **파이썬 시스템에서 미리 제공하는 변수와 메서드**

- **변 수 형태 : _ _변수명_ _**
  → 파이썬 시스템에서 변수값 설정

- **메서드 형태 : _ _매서드명_ _( )**
  → 파이썬 시스템에서 필요 시 자동 호출하는 메서드
  → 메서드 구현 코드는 개발자가 직접 작성해야 하는 경우도 있음

ABOUT CLASS

◆ 연산자 오버로딩(Overloading) 메서드

❖ 메서드명 동일하며 매개변수 개수, 타입, 순서가 다른 메서드 의미

❖ 연산자 오버로딩

- 객체에서 연산자를 클래스 목적에 맞게 기능 부여 사용
- 이름 앞뒤에 언더스코어(_ _) 두개 연속으로 붙은 함수

- 형태 : def _ _메서드명_ _(self, other)

매직함수명	연산자
def __add__(self, other)	+
def __sub__(self, other)	-
def __mul__(self, other)	*
def __truediv__(self, other)	/
def __floordiv__(self, other)	//
def __mod__(self, other)	%
def __pow__(self, other)	**

ABOUT CLASS

◆ 상속(Inheritance)

❖ 정의

- 클래스 확대 및 기존 **클래스 재사용 & 기능 확장**
- 부모 클래스가 가진 것 모두 자식 클래스에서 사용
- 부모 클래스로부터 상속받은 함수를 재정의 가능 ➔ 오버라이딩(Overriding)

- 형식 : class `class` 자식_클래스(부모_클래스)

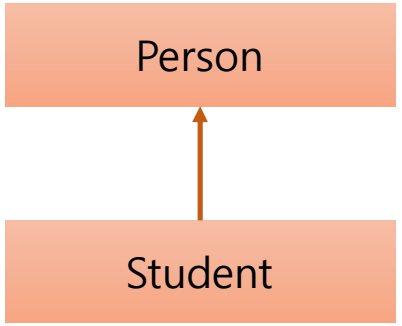
ABOUT CLASS

◆ 상속(Inheritance)

❖ 표현 : issubclass(자식클래스, 부모클래스)

```
class Person:
    pass


class Student(Person):
    pass
```



ABOUT CLASS

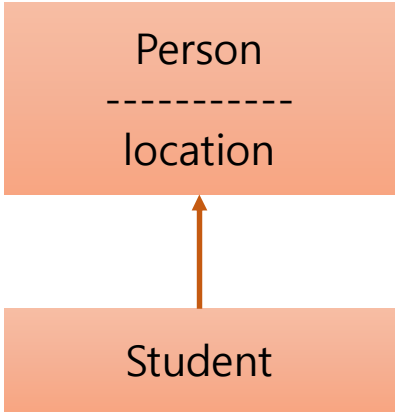
◆ 상속(Inheritance)

❖ 부모 클래스의 속성 사용

```
class Person:
    def __init__(self):
        self.location='Korea'



class Student(Person):
    def showInfo(self):
        print(f'Location : {self.location}')
```



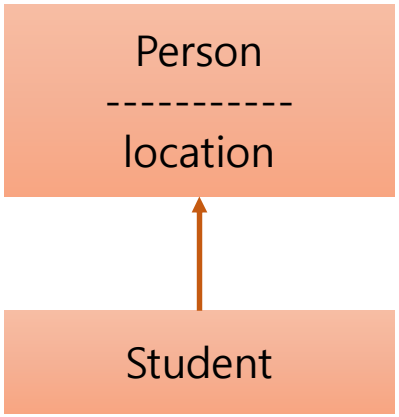
ABOUT CLASS

◆ 상속(Inheritance)

❖ **super() : 부모 클래스 생성**

```
class Person:
    def __init__(self, age, gender):
        self.age=age
        self.gender=gender

class Student(Person):
    def __init__(self, name, age, gender):
        super().__init__(age, gender)
        self.name=name
```



ABOUT CLASS

◆ 상속(Inheritance) - 오버라이딩

- ❖ 함수 구현 부분만 다시 재정의
- ❖ 상속관계에서 부모에서 상속 받은 메소드에 한정
- ❖ **필수 오버라이딩 메서드도 존재**

```
class Person:
    def __init__(self):
        self.location='Korea‘

    def showInfo(self):
        print(f'Location : {self.location}')

class Student(Person):

    def showInfo(self):
        print(f‘I am happy')
```

ABOUT CLASS

◆ Object 클래스

❖모든 클래스의 부모 클래스

❖자동으로 상속받게 되는 클래스

```
__class__                                   object
__str__(self)                               object
__annotations__                             object
__delattr__(self, name)                     object
__dir__(self)                               object
__eq__(self, o)                             object
__format__(self, format_spec)               object
__getattribute__(self, name)                object
__hash__(self)                              object
__init_subclass__(cls)                      object
__ne__(self, o)                             object
__new__(cls)                                object
```