Cf. super(), 메소드 오버라이딩

- **super()**
  - 자식 클래스에서 부모 클래스의 method를 호출할 때 사용
  - super().부모 클래스의 method명, 부모클래스 이름도 사용 가능(self추가)

```
class Person:
    def __init__(self, name, age, gender):
        self.Name = name
        self.Age = age
        self.Gender = gender

    def introduce(self):
        print("저의 이름은 " + self.Name + "이구요, 제 나이는 " + self.Age + "살 입니다."


class Employee(Person):
    def __init__(self, name, age, gender, salary, time):
        super().__init__(name, age, gender)
        #Person.__init__(self, name, age, gender)
        #도 가능
        self.salary=salary
        self.time=time

    def introduce(self): #메소드 오버라이딩(기존의 함수 기능을 사용하면서 기능추가)
        super().introduce() #기존의 부모 클래스의 함수 이용
        print("제 급여는 "+self.salary+"원 입니다.") #자식 클래스에서 기능 추가

    def worktime(self):  #새로운 메소드 생성
        print("1주 일하는 시간은" +self.time+"시간 입니다.")

Employee1=Employee("홍길동 ", "25", "남", "5000000", "52")
Employee1.introduce()
Employee1.worktime()
```

- **메소드 오버라이드**

```
class Person:
    def __init__(self, name, age, gender):
        self.Name = name
        self.Age = age
        self.Gender = gender

    def introduce(self):
        print("저의 이름은 " + self.Name + "이구요, 제 나이는 " + self.Age + "살 입니다."


class Employee(Person):
    def __init__(self, name, age, gender, salary, time):
        super().__init__(name, age, gender)
        #Person.__init__(self, name, age, gender)
        #도 가능
        self.salary=salary
        self.time=time

    def introduce(self): #메소드 오버라이딩(기존의 함수 기능을 사용하면서 기능추가)
        super().introduce() #기존의 부모 클래스의 함수 이용
        print("제 급여는 "+self.salary+"원 입니다.") #자식 클래스에서 기능 추가

    def worktime(self):  #새로운 메소드 생성
        print("1주 일하는 시간은" +self.time+"시간 입니다.")

Employee1=Employee("홍길동 ", "25", "남", "5000000", "52")
Employee1.introduce()
Employee1.worktime()
```

```
저의 이름은 홍길동 이구요, 제 나이는 25살 입니다.
제 급여는 5000000원 입니다.
1주 일하는 시간은52시간 입니다.
```