# Solution Review 2: Implement an Animal Class

This lesson provides a solution review for the `Implement an Animal Class` challenge.

**We'll cover the following**

- Solution
- Explanation

## Solution #

solution

```python
class Animal:
    def __init__(self, name, sound):
        self.name = name
        self.sound = sound

    def Animal_details(self):
        print("Name:", self.name)
        print("Sound:", self.sound)


class Dog(Animal):
    def __init__(self, name, sound, family):
        super().__init__(name, sound)
        self.family = family

    def Animal_details(self):
        super().Animal_details()
        print("Family:", self.family)


class Sheep(Animal):
    def __init__(self, name, sound, color):
        super().__init__(name, sound)
        self.color = color

    def Animal_details(self):
        super().Animal_details()
        print("Color:", self.color)


d = Dog("Pongo", "Woof Woof", "Husky")
```

## Explanation #

- We have implemented an `Animal` class which has `name` and `sound` *properties*, and a *method* `Animal_details()` which is overridden in its child classes.
- Then, we implemented the `Dog` and `Sheep` classes, which are inherited from the `Animal` class.
- `Dog` has an additional property, `family`, and the overridden method, `Animals_details()`. This method calls the parent method using the `super()` function and also prints the `family` property.
- `Sheep` has an additional property, `color`, and the overridden method, `Animals_details()`. This method calls the parent method using the `super()` function and also prints the `color` property.
- Created and initialized `Dog` and `Sheep` objects and printed their traits by calling their respective methods.

---

This is it for polymorphism. In the next chapter, we will learn about the different relationships between classes.