```
# When we create a class we can inherit all of the fields and methods from another class
# This is called inheritance
# The class that inherits is called the subclass and the class we inherit from is the super class
# This will be our super class
class Animal:
    def __init__(self, birth_type="Unknown", appearance="Unknown", blooded="Unknown"):
        self.__birth_type = birth_type
        self.__appearance = appearance
        self.__blooded = blooded

    # The getter method
    @property
    def birth_type(self):

    # When using getters and setters don't forget the __
        return self.__birth_type

    @birth_type.setter
    def birth_type(self, birth_type):
        self.__birth_type = birth_type

    @property
    def appearance(self):
        return self.__appearance

    @appearance.setter
    def appearance(self, appearance):
        self.__appearance = appearance

    @property
    def blooded(self):
        return self.__blooded

    @blooded.setter
    def blooded(self, blooded):
        self.__blooded = blooded

    # Can be used to cast our object as a string
    # type(self).__name__ returns the class name
    def __str__(self):
        return "A {} is {} it is {} it is " \
               "{}".format(type(self).__name__,
                                    self.birth_type,
                                    self.appearance,
                                    self.blooded)

# Create a Mammal class that inherits from Animal
# You can inherit from multiple classes by separating
# the classes with a comma in the parentheses
class Mammal(Animal):
    def __init__(self, birth_type="born alive",
                 appearance="hair or fur",
                 blooded="warm blooded",
                 nurse_young=True):
```

```
        # Call for the super class to initialize fields
        Animal.__init__(self, birth_type,
                        appearance,
                        blooded)

        self.__nurse_young = nurse_young

    # We can extend the subclasses
    @property
    def nurse_young(self):
        return self.__nurse_young

    @nurse_young.setter
    def nurse_young(self, nurse_young):
        self.__nurse_young = nurse_young

    # Overwrite __str__
    # You can use super() to refer to the superclass
    def __str__(self):
        return super().__str__() + " and it is {} they nurse " \
            "their young".format(self.nurse_young)


class Reptile(Animal):
    def __init__(self, birth_type="born in an egg or born alive",
                 appearance="dry scales",
                 blooded="cold blooded"):

        # Call for the super class to initialize fields
        Animal.__init__(self, birth_type,
                        appearance,
                        blooded)


def main():
    animal1 = Animal("born alive")

    print(animal1.birth_type)

    # Call __str__()
    print(animal1)
    print()

    mammal1 = Mammal()

    print(mammal1)

    print(mammal1.birth_type)
    print(mammal1.appearance)
    print(mammal1.blooded)
    print(mammal1.nurse_young)
    print()
```

```
    reptile1 = Reptile()

    print(reptile1.birth_type)
    print(reptile1.appearance)
    print(reptile1.blooded)

main()
```