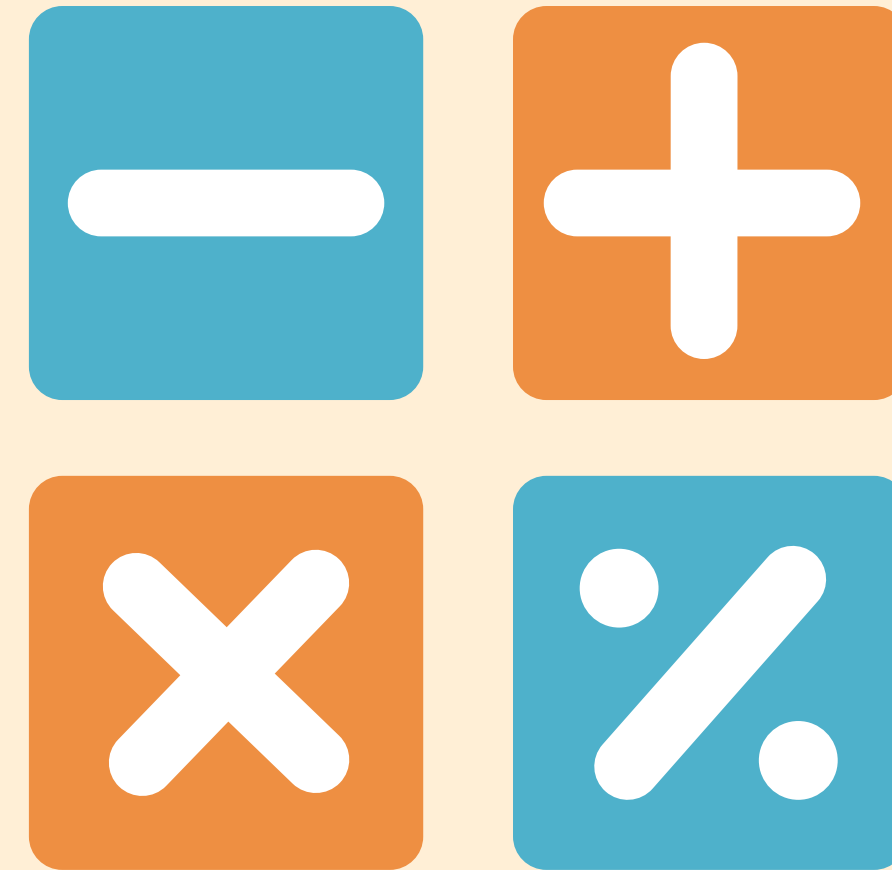


Multiple Class Inheritance



*Think about Inheritance but accepting more
than just one class!*

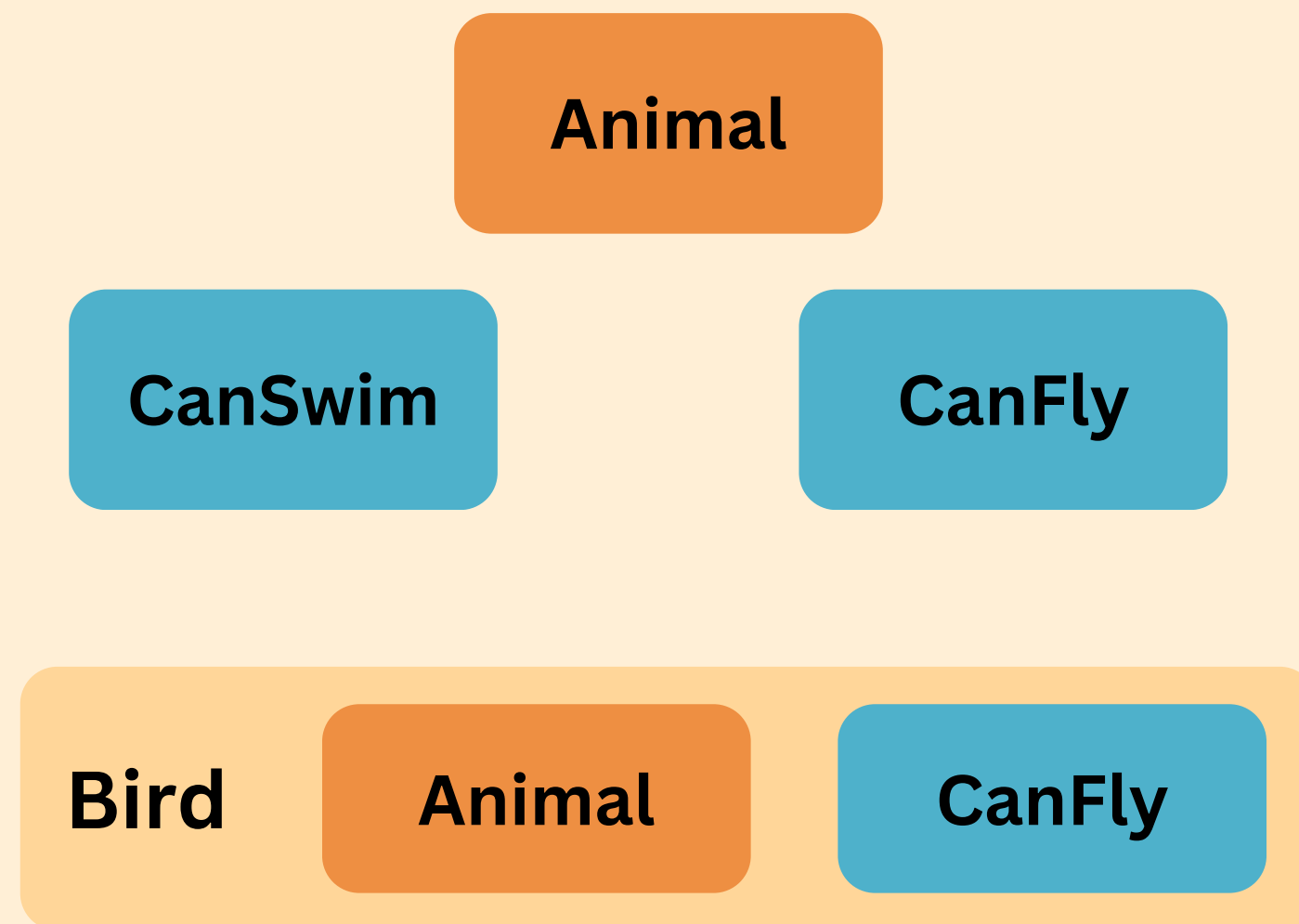
Inheriting Multiple Classes:

That's right! You can give a child class, multiple classes.

Example: The **Animal Class** is the "**Base class**" that provides the **basics for all the animals**.

Each animal can do something so we have two other classes, **CanFly** and **CanSwim**.

Finally we have an animal, **bird**. This object **will inherit both, Animal and CanFly**



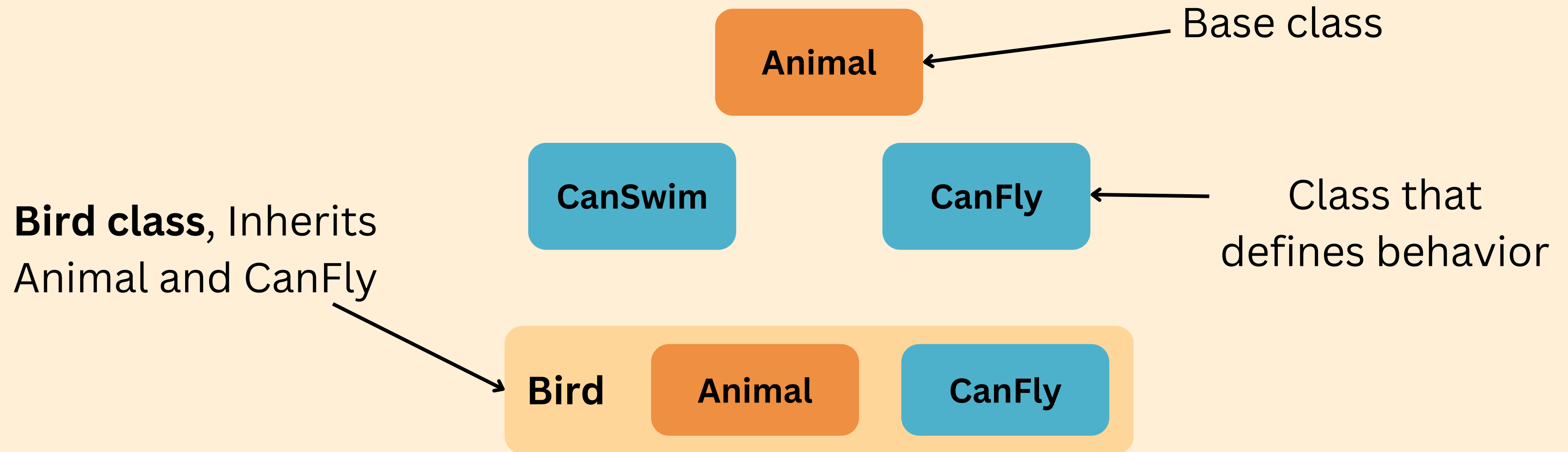
Inheriting Multiple Classes:

That's right! You can give a child class, multiple classes.

Example: The **Animal Class** is the "**Base class**" that provides the **basics for all the animals**.

Each animal can do something so we have two other classes, **CanFly** and **CanSwim**.

Finally we have an animal, **bird**. This object **will inherit both, Animal and CanFly**



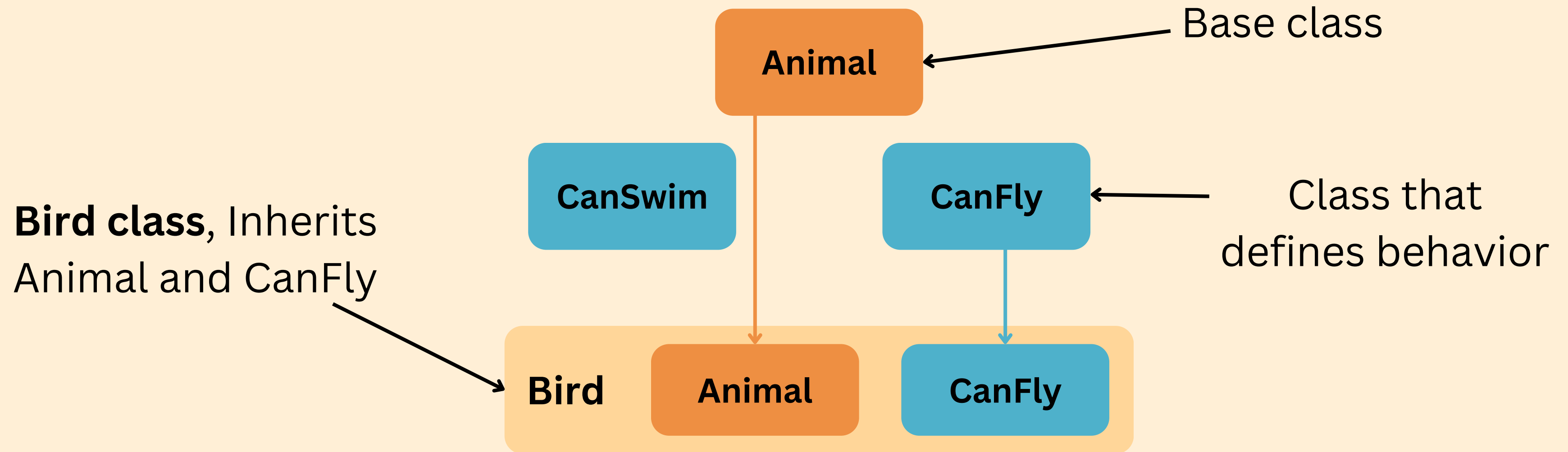
Inheriting Multiple Classes:

That's right! You can give a child class, multiple classes.

Example: The **Animal Class** is the "**Base class**" that provides the **basics for all the animals**.

Each animal can do something so we have two other classes, **CanFly** and **CanSwim**.

Finally we have an animal, **bird**. This object **will inherit both, Animal and CanFly**



Inheriting Multiple Classes:

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

    def eat(self):
        print(f"{self.name} is eating.")

class CanFly:
    def fly(self):
        print(f"{self.name} is flying.")

class Bird( Animal, CanFly ):
    def __init__(self, name, color ):
        super().__init__( name )
        self.color = color
    def sleep(self):
        print(f"The {self.color} bird is sleeping")

my_bird = Bird("Canary")
my_bird.eat()
my_bird.fly()
my_bird.sleep()
```

Our Base Class

Our Behavior Class

Our final/specific class

Creating an Object

Using the methods from
all 3 classes

Inheriting Multiple Classes:

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

Our Base Class

```
    def eat(self):
        print(f"{self.name} is eating.")
```

Our Behavior Class

```
class CanFly:
    def fly(self):
        print(f"{self.name} is flying.")
```

Our final/specific class

```
class Bird( Animal, CanFly ):
    def __init__(self, name, color ):
        super().__init__( name )
        self.color = color
    def sleep(self):
        print(f"The {self.color} bird is sleeping")
```

Creating an Object


```
my_bird = Bird("Canary")
my_bird.eat()
my_bird.fly()
my_bird.sleep()
```

Using the methods from
all 3 classes

Inheriting Multiple Classes:

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

    def eat(self):
        print(f"{self.name} is eating.")
```



```
class CanFly:
    def fly(self):
        print(f"{self.name} is flying.")
```

```
class Bird(Animal, CanFly):
    def __init__(self, name, color):
        super().__init__(name)
        self.color = color

    def sleep(self):
        print(f"The {self.color} bird is sleeping")
```

```
my_bird = Bird("Canary", "red")
my_bird.eat()
my_bird.fly()
my_bird.sleep()
```

Our Base Class

Our Behavior Class

Our final/specific class

Creating an Object

Using the methods from
all 3 classes

Because we are
using `super()` in
the Bird class,
we can **pass the
name to the
parent classes**

Inheriting Multiple Classes:

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

    def eat(self):
        print(f"{self.name} is eating.")
```

Our Base Class

```
class CanFly:
    def fly(self):
        print(f"{self.name} is flying.")
```

Our Behavior Class

```
class Bird(Animal, CanFly):
    def __init__(self, name, color):
        super().__init__(name)
        self.color = color

    def sleep(self):
        print(f"The {self.color} bird is sleeping")
```

Our final/specific class

```
my_bird = Bird("Canary", "red")
my_bird.eat()
my_bird.fly()
my_bird.sleep()
```

Creating an Object

Using the methods from
all 3 classes

Because we are
using `super()` in
the Bird class,
we can **pass the
name to the
parent classes**

Challenge #1 in VS code:

DO NOT GO BACK!

Try to recreate the code from previous slide!

Use it as an example to build your own

One Base Class - (Animal)

Two Behavior or Style Classes (CanFly, CanWalk)

One Child Class (final class)

Use **print()** to interact with each class

Use **super()** to **pass the arugument to each class**

