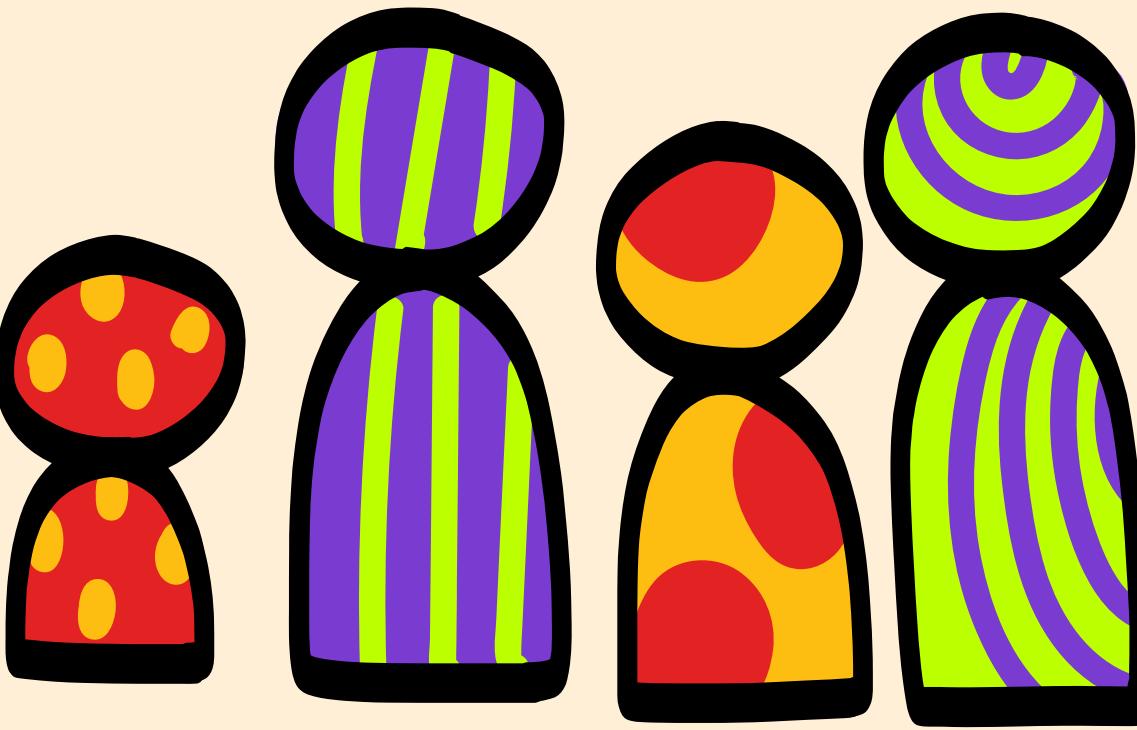


The Path to Class Inheritance



*Think of Inheritance, what do you think about?
Well it's probably the same for Python too!*

What we already know:

Computers

Macbook

Windows

Linux

Animals

Cats

Dogs

Birds

Tiger

Bear

Games

RPG

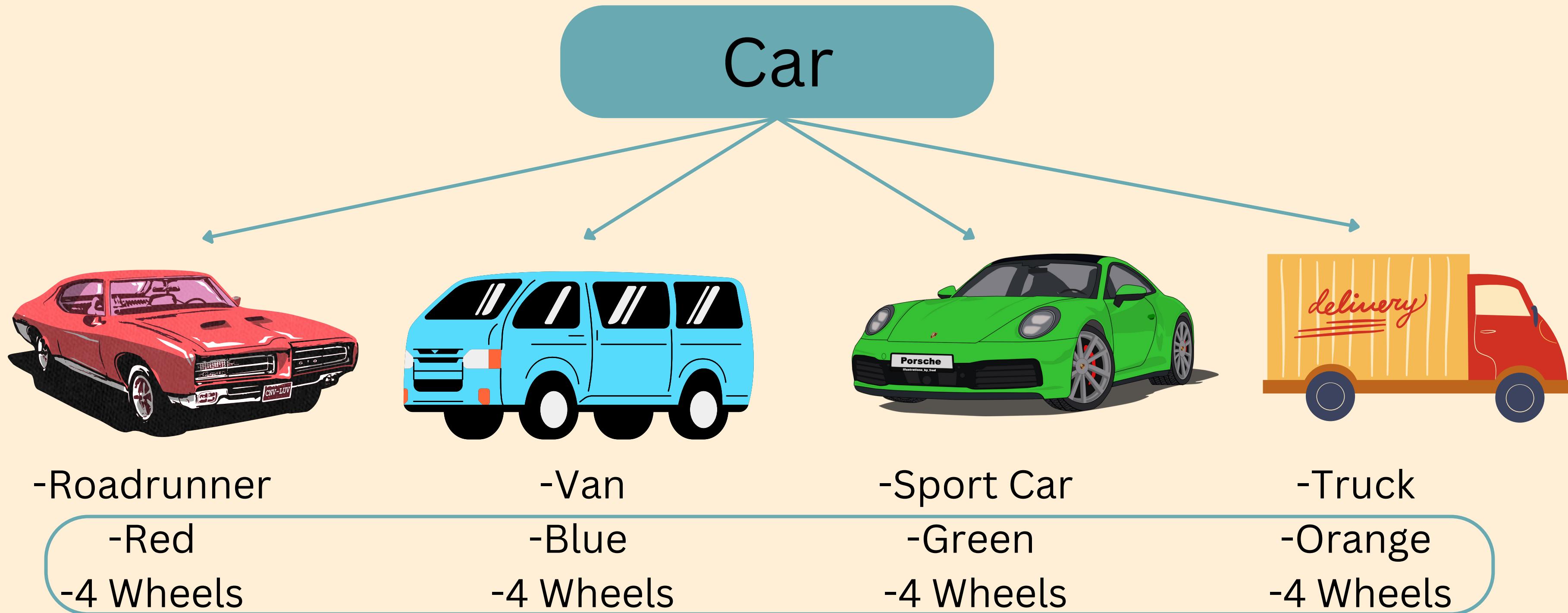
Racing

Single Player

Example: Computers -> **Computers is a Class (Family)**, while **MacBook, Windows & Linux are all Objects**. We could create a Class **for each** type of Computer as well, **taking key data from the parent**.

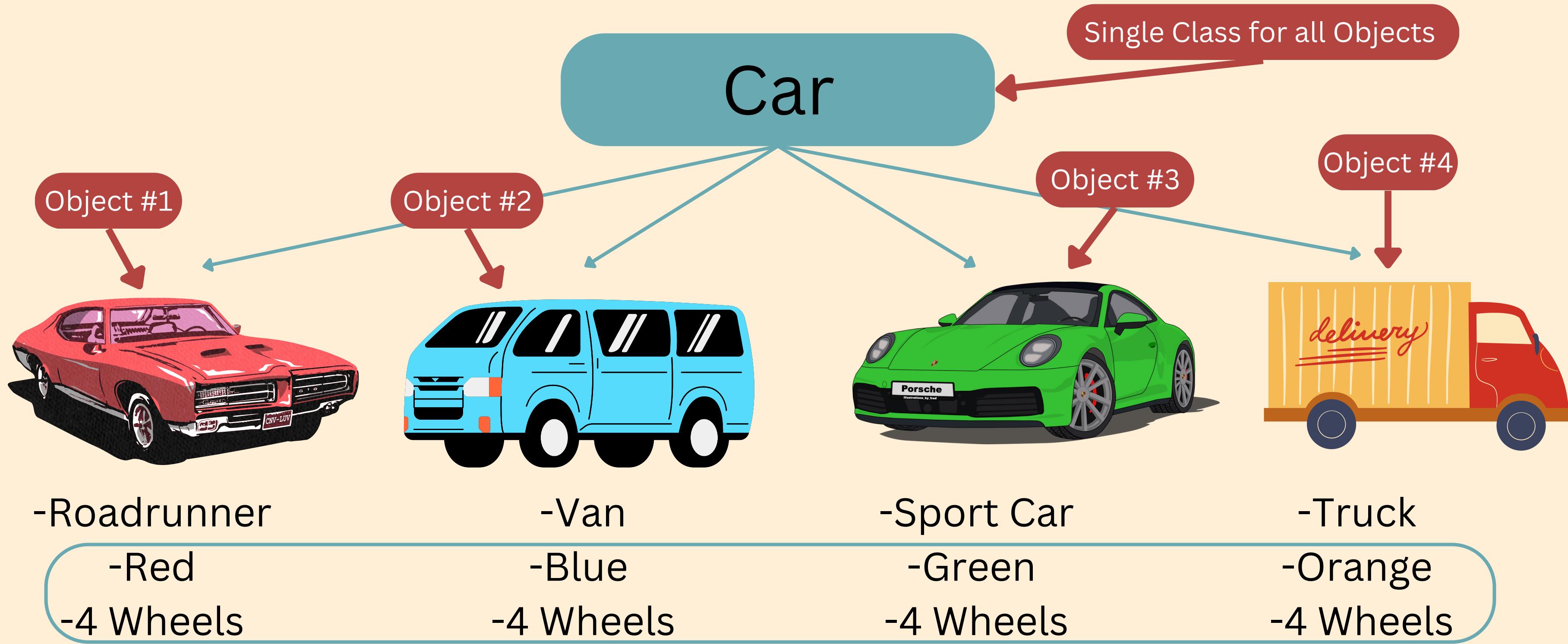
What we currently know:

Here is a **car class**. There are **4 objects**, each with **3 properties**.



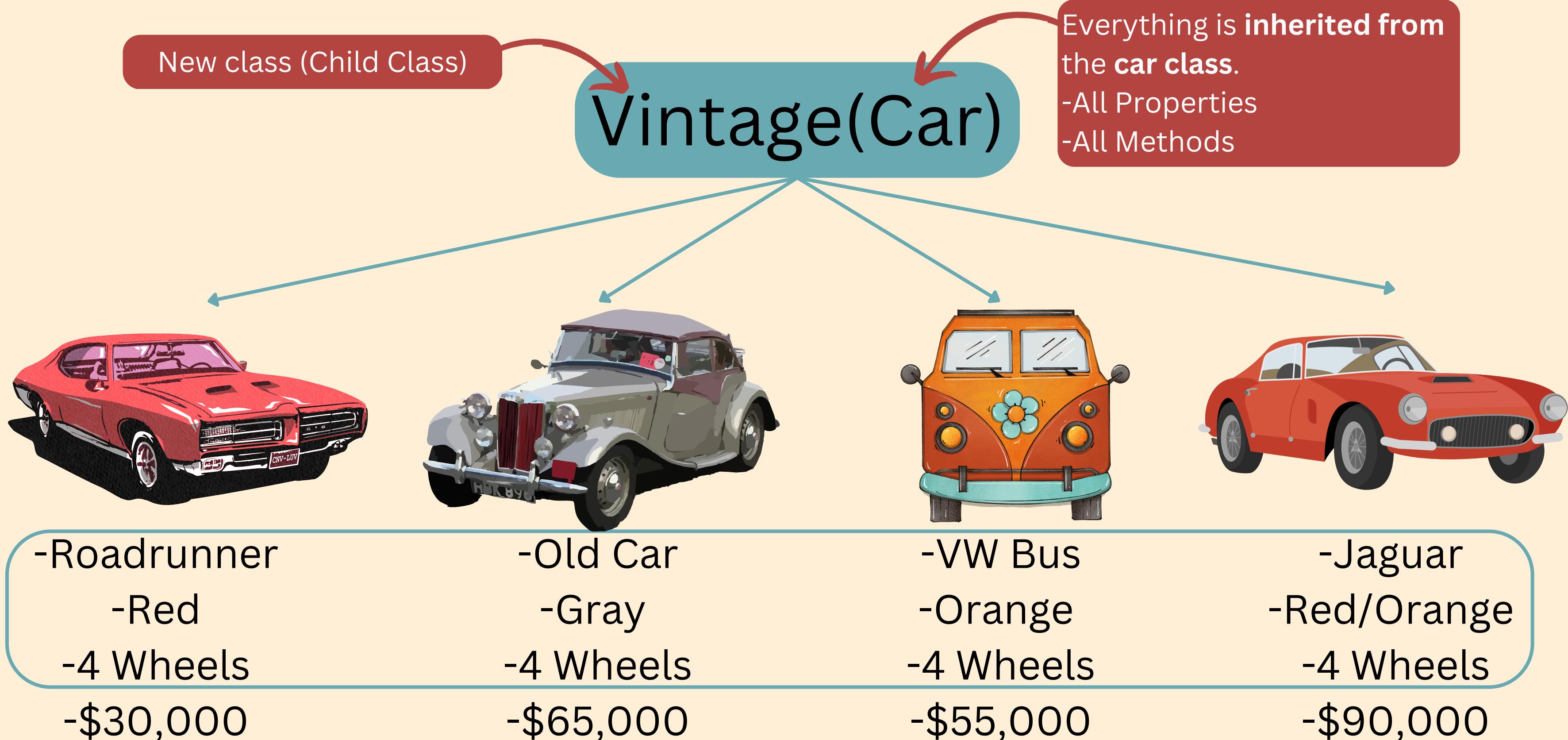
What we currently know:

Here is a **car class**. There are **4 objects**, each with **3 properties**.



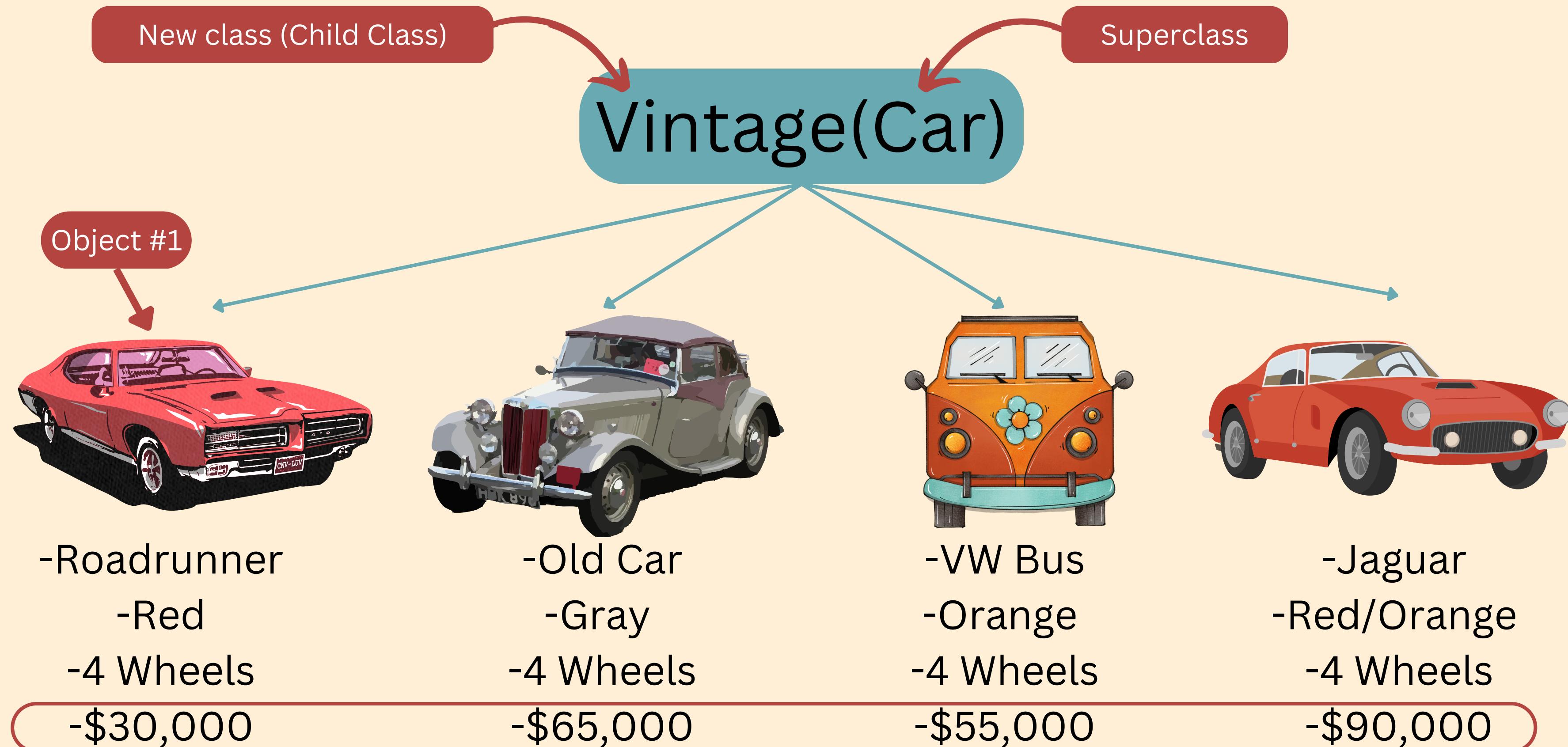
Creating a derived class (child class)

A new Class with the name Vintage. This is Inheriting the car class. This is an example of **Class Inheritance** in Python



Creating a derived class (child class)

This new class also has a **new property** that is only found here



Superclasses & Derived Classes:

All Audi's are Cars

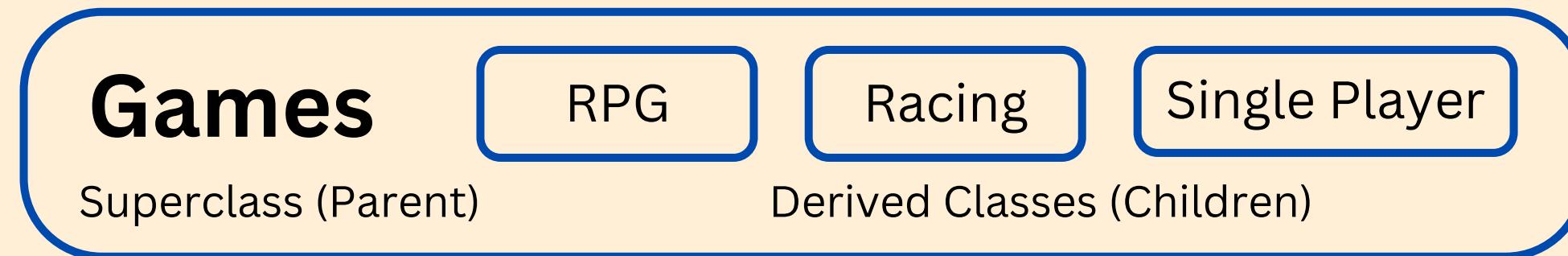
All sofas are furniture

All dogs are animals

All jeans are pants

Route	Inheritance Type	Note
Route One	The Child Class is given only given new methods, no new properties are given	We do not need a Constructor Method (<code>__init__</code>). We will use the Superclass constructor
Route Two	The Child Class is given both new properties and new methods	A new constructor is needed , and we must activate the superclass constructor as well

Superclass and Inheritance:



Example: Computers -> **Computers is a Class** (Parent), while **MacBook, Windows & Linux are all a sub-class of their own**. We can now create more specific objects using our derived classes (child class).

Inheritance - Route #1

Inheriting everything from a super class

Creating a child class ~ Routes #1:

Creating a Child Class that **only needs new methods** not properties

When we make an instance of a child class, the superclass constructor (`__init__`) will be **called and used in the child class**

```
class Main:
```

```
    def __init__(self, para1, para2):
```

```
        self.property1 = para1
```

```
        self.property2 = para2
```

```
    def method_one(self):
```

```
        #run this code
```

```
class Child_Class(Main):
```

```
    def method_one(self, parameter1):
```

```
        x = self.property1 + parameter1
```

```
        return x
```

```
    def method_two(self, parameter1):
```

```
        #run this code
```

Creating a child class ~ Routes #1:

1. When creating a Child class, **pass the Superclass in as a parameter** to the Child Class
2. Add any new methods

```
class Main:
```

```
    def __init__(self, para1, para2):
```

```
        self.property1 = para1
```

```
        self.property2 = para2
```

```
    def method_one(self):
```

```
#run this code
```

```
class Child_Class(Main):
```

```
    def method_one(self, parameter1):
```

```
        x = self.property1 + parameter1
```

```
        return x
```

```
    def method_two(self):
```

```
#run this code
```

How inheritance works in our code:

```
class Main:
```

```
    def __init__( self, name, age, location ):
```

```
        self.name = name
```

```
        self.age = int( age )
```

```
        self.location = location
```

```
    def user_info(self):
```

```
        print("Welcome," , self.name )
```

```
        print("You are:", self.age )
```

```
        print("You live in:", self.location )
```

```
class UserScore( Main ):
```

```
    def calc_score(self, number ):
```

```
        score = self.age * number
```

```
        return score
```

```
    def check_age(self):
```

```
        if self.age >= 70:
```

```
            return "Senior"
```

```
        elif self.age <= 17:
```

```
            return "Minor"
```

```
        else:
```

```
            return "Normal"
```

*We created a **child class called “UserScore”**. This will be a class which **uses the superclass properties**. We **inherit all the properties from our superclass**, which we can use throughout methods in our child class

How inheritance works in our code:

```
class Main:  
    def __init__( self, name, age, location ):  
        self.name = name  
        self.age = int( age )  
        self.location = location  
  
    def user_info(self):  
        print("Welcome," , self.name )  
        print("You are:", self.age )  
        print("You live in:", self.location )
```

```
class UserScore(Main):  
    def calc_score(self, number ):  
        score = self.age * number  
        return score  
  
    def check_age(self):  
        if self.age >= 70:  
            return "Senior"  
        elif self.age <= 17:  
            return "Minor"  
        else:  
            return "Normal"
```

*We created a **child class called “UserScore”**. This will be a class which **uses the superclass properties**. We **inherit all the properties from our superclass**, which we can use throughout methods in our child class

Inheritance - Route #2

Using the `super()` function to inherit
while creating!

Creating a child class ~ Routes #2:

1. When creating a Child class, pass the Superclass in as a parameter
2. Add old & new properties
3. Add new methods

```
class UserScore( Main ):  
    def __init__(self, name, age, location, score):  
        super().__init__( name, age, location)  
        self.score = int(score)
```

```
def checkAvg(self, list1 ):  
    x = self.score / len(list1) * 100  
    return x
```

The constructor in a child class **takes the properties of the superclass** and any **new properties we create**

super() allows us to **inherit all the properties and methods** from the superclass (parent class)

Creating a child class ~ Routes #2:

1. When creating a Child class, pass the Superclass in as a parameter
2. Add any new methods

```
class Child_Class(Main):  
    def __init__(self, x , y , z , name , age):  
        super().__init__(x , y , z)  
        self.name = name  
        self.age = int(age)  
  
    def method_two(self, parameter1):  
        #run this code
```

The constructor in a child class **takes the properties of the superclass** and any **new properties we create**

super() allows us to **inherit all the properties and methods** from the superclass (parent class)

Creating a child class ~ Routes #2:

1. When creating a Child class, pass the Superclass in as a parameter

2. Add any new methods

```
class Child_Class(Main):  
    def __init__(self, x , y , z, name , age):  
        super().__init__(x , y , z)  
        self.name = name  
        self.age = int(age)
```

```
def method_two(self, parameter1):  
    #run this code
```

The constructor in a child class **takes the properties of the superclass** and any **new properties we create**

super() allows us to **inherit all the properties and methods** from the superclass (parent class)

When we create an object of the child class, the superclass constructor (`__init__`) is **automatically called** and used.

Creating a child class ~ Routes #2:

```
class Main:  
    def __init__( self, name, age, location ):  
        self.name = name  
        self.age = int( age )  
        self.location = location  
  
    def user_info(self):  
        print("Welcome," , self.name )  
        print("You are:", self.age )  
        print("You live in:", self.location )  
  
class UserScore( Main ):  
    def __init__(self, name, age, location, score):  
        super().__init__( name, age, location)  
        self.score = int( score )  
  
    def checkAvg(self, list1 ):  
        x = self.score / len(list1) * 100  
        print("Results:", x)
```

We are **initializing the properties from the Superclass** by using the **super()** function as well as creating **1 new property** for the child class.

Creating a child class ~ Routes #2:

```
class Main:  
    def __init__(self, name, age, location):  
        self.name = name  
        self.age = int(age)  
        self.location = location  
  
    def user_info(self):  
        print("Welcome,", self.name)  
        print("You are:", self.age)  
        print("You live in:", self.location)
```

```
class UserScore(Main):  
    def __init__(self, name, age, location, score):  
        super().__init__(name, age, location)  
        self.score = int(score)
```

```
def checkAvg(self, list1):  
    x = self.score / len(list1) * 100  
    print("Results:", x)
```

```
test_list = [4, 5, 5, 4, 3, 5, 5, 4]
```

```
user = UserScore("Josh", 25, "HCMC", 5)  
user.checkAvg(test_list)
```

*Never use `self` outside of the class

Output in Terminal

Results: 5