Inheritance

We can define a class in such a way the class having its own features, also the class is able to inherit the features from some existing class. This is called Inheritance. Also the classes involve which provide inheritance is called Parent/Base/Super class and the class which is inherit the features is called Child/Derive/Sub class.

Syntax for Inheritance

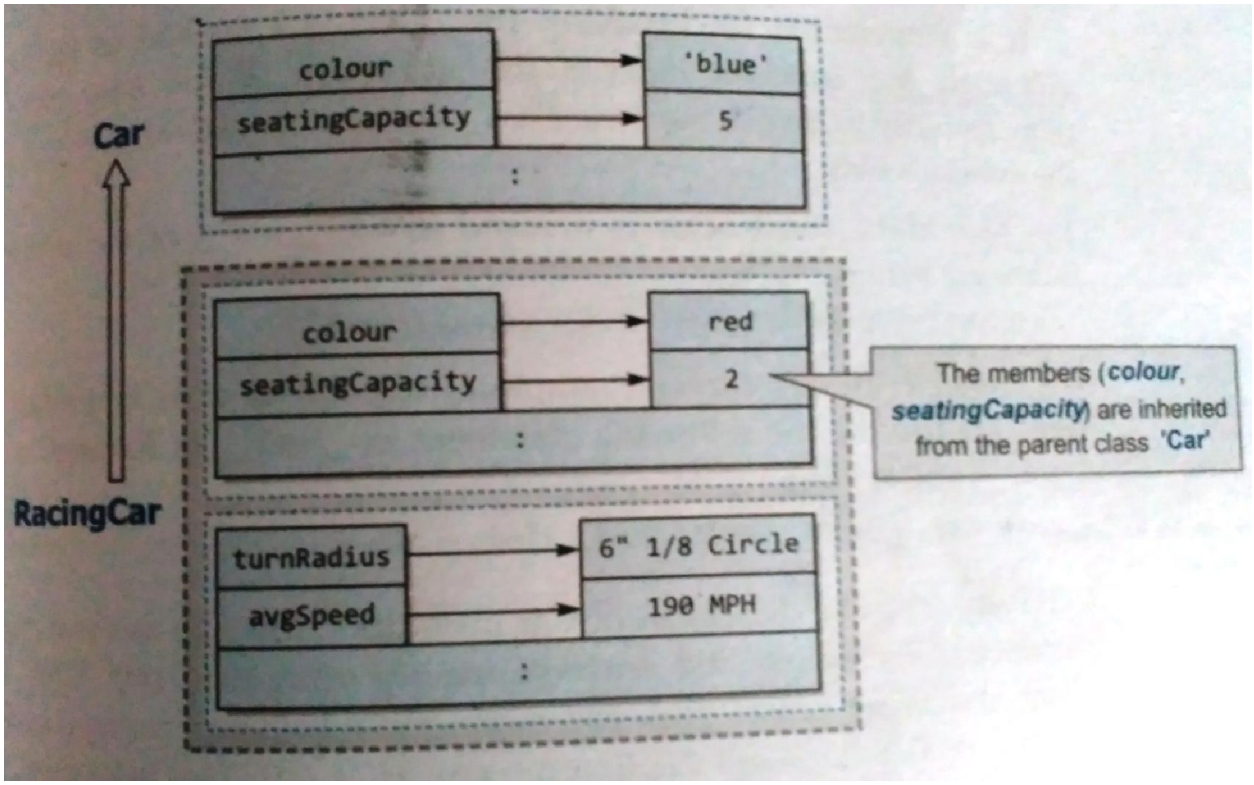
```
Class <Sublcass-Name>(<Parent_Class_Name>):
    [<Class' docstring>]
<suite of statement>
```

```
class Car:
    def __init__(self, clr, seats):
        self.colour=clr
        self.seatingCapacity=seats
    def Accelerate(self):
        :
```

Now define a sub class ie RacingCar

```
class RacingCar (Car):
    def __init__(self, turnRadius, speed):
        self.turnRadius=turnRadius
        self.speed=speed
    :
```

After above declarations, if you create objects of these two classes, the memory allocation would be somewhat like, as show below



The memory will be allocated for all the members of base class as well as derive class

In Python new style classes, every class inherit from a built in basic class called **object.** It is predefined by Python and always exists. As per new style classes, you can use object when you have no other super class.

The __init___ () of sub class

The __init__() method of a subclass requires special attention. The __init__() method of a subclass in Python has to perform the following things:

1. It is responsible for defining the additional ie, new members of the sub class, ie the members which only part of subclass and not of the super class.

```
class RacingCar (Car):

    def __init__(self, ……):

        self.turnRadius=turnRadius

        self.speed=speed
```

2. It is responsible for invoking the __init__() method of its super class.
   The base class constructor is invoked in the __init__() of the subclass as per the following syntax
   <BaseclassName>.__init__(<parameterlist>)
   For eg in our earlier example the RacingCar inherit from Car class, we will write following statement in __init__ of RacingCar

```
Car.__init__(self, 'red', 2)
```

While invoking __init__ of a class from outside its own class, you need to send self too.

If you forget to invoke the __init__ of base class then the instance variables of base class would not be created and hence will not be available to the sub class.

```
class A:
        def __init__(self, i='main'):
                self.name=i
class B(A):
        def __init__(self, j=10):
                self.age=j
ob1 = B(4)
print(ob1.name, ob1.age)
AttributeError: 'B' object has no attribute 'name'
```

This is because the subclass is responsible for invoking the __init__ of its base class and in above case, class B did not invoke the __init__ of class A, hence the attribute name was never created and hence the error.

3. The parameter list of __init__ of subclass includes parameters for the subclass as well as the super class.

```
class RacingCar(Car):
    def __init__(self, clr, seats, tr, spd):
        Car.__init__(self, clr, seats)
        self.turnRadius = tr
        self.avgSpeed = spd
```

Program :

```
class Car(object):
    def __init__(self, clr, seats):
        self.color= clr
        self.seatingCapacity=seats
    def accelerate(self, time, direction):
        print("Inside accelerate method")
    def turn(self, direction):
        print("Inside turn method")
    def applyBrakes(self,  pressure):
        print("Inside applyBrakes method")
    def __str__(self):
        return self.color + "Coloured Car with " +
str(self.seatingCapacity) + " seats"
#==========Derive class RacingCar==============
class RacingCar(Car):
    def __init__(self, clr, seats, tr, spd):
        Car.__init__(self, clr, seats)
```

```
                self.turnRadius = tr

                self.avgSpeed = spd

                print("Racing Car intance created")

        def __str__(self):

                string = self.color + "RacingCar with " +
str(self.turnRadius) + "\" turn radius speed " +
str(self.avgSpeed) + "rpm"

                return string

#================main====================

myCar = Car('Blue', 5)

rCar = RacingCar('Red', 2, 6, 190)

print("Base class Object \n", myCar)

print("Derive class Object ", rCar)
```

Class members and Inheritance

When a class inherit from a base-class, what all members of the base class are inherited by the subclass?

i) The public instance variable of Super class become accessible to sub class.

ii) The private instance variables of Super class are NOT available to sub class.

iii) The public class variables of super class become accessible to subclass.

```
class A:
        name = "interesting"
        def __init__(self):
                self.a, self.b=10, 20
class B(A):
        def __init__(self):
                self.c=30
```

```
            A.__init__(self)
        def __str__(self):
            string = "a :" + str(self.a) + " b :" +
            str(self.b) + " c :" + str(self.c)
            string += "& name : " + B.name
            return string
#================main====================
ob1 = B()
print(ob1)
print(B.name)
```

`B.name` , used class variable of super class with sub class name.

The Output of above code is

a :10 b :20 c :30& name : interesting

interesting

iv) The private class variable are NOT inheritable to the sub class.

Two useful functions

Python has two built-in functions that work with inheritance:

1. isinstance() function that can check an object's type. It is used as per following syntax
   isinstance(<object-name>, <class-name>)
   >>>isinstance(mycar, Car)
   True
2. issubclass() function that can check class inheritance. it is used as per following syntax
   issubclass(<subclassName>, <superclassName>)
   >>>issubclass(RacingCar, Car)
   True