**Experiment 3.3**

**Aim:** To write python program to Implement inheritance and polymorphism, Method overriding and overloading

**Solution:**

Inheritance and Polymorphism:

class Vehicle:

def \_\_init\_\_(self, name, color):

self.\_\_name = name

self.\_\_color = color

def getColor(self):

return self.\_\_color

def setColor(self, color):

self.\_\_color = color

def getName(self):

return self.\_\_name

class Car(Vehicle):

def \_\_init\_\_(self, name, color, model):

super().\_\_init\_\_(name, color)

self.\_\_model = model

def setColor(self):

print ("Its Overriding")

def getDescription(self):

return self.getName() + self.\_\_model + " in " +

self.getColor() + " color"

class Bike(Vehicle):

def \_\_init\_\_(self, name, color, model):

self.\_\_name = name

self.\_\_color = color

self.\_\_model = model

def getColor(self):

print ("Its Overloading")

return self.\_\_color

def getDescription(self):

return self.getName() + self.\_\_model + " in "

+ self.getColor() + " color"

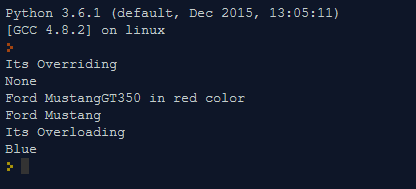
c = Car("Ford Mustang", "red", "GT350")

print(c.setColor())

print(c.getDescription())

print(c.getName())

b = Bike("Avenger", "Blue", "GTX220")

print(b.getColor())

Operator Overloading

import math

class Circle:

def \_\_init\_\_(self, radius):

self.\_\_radius = radius

def setRadius(self, radius):

self.\_\_radius = radius

def getRadius(self):

return self.\_\_radius

def area(self):

return math.pi \* self.\_\_radius \*\* 2

def \_\_add\_\_(self, another\_circle):

return Circle( self.\_\_radius + another\_circle.\_\_radius )

def \_\_gt\_\_(self, another\_circle):

return self.\_\_radius > another\_circle.\_\_radius

def \_\_lt\_\_(self, another\_circle):

return self.\_\_radius < another\_circle.\_\_radius

def \_\_str\_\_(self):

return "Circle with radius " + str(self.\_\_radius)

c1 = Circle(4)

print(c1.getRadius())

c2 = Circle(5)

print(c2.getRadius())

c3 = c1 + c2

print(c3.getRadius())

print( c3 > c2)

print( c1 < c2)

print(c3)

