##class and instance object

class Cat:

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

self.color=color

self.legs=legs

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

return self.color+','+str(self.legs)

if \_\_name\_\_=="\_\_main\_\_":

pet1=Cat("ginger",4)

print(pet1.legs)

print(pet1.color)

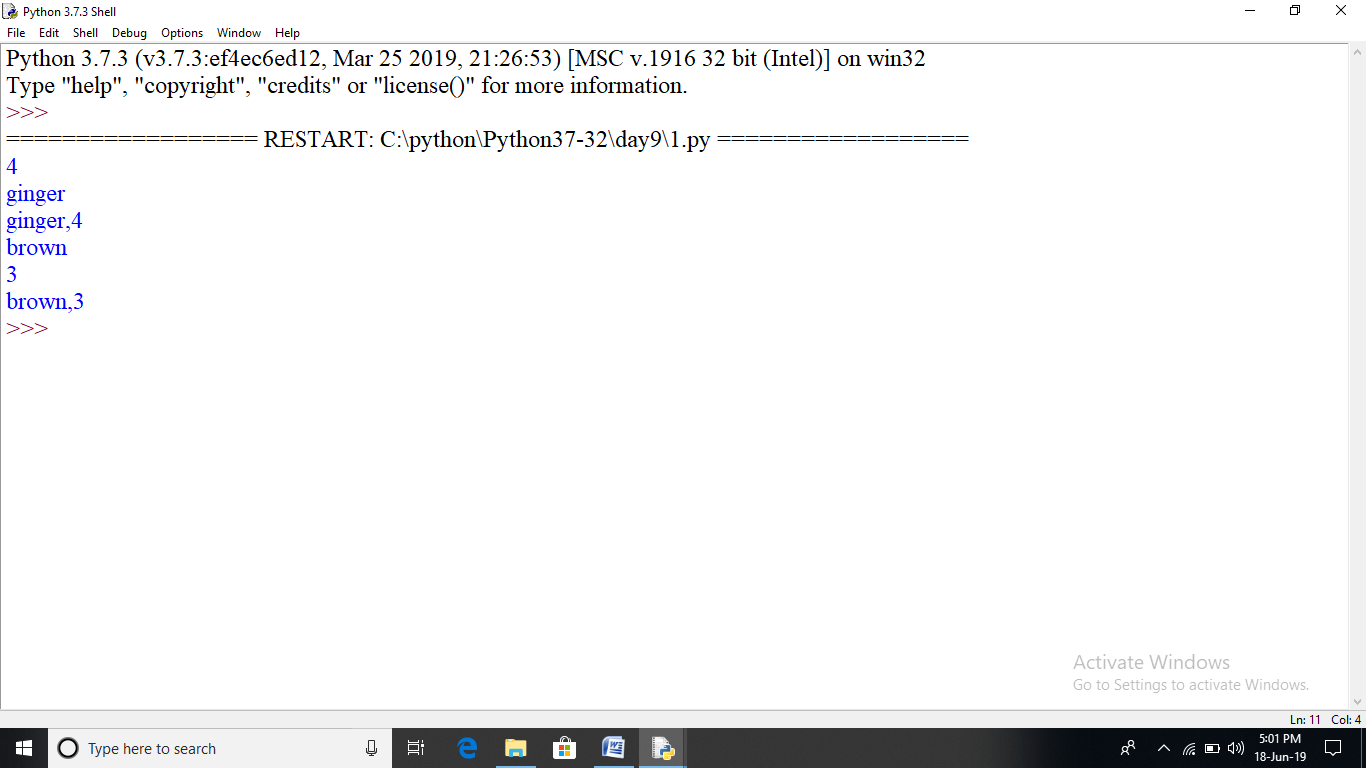
print(pet1)

pet2=Cat("brown",3)

print(pet2.color)

print(pet2.legs)

print(pet2)



class Dog:

price=400

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

self.color=color

self.name=name

def bark(self):

print("wool")

print(self.name,"has",self.price,"price and its color is ",self.color)

if \_\_name\_\_=='\_\_main\_\_':

pet1=Dog("tommy","brown")

pet2=Dog("shern","white")

pet1.bark()

pet2.bark()

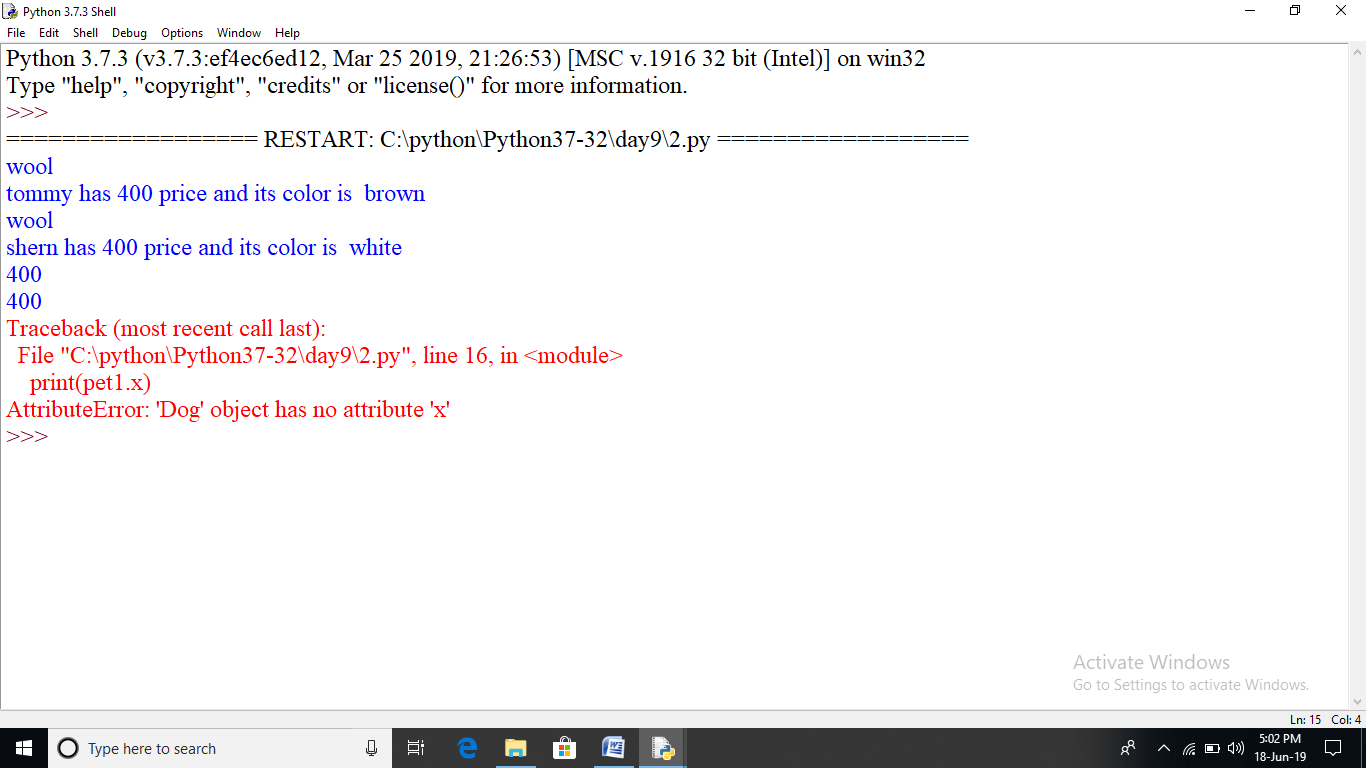
print(pet1.price)

print(pet2.price)

print(pet1.x)

print(Dog.price)

Dog("abc","blue").bark()



class Wolf:

price=500

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

self.name=name

self.color=color

def bark(self):

print("Grr")

class Dog(Wolf):

def bark1(self):

print("wool")

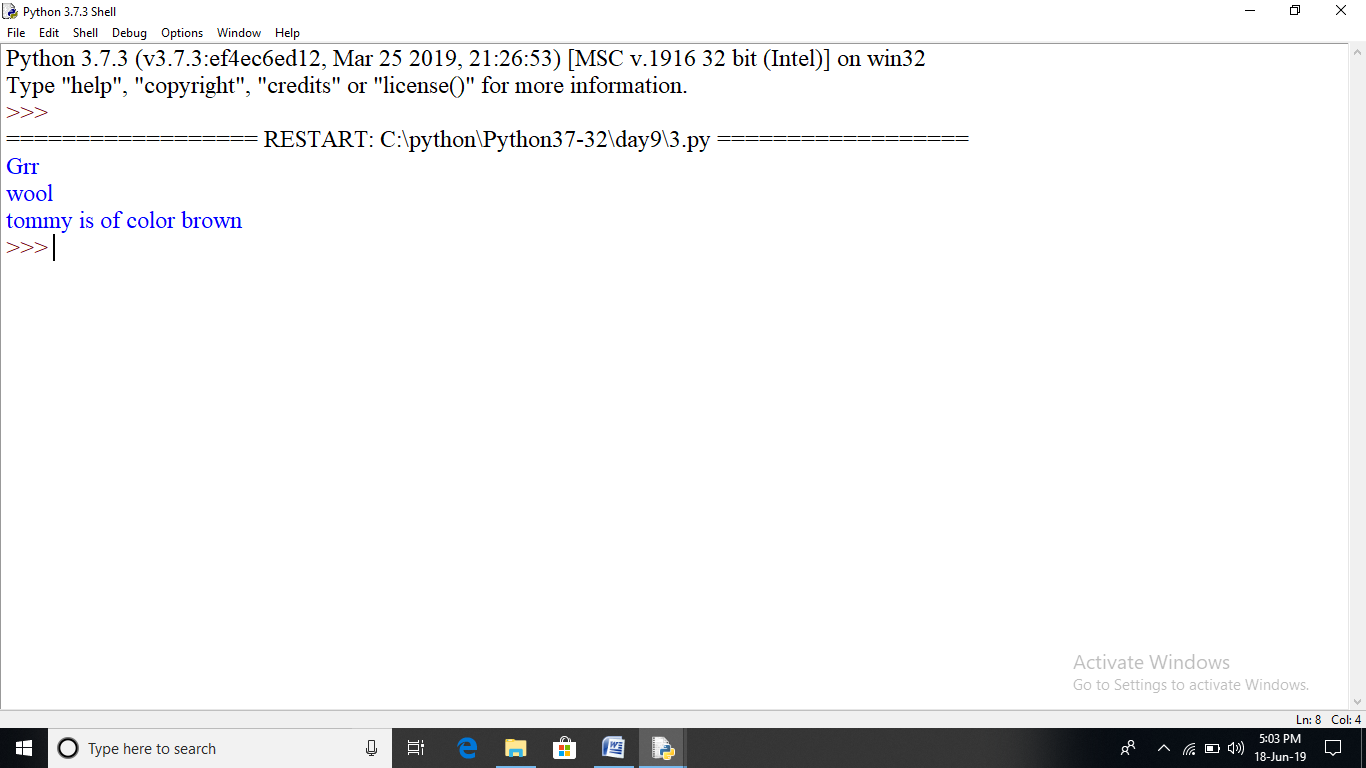
if \_\_name\_\_=="\_\_main\_\_":

pet1=Dog("tommy","brown")

pet1.bark()

pet1.bark1()

print(pet1.name,"is of color",pet1.color)



##multiple inheritance

class Animal:

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

self.name=name

self.color=color

class Cat(Animal):

def mew(self):

print("cat meows")

class Dog(Animal):

def bark(self):

print("woof")

if \_\_name\_\_=="\_\_main\_\_":

pet1=Dog("tomy","brown")

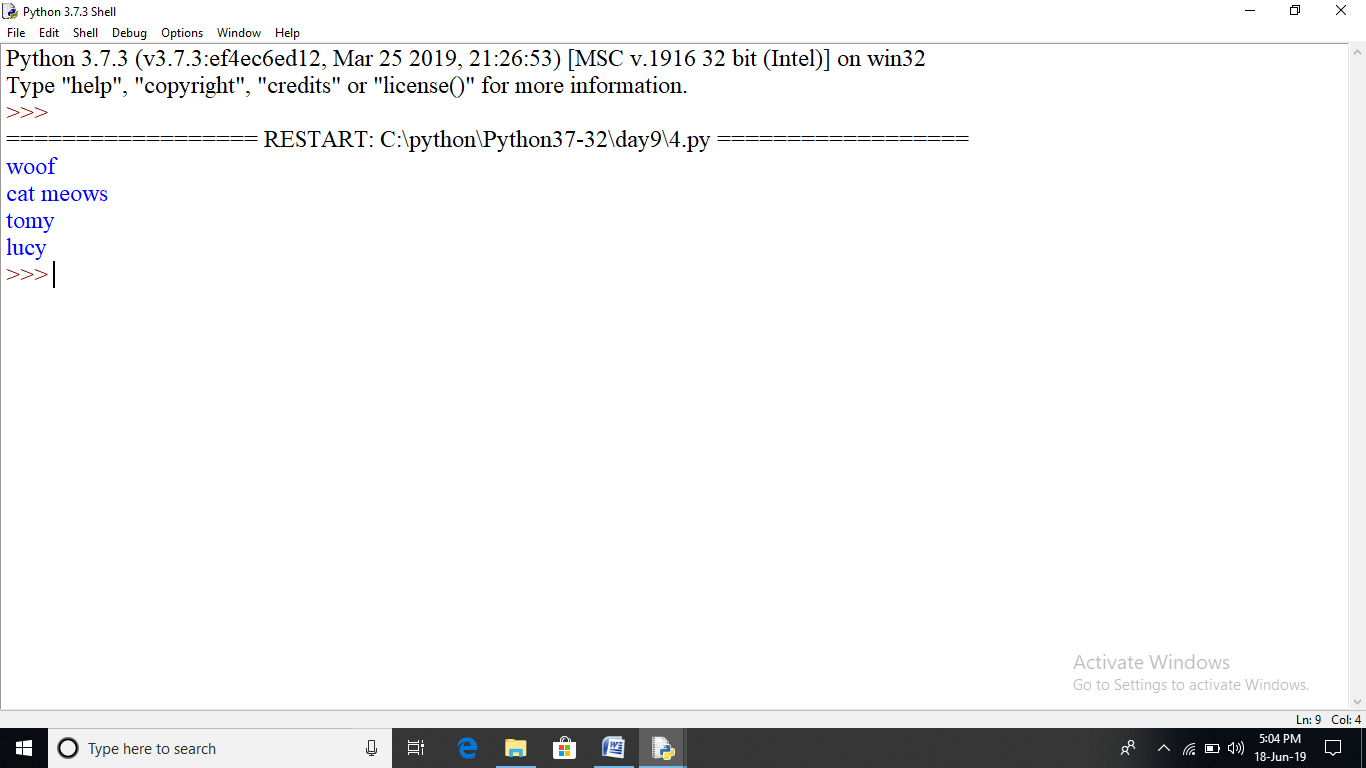
pet2=Cat("lucy","white")

pet1.bark()

pet2.mew()

print(pet1.name)

print(pet2.name)



class Wolf:

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

self.name=name

self.color=color

def bark(self):

print("Grr")

class Dog(Wolf):

def bark(self):

print("woof")

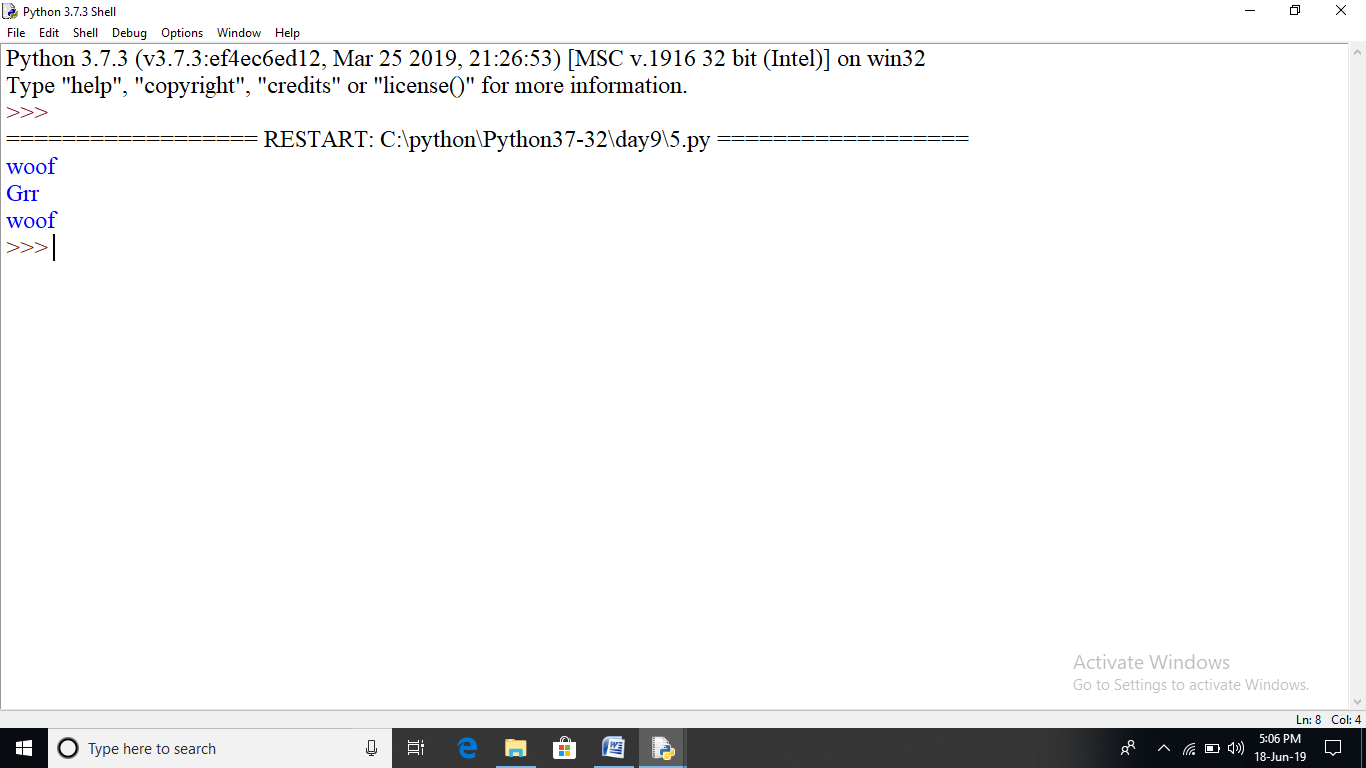
pet1=Dog("tommy","brown")

pet1.bark()

pet2=Wolf("jimmy","grey")

pet2.bark()

Dog("abc","xyz").bark()



#multilevel inheritance

class A:

def first\_method(self):

print("method of class A..")

class B(A):

def second\_method(self):

print("method of class B..")

class C(B):

def third\_method(self):

print("method of class C..")

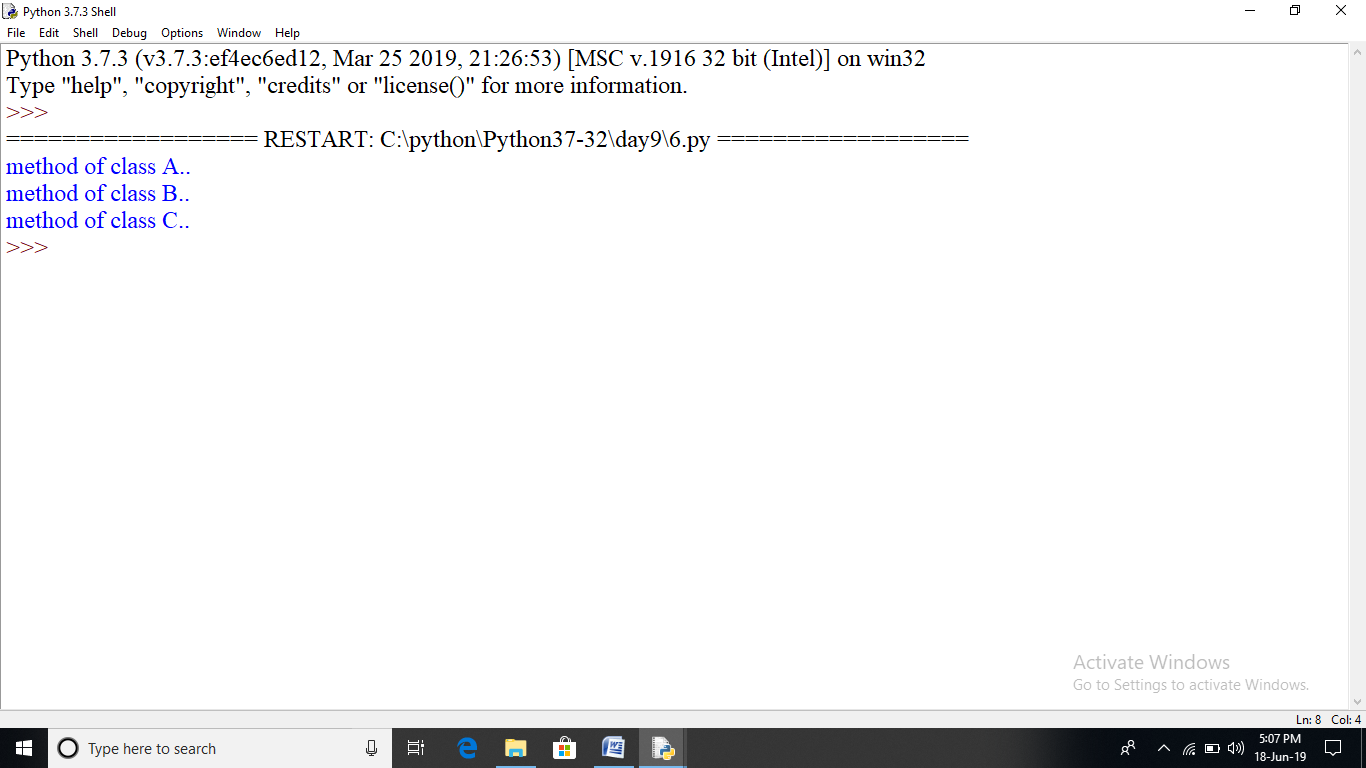
if \_\_name\_\_=="\_\_main\_\_":

ob=C()

ob.first\_method()

ob.second\_method()

ob.third\_method()



#multilevel inheritance

class A:

def first\_method(self):

print("method of class A..")

class B:

def second\_method(self):

print("method of class B..")

class C(A,B):

def third\_method(self):

print("method of class C..")

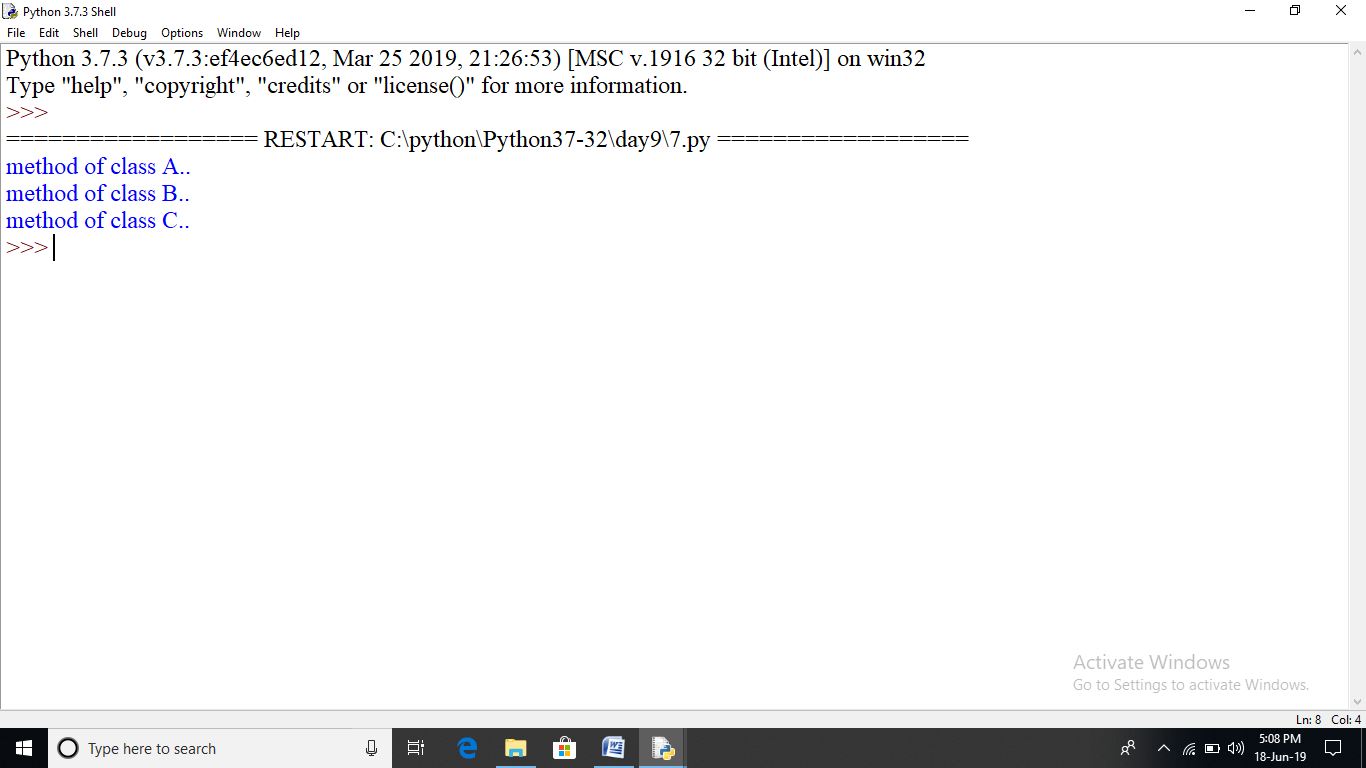
if \_\_name\_\_=="\_\_main\_\_":

ob=C()

ob.first\_method()

ob.second\_method()

ob.third\_method()



#multilevel inheritance or indirect inheritannce.

class A:

def first\_method(self):

print("method of class A..")

class B:

def first\_method(self):

print("method of class B..")

class C(B,A):

def third\_method(self):

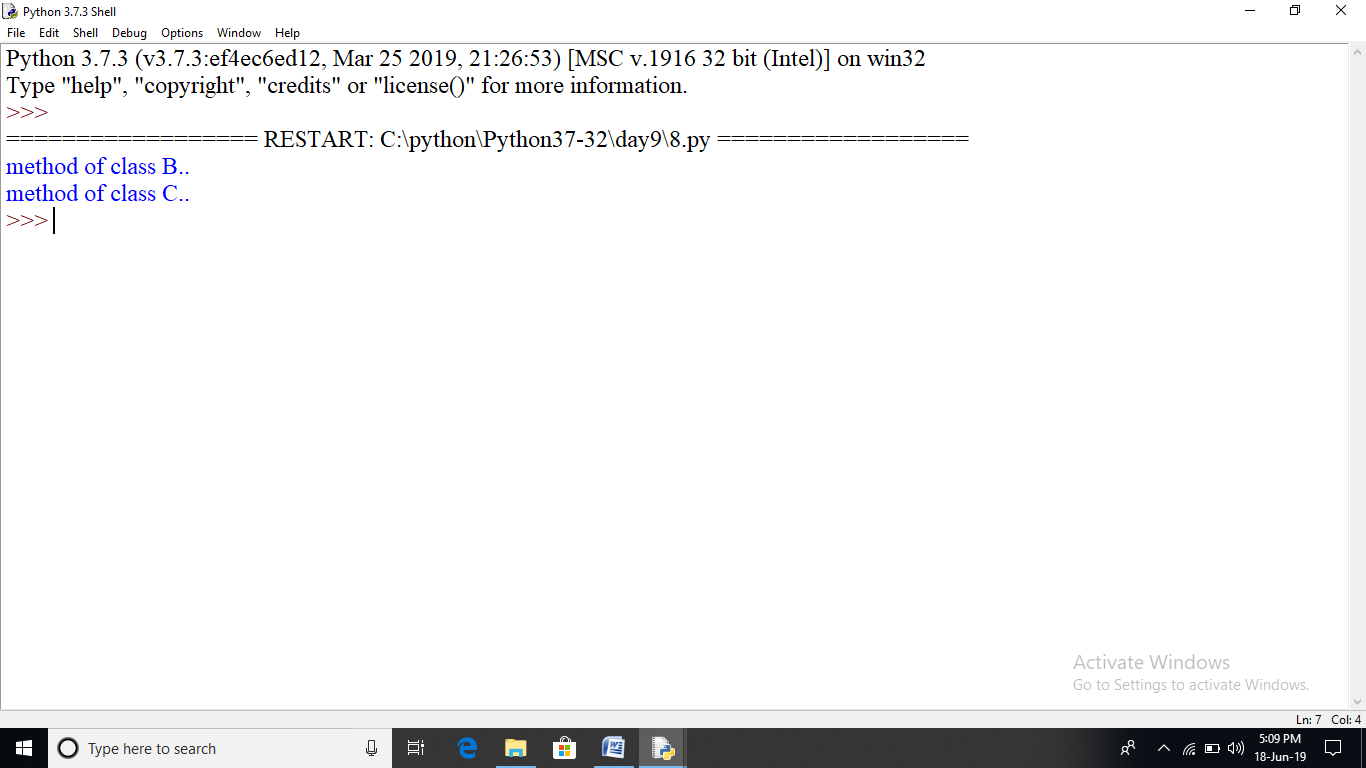
print("method of class C..")

if \_\_name\_\_=="\_\_main\_\_":

ob=C()

ob.first\_method()

ob.third\_method()



class A:

def first\_method(self):

print("method of class A..")

class B(A):

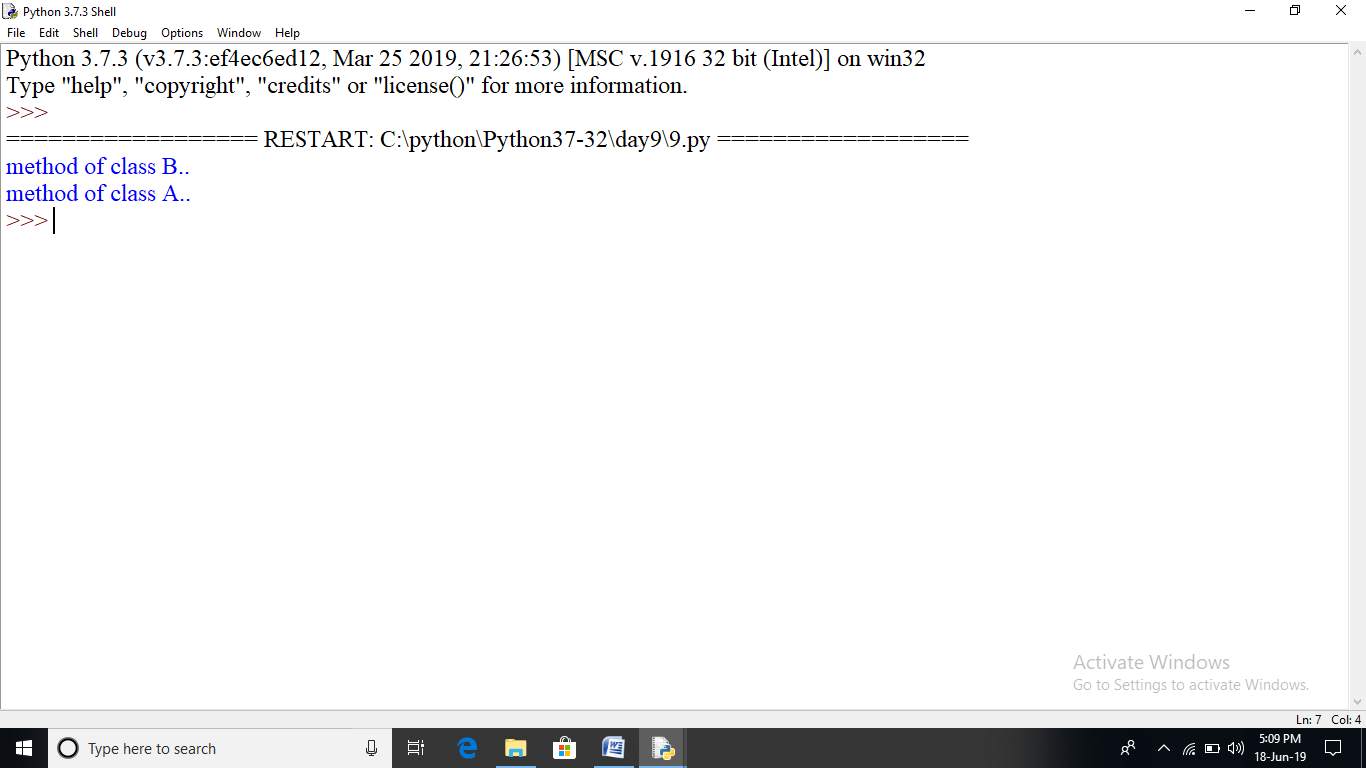
def first\_method(self):

print("method of class B..")

super().first\_method()

ob=B()

ob.first\_method()



class A:

def first\_method(self):

print("method of class A..")

class B(A):

def first\_method(self):

print("method of class B..")

class C(B):

def first\_method(self):

print("method of class C1..")

def third\_method(self):

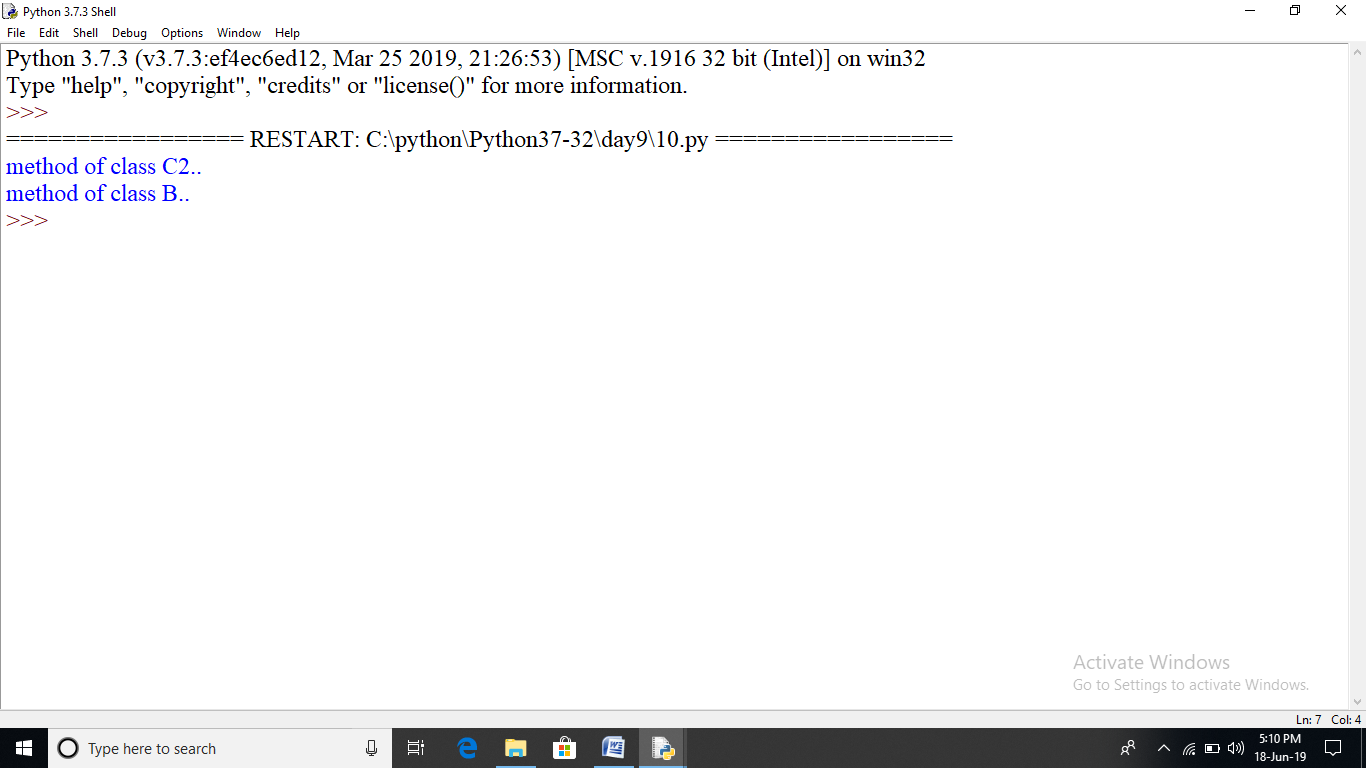
print("method of class C2..")

super().first\_method()

if \_\_name\_\_=="\_\_main\_\_":

ob=C()

ob.third\_method()



#magic method "dunders"

class Complex:

def \_\_init\_\_(self,real,img):

self.real=real

self.img=img

def \_\_add\_\_(self,ob):

return Complex(self.real+ob.real,self.img+ob.img)

def \_\_sub\_\_(self,ob1):

return Complex(self.real-ob1.real,self.img-ob1.img)

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

return str(self.real)+"+"+str(self.img)+"i"

z1=Complex(5,7)

z2=Complex(3,8)

z=z1+z2

print(z.real)

print(z.img)

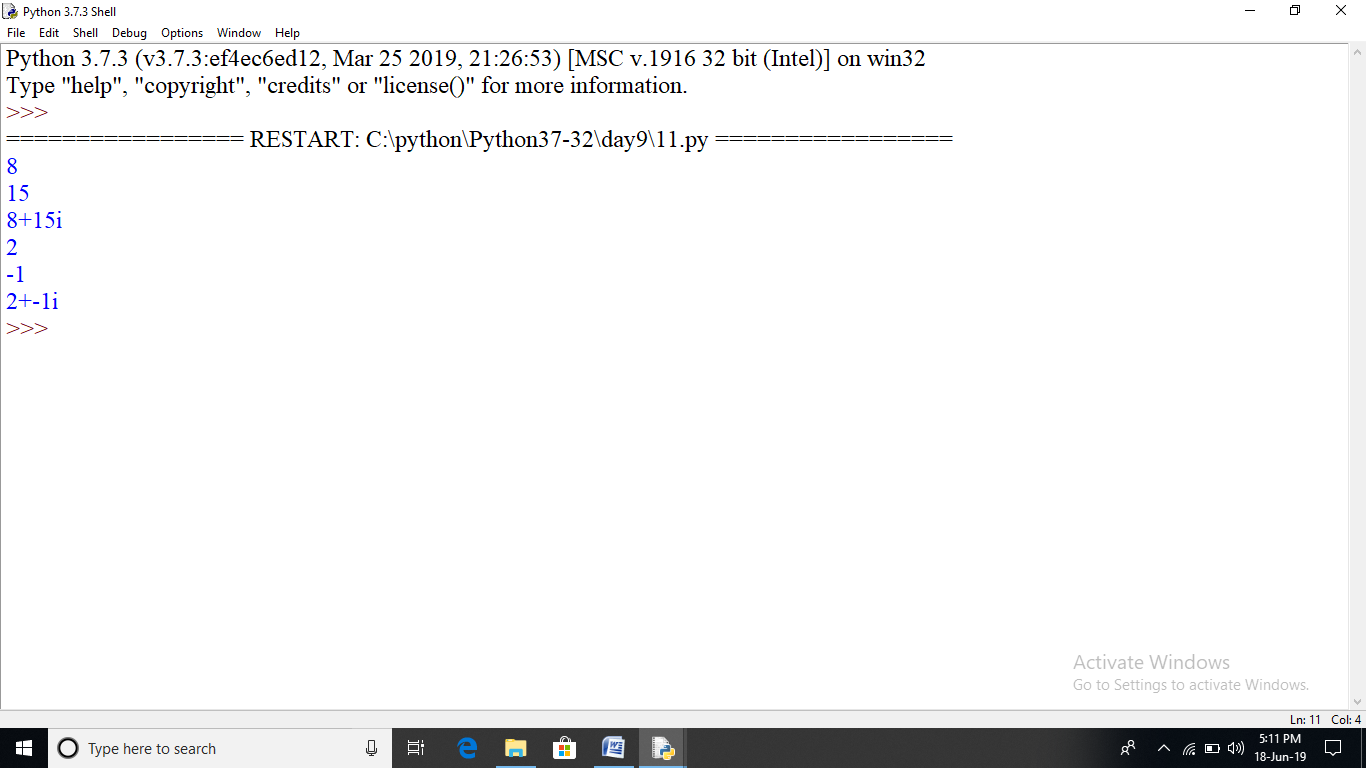
print(z)

z=z1-z2

print(z.real)

print(z.img)

print(z)



class Employee:

empCount=0

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

self.name=name

self.salary=salary

Employee.empCount+=1

def displayCount(self):

print("total employee %d" % Employee.empCount)

def displayEmployee(self):

print("Name;",self.name,",salary:",self.salary)

emp1=Employee("mahima",55000)

print("total Employee ",Employee.empCount)

emp2=Employee("Abhinn",54000)

emp1.displayEmployee()

emp2.displayEmployee()

print("total Employee ",Employee.empCount)

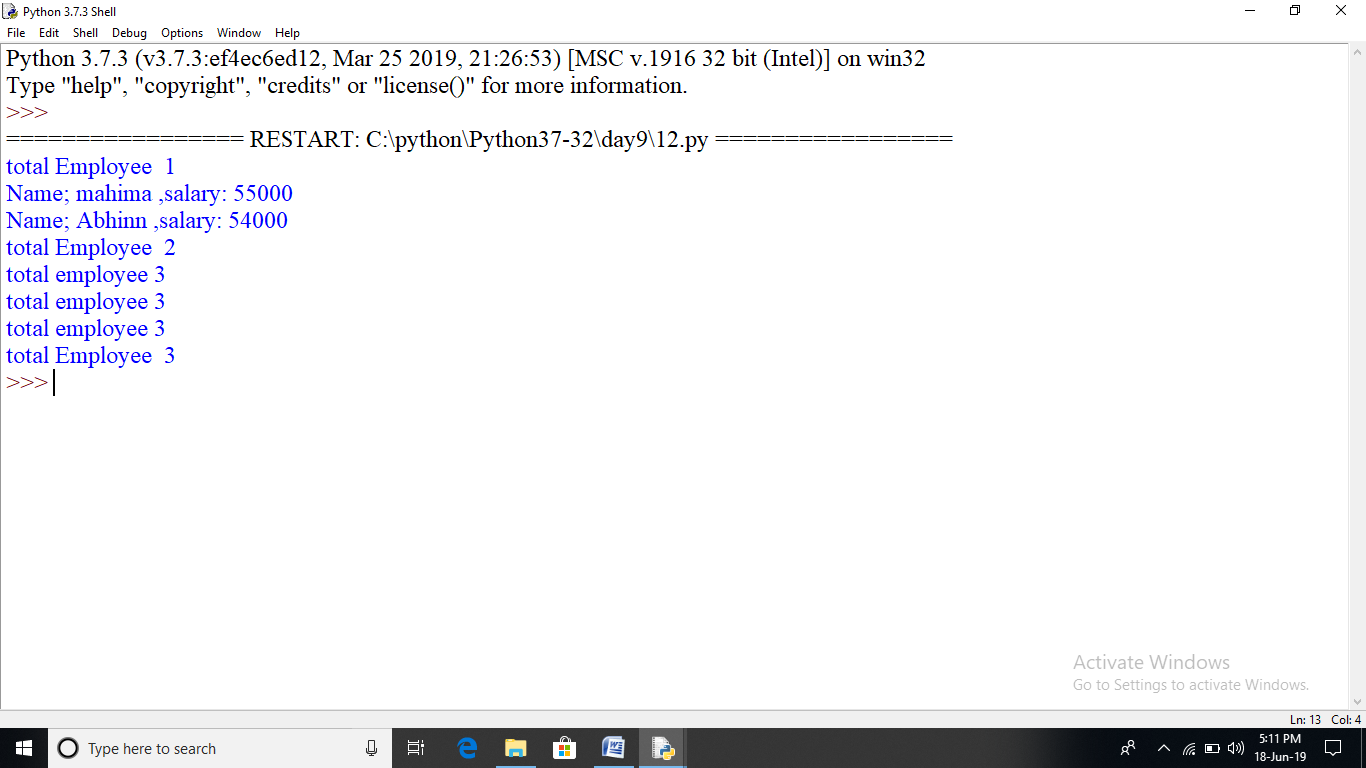
emp3=Employee("manu gupta",55500)

emp3.displayCount()

emp2.displayCount()

emp1.displayCount()

print("total Employee ",Employee.empCount)



class Employee:

empCount=0

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

self.name=name

self.salary=salary

Employee.empCount+=1

def displayCount(self):

print("total employee %d" % Employee.empCount)

def displayEmployee(self):

print("Name;",self.name,",salary:",self.salary)

emp1=Employee("nikhil",9999)

emp1.displayEmployee()

emp1.salary=17000

emp1.experience=3

emp1.displayEmployee()

emp1.name='manoj'

emp1.displayEmployee()

print(emp1.experience)

emp1.displayEmployee()

