## CLASS AND OBJECT

class car:

def \_\_init\_\_(self,brand,year):

self.brand=brand

self.year=year

def printf(self):

print(self.brand,',',self.year)

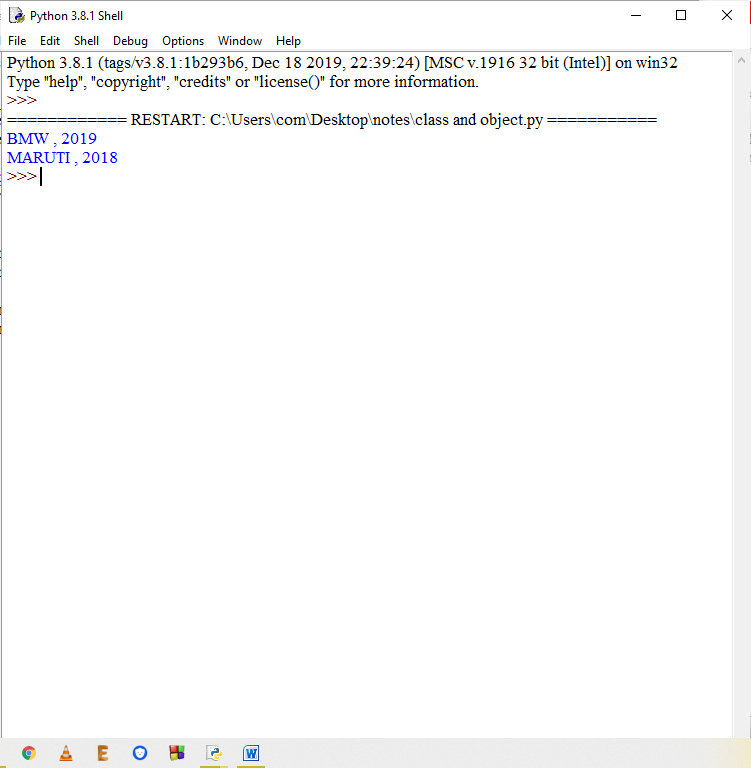
#main

c1=car('BMW',2019)

c2=car('MARUTI',2018)

c1.printf()

c2.printf()



## CLASS VARIABLES

class car:

wheels=4

def \_\_init\_\_(self,brand,year):

self.brand=brand

self.year=year

def printf(self):

print(self.brand,',',self.year,',',self.wheels)

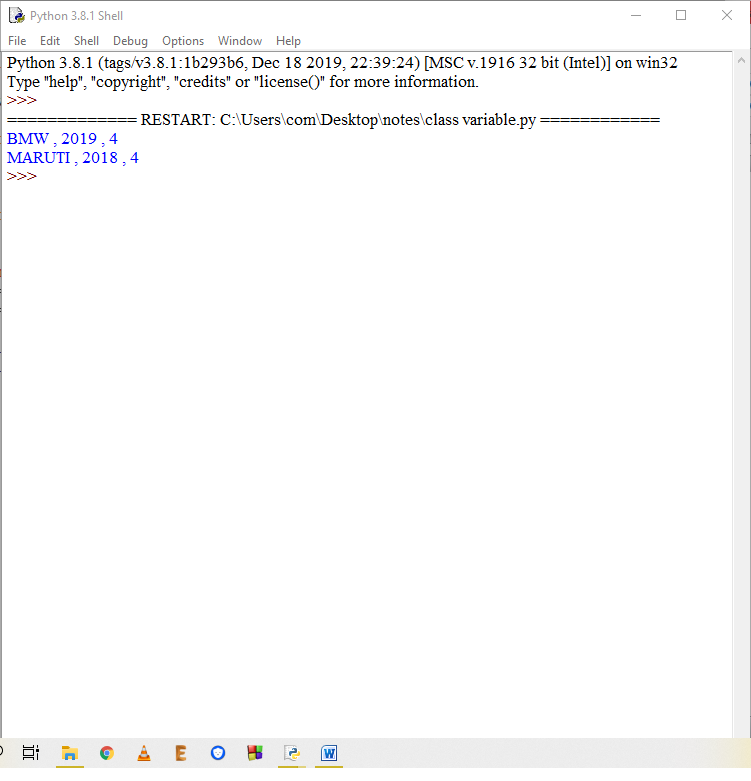
#main

c1=car('BMW',2019)

c2=car('MARUTI',2018)

c1.printf()

c2.printf()



## USE OF CLASS VARIABLES

class stud:

seats=10

def \_\_init\_\_(self,b,y):

self.name=b

self.age=y

def printf(self):

print(self.name,',',self.age)

def change():

stud.seats=stud.seats-1

#main

q=input("enter name: ")

w=int(input('Enter age: '))

c1=stud(q,w)

stud.change()

print("No of seats remaining: ",stud.seats)

input()

e=input("enter name: ")

r=int(input('Enter age: '))

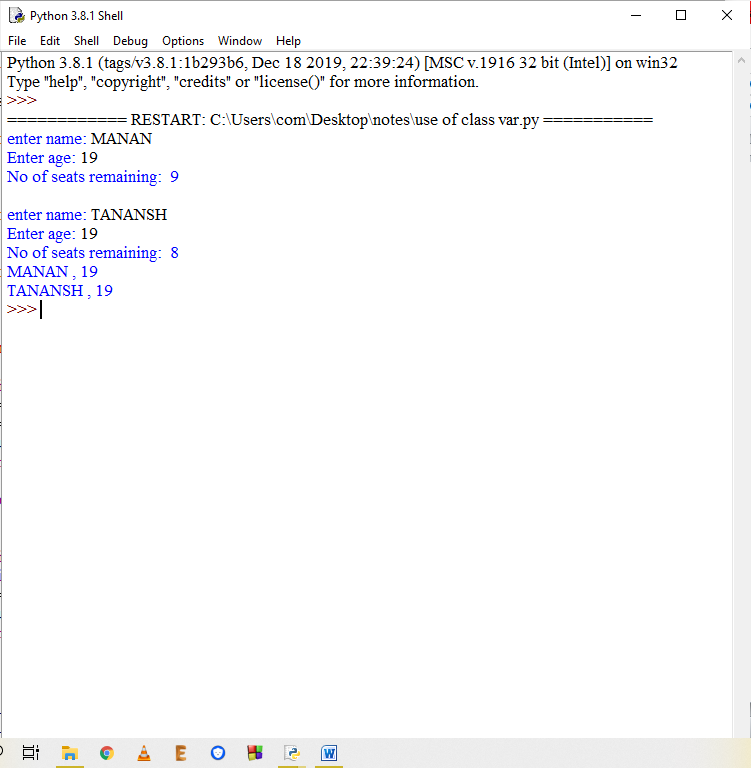
c2=stud(e,r)

stud.change()

print("No of seats remaining: ",stud.seats)

c1.printf()

c2.printf()



## TYPES OF METHODS

class student:

school='CSOFT'

def \_\_init\_\_(self,m1,m2,m3):

self.m1=m1

self.m2=m2

self.m3=m3

def avg(self):

return (self.m1+self.m2+self.m3)/3

@classmethod #this is a decorator

def info(cls):

return cls.school

@staticmethod #this is a static method

def thisclass():

print('This is student class.')

#main

s1=student(34,45,67)

s2=student(44,89,78)

print(s1.avg())

print(s2.avg())

print(student.info())

student.thisclass()



## INHERITANCE

class baseclass:

def feat1():

print('This is feature 1')

def feat2():

print('This is feature 2')

class derivedclass(baseclass):

def feat3():

print('This is feature 3')

def feat4():

print('This is feature 4')

obj = derivedclass

obj.feat1()

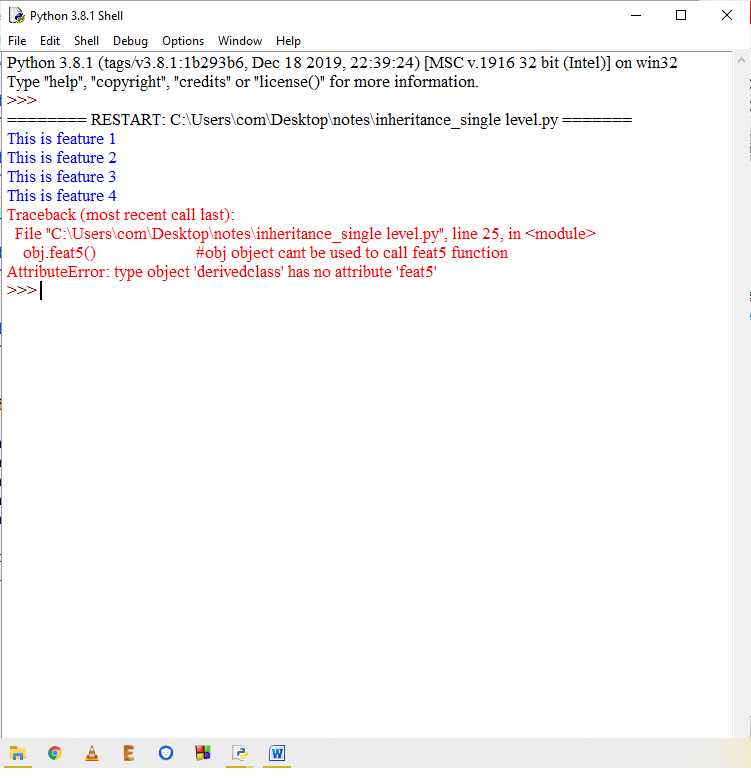
obj.feat2()

obj.feat3()

obj.feat4()

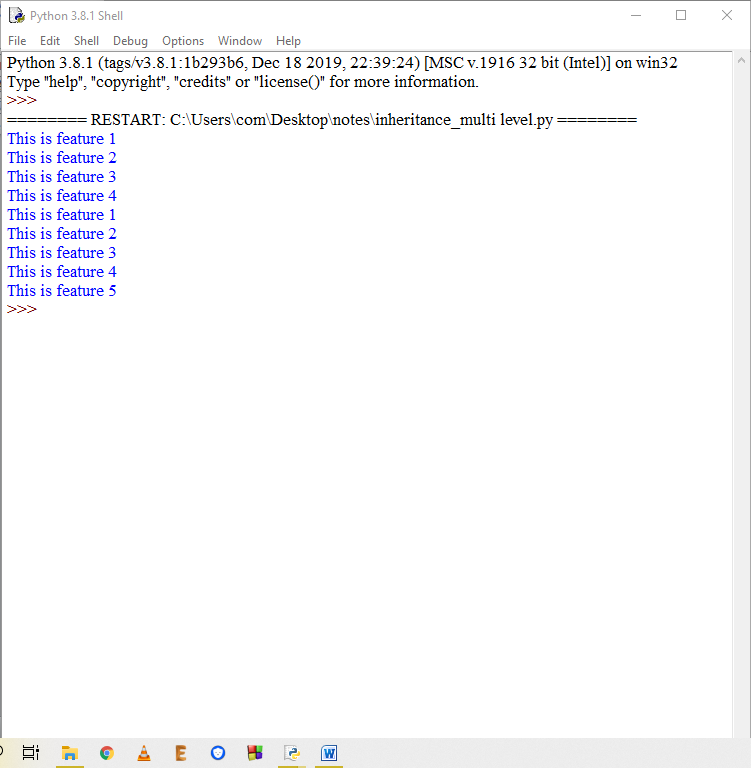
#with this you can conclude that even when 'obj' is an object of class 'derivedclass' , even then

#it is able to use the features of base class because derived class is inheriting base class



## INHERITANCE MULTILEVEL

class baseclass:



def feat1():

print('This is feature 1')

def feat2():

print('This is feature 2')

class derivedclass(baseclass):

def feat3():

print('This is feature 3')

def feat4():

print('This is feature 4')

class last(derivedclass):

def feat5():

print('This is feature 5')

obj = derivedclass

obj1=last

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()

#obj.feat5() #obj object cant be used to call feat5 function

obj1.feat1()

obj1.feat2()

obj1.feat3()

obj1.feat4()

obj1.feat5()

#with this you can conclude that even when 'obj' is an object of class 'derivedclass' , even then

#it is able to use the features of base class because derived class is inheriting base class

#also when obj1 is an object of last class it is able to use all the methods of its above inherited

#classes, this is what we call multilevel inheritance

## INHERITANCE MULTIPLE

class baseclass:

def feat1():

print('This is feature 1')

def feat2():

print('This is feature 2')

class derivedclass():

def feat3():

print('This is feature 3')

def feat4():

print('This is feature 4')

class last(baseclass,derivedclass):

def feat5():

print('This is feature 5')

obj1=last

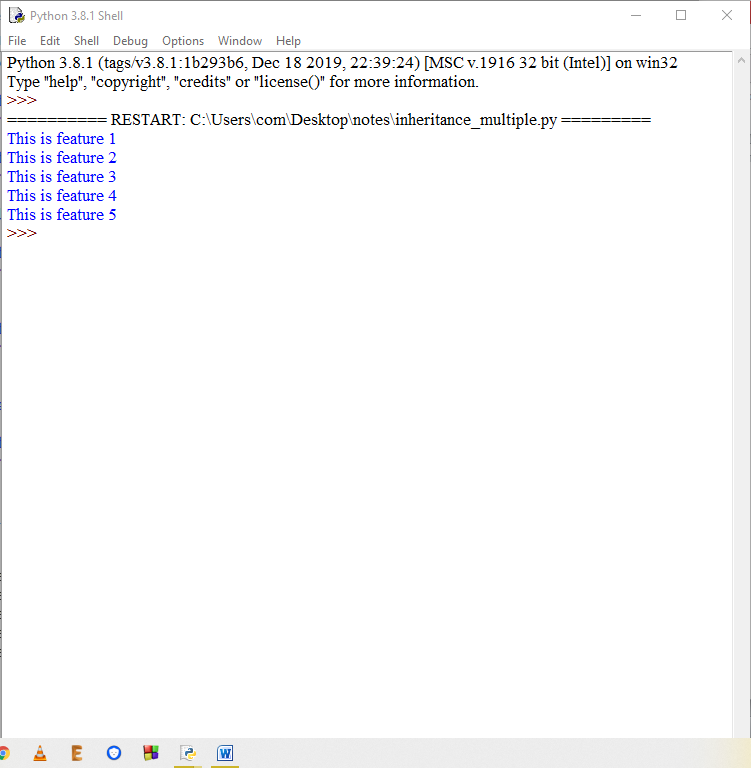
obj1.feat1()

obj1.feat2()

obj1.feat3()

obj1.feat4()

obj1.feat5()



## INHERITANCE INIT SINGLELEVEL

class baseclass:

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

print('In baseclass init')

def feat1(self):

print('This is feature 1')

def feat2(self):

print('This is feature 2')

class derivedclass(baseclass):

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

print('In derivedclass init')

def feat3(self):

print('This is feature 3')

def feat4(self):

print('This is feature 4')

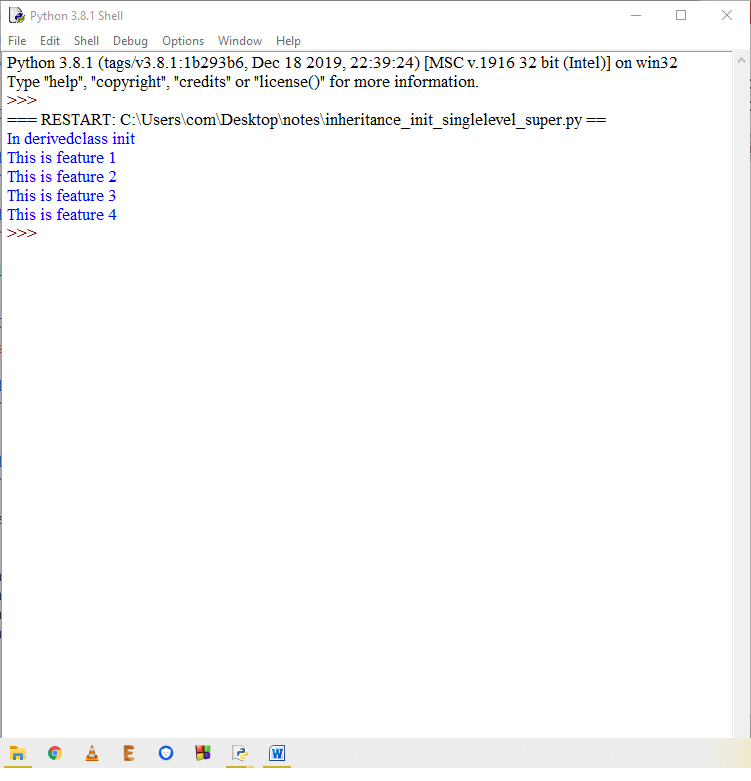
obj=derivedclass()

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()



## INHERITANCE INIT SUPER SINGLELEVEL

class baseclass:

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

print('In baseclass init')

def feat1(self):

print('This is feature 1')

def feat2(self):

print('This is feature 2')

class derivedclass(baseclass):

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

print('In derivedclass init')

super().\_\_init\_\_()

def feat3(self):

print('This is feature 3')

def feat4(self):

print('This is feature 4')

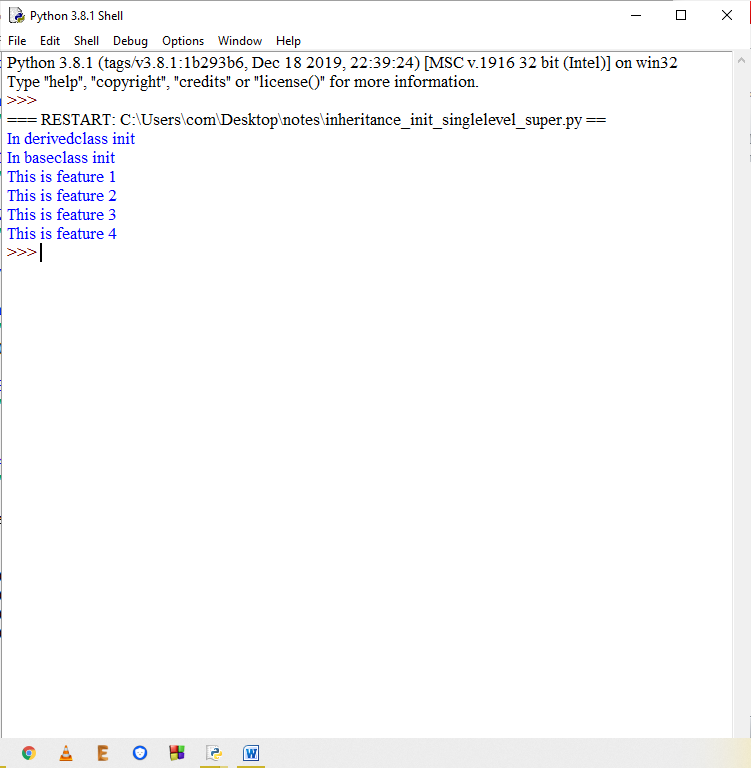
obj=derivedclass()

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()



## INHERITANCE INIT MULTILEVEL

class baseclass:

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

print('In baseclass init')

def feat1(self):

print('This is feature 1')

def feat2(self):

print('This is feature 2')

class derivedclass(baseclass):

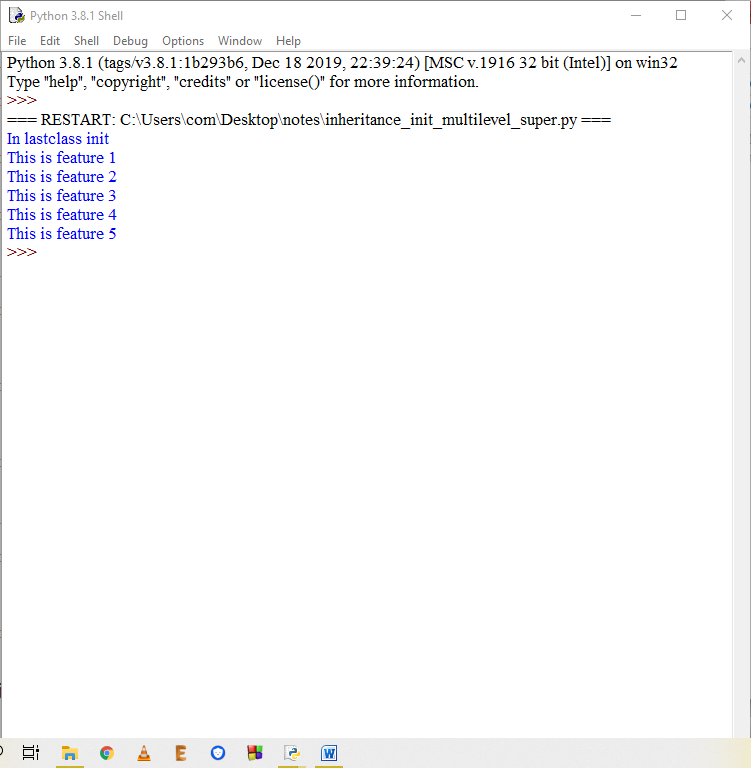
def \_\_init\_\_(self):

print('In derivedclass init')

def feat3(self):

print('This is feature 3')

def feat4(self):



print('This is feature 4')

class lastclass(derivedclass):

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

print('In lastclass init')

def feat5(self):

print('This is feature 5')

obj=lastclass()

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()

obj.feat5()

## INHERITANCE INIT MULTILEVEL SUPER

class baseclass:

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

print('In baseclass init')

def feat1(self):

print('This is feature 1')

def feat2(self):

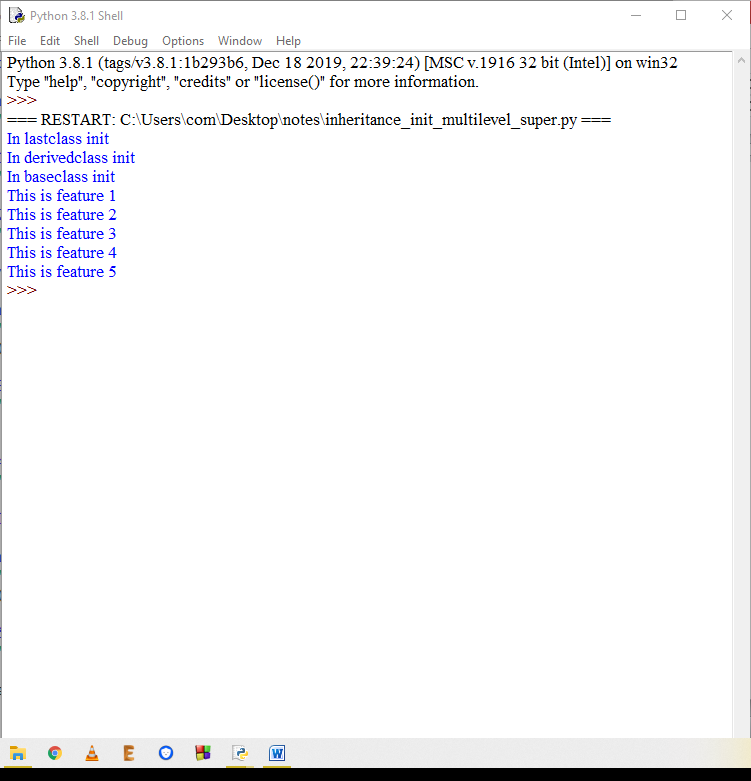
print('This is feature 2')

class derivedclass(baseclass):

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

print('In derivedclass init')

super().\_\_init\_\_()



def feat3(self):

print('This is feature 3')

def feat4(self):

print('This is feature 4')

class lastclass(derivedclass):

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

print('In lastclass init')

super().\_\_init\_\_()

def feat5(self):

print('This is feature 5')

obj=lastclass()

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()

obj.feat5()

## INHERITANCE INIT MULTILEVEL SUPER

class baseclass:

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

print('In baseclass init')

def feat1(self):

print('This is feature 1')

def feat2(self):

print('This is feature 2')

class derivedclass(baseclass):

def feat3(self):

print('This is feature 3')

def feat4(self):

print('This is feature 4')

class lastclass(derivedclass):

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

print('In lastclass init')

super().\_\_init\_\_()

def feat5(self):

print('This is feature 5')

obj=lastclass()

obj.feat1()

obj.feat2()

obj.feat3()

obj.feat4()

obj.feat5()

