**13/04/2019 Lecture26**

**Inheritance**

**Importing class**

**Person.py**

#!/usr/bin/python

class Person:

def \_\_init\_\_(self,name,address,dob):

self.\_\_name = name

self.\_\_address = address

self.\_\_dob = dob

def getName(self):

return self.\_\_name

def getAddress(self):

return self.\_\_address

def getDOB(self):

return self.\_\_dob

def updateAddress(self, address):

self.\_\_address = address

**Student\_management\_system.py**

#!/usr/bin/python

from Person import Person

class Student(Person):

auto\_roll\_no = 1

def \_\_init\_\_(self, name, address, dob, course, division):

Person.\_\_init\_\_(self,name, address, dob)

self.\_\_roll\_no = Student.auto\_roll\_no

Student.auto\_roll\_no += 1

self.\_\_name = name

self.\_\_address = address

self.\_\_dob = dob

self.\_\_course = course

self.\_\_division = division

self.\_\_marks = dict()

def getRollNo(self):

return self.\_\_roll\_no

def getName(self):

return Person.getName(self)

def getAddress(self):

return Person.getAddress(self)

def getDOB(self):

return Person.getDOB(self)

def getCourse(self):

return self.\_\_course

def getDivision(self):

return self.\_\_division

def getMarks(self):

return self.\_\_marks

def updateMarks(self, subject, marks):

self.\_\_marks[subject] = marks

def updateCourse(self, course):

self.\_\_course = course

def updateDivision(self, division):

self.\_\_division = division

def \_\_repr\_\_(self):

return "Name:"+self.getName()+"\nAddress:"+self.getAddress()+"\nDOB:"+self.getDOB()+"\nRoll No:"+str(self.\_\_roll\_no)

**#Homework : Implement Library Management System**

**Multiple inheritance with 2.7 and 3.X**

import inspect

class A(): #use in 3.x directly

#class A(object): #USE in 2.7

def mMethod(self):

print("In m of A")

def kMethod(self):

print("In k of A")

class B(A):

def lMethod(self):

print("in l of B")

class C(A):

def nMethod(self):

print("in n of C")

def mMethod(self):

print("in m of C")

class D(B,C):

def bMethod(self):

print("in b of D")

def cMethod(self):

print("in c of D")

def main():

d = D()

d.mMethod()

d.cMethod()

print(inspect.getmro(D))

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

main()

**To mke object callable use \_\_Call\_\_(self)**

class Simple:

def \_\_call\_\_(self):

print("Object Invoked")

s = Simple()

s()

**# Static method calling (using decorators (@))**

class Demo:

@staticmethod

def InvokeStatic():

print("In static method call")

@classmethod

def InvokeClassMethod(cls):

print("In class method: ",type(cls),id(cls))

def InvokeObjectMethod(self):

print("In object method: ")

print(id(Demo))

Demo.InvokeClassMethod()

Demo.InvokeStatic()

d= Demo()

Demo.InvokeObjectMethod(d)

d.InvokeStatic()