# Welcome to the Python\_and\_Deep\_Learning\_Course-CSEE5590 Lab-2 submission

By:

Mani Sai Srinivas, Kandukuri

Class-id: 20

## Objective

This lab mainly consists of learning Python Objective-Oriented Programming.

First, second, fourth were done in "One" jupyter notebook(both .py and .ipnyb format was exported) and third was done in PyCharm.

All of the code is commented for understanding. Code can be found [here](https://github.com/manisaisrinivask/Python_and_Deep_Learning_Course-CSEE5590/tree/master/lab-2/code)

## Configuration

* Python 2.7
* Jupyter Notebook
* PyCharm

## Task-1

Consider a shop UMKC with dictionary of all book items with their prices. Write a program to find the books from the dictionary in the range given by use

Here is the screenshot of the code(both input and output):



Code Snippet:

# giving names of books and prices in a dictionary called d

d={"python":20,"web":40,"c":30,"java":10}

lst=[]

# books in range of 10 and 20

for k,v in d.iteritems():

if v>=10 and v<=20:

lst.append(k)

print "these are the books you can buy: " + " ".join(str(x)for x in lst)

Output:

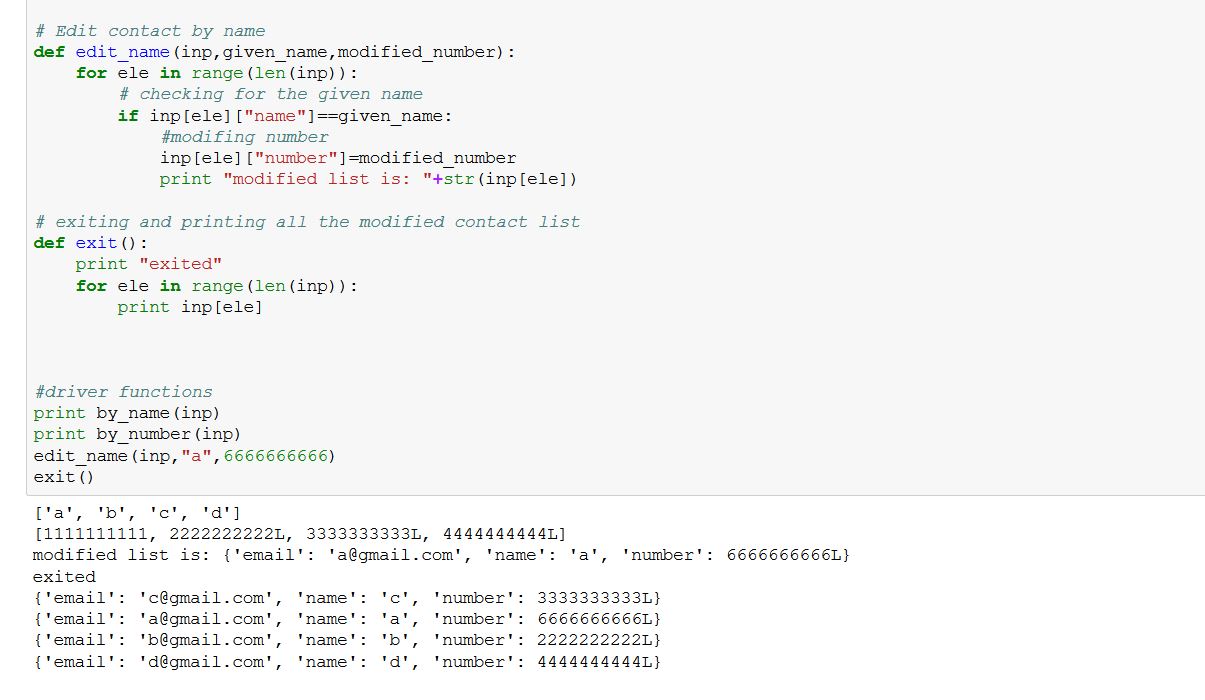
The output was java and python as they are in range

## Task-2

With any given number n, In any mobile , there is contact list. Create a list of contacts and then prompt the user to do the following: a)Display contact by name b)Display contact by number c)Edit contact by name d)Exit

This is a screenshot of the code(both input and output):





Code Snippet:

inp = [{"name": 'c', "number": 3333333333, "email": "c@gmail.com"},

{"name": "a", "number": 1111111111, "email": "a@gmail.com"},

{"name": "b", "number": 2222222222, "email": "b@gmail.com"},

{"name": "d", "number": 4444444444, "email": "d@gmail.com"}]

# Sorting b\_name

def by\_name(inp):

lst = []

# Searching foe name

for ele in range(len(inp)):

lst.append(inp[ele]["name"])

# Sorting them

return sorted(lst)

# Sorting by number

def by\_number(inp):

lst = []

# Searching for number

for ele in range(len(inp)):

lst.append(inp[ele]["number"])

# Sorting them

lst.sort()

return lst

# Edit contact by name

def edit\_name(inp, given\_name, modified\_number):

for ele in range(len(inp)):

# checking for the given name

if inp[ele]["name"] == given\_name:

# modifing number

inp[ele]["number"] = modified\_number

print "modified list is: " + str(inp[ele])

# exiting and printing all the modified contact list

def exit():

print "exited"

for ele in range(len(inp)):

print inp[ele]

# driver functions

print by\_name(inp)

print by\_number(inp)

edit\_name(inp, "a", 6666666666)

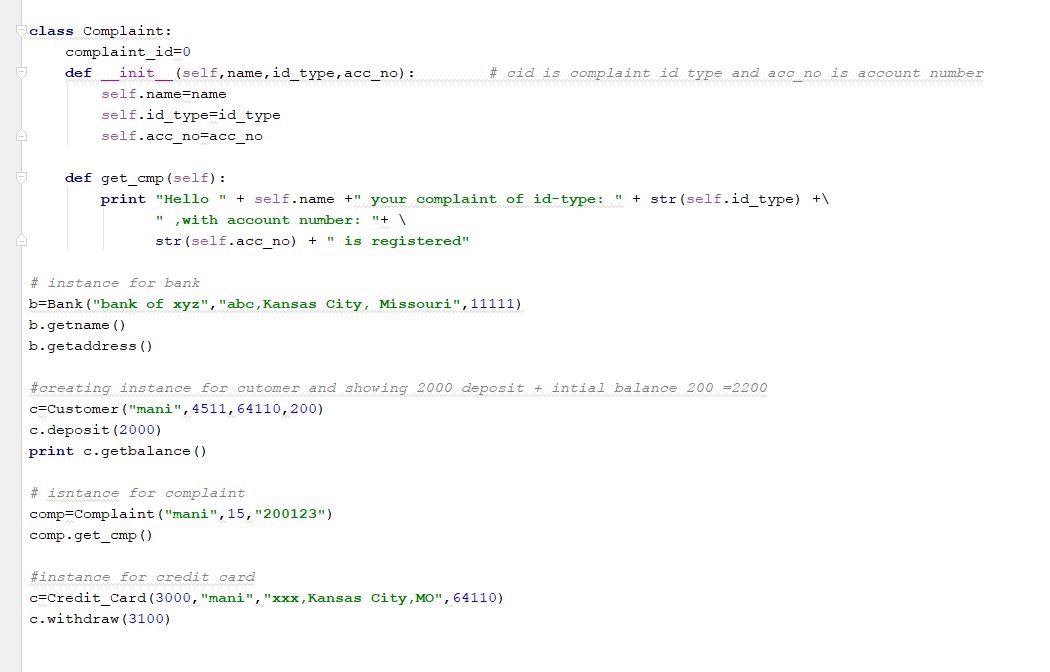
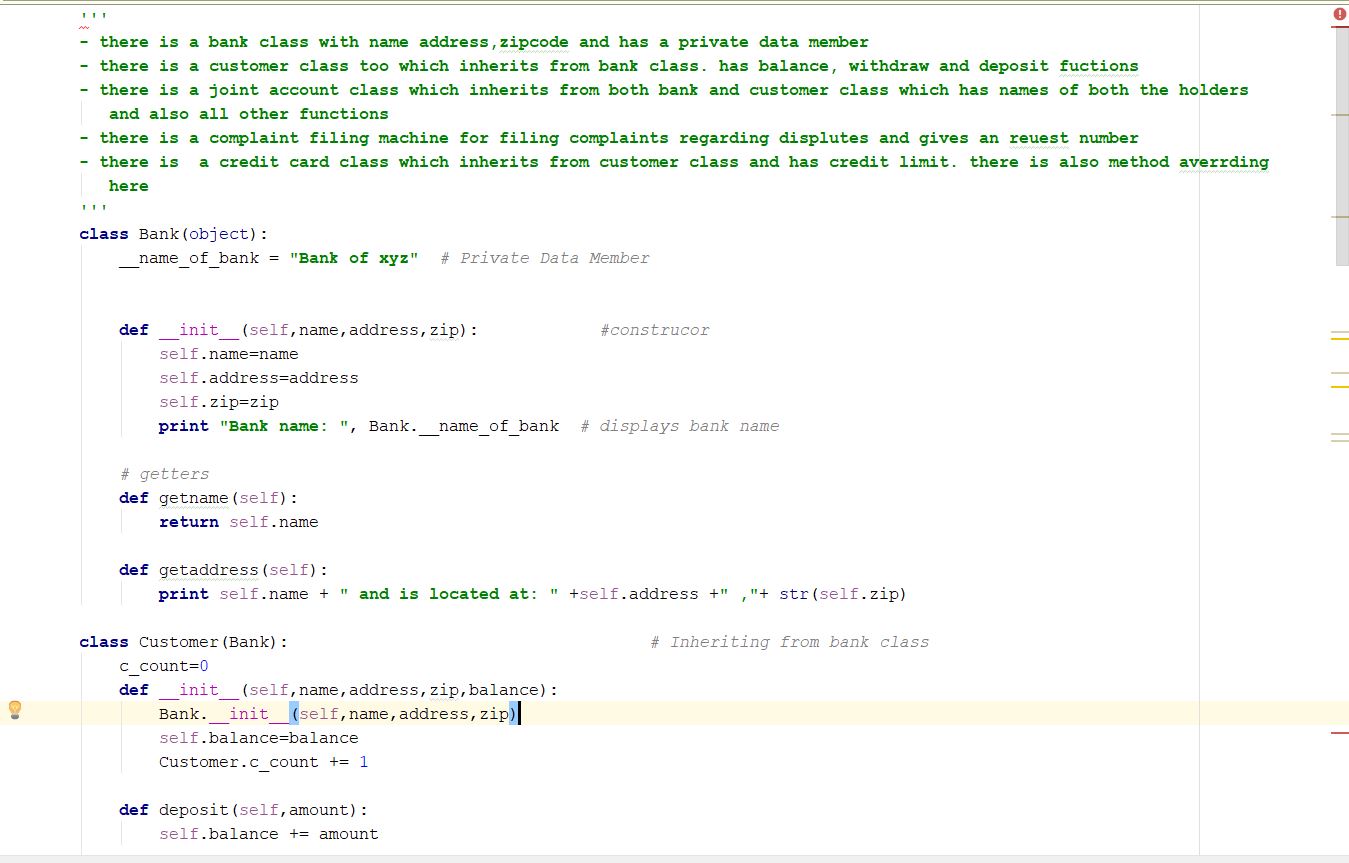
exit()

## Task-3

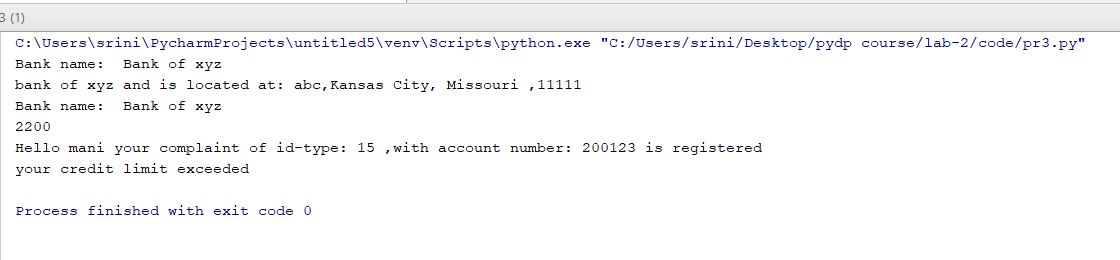
Write a python program to create any one of the following management systems. You can also pick one of your own.

I created a banking System

Screenshot of the code(Input and Output):



Output:



Code Snippet: ''' - there is a bank class with name address,zipcode and has a private data member - there is a customer class too which inherits from bank class. has balance, withdraw and deposit functions - there is a joint account class which inherits from both bank and customer class which has names of both the holders and also all other functions - there is a complaint filing machine for filing complaints regarding disputes and gives an request number - there is a credit card class which inherits from customer class and has credit limit. there is also method overriding here ''' class Bank(object): \_\_name\_of\_bank = "Bank of xyz" # Private Data Member

def \_\_init\_\_(self,name,address,zip): #construcor

self.name=name

self.address=address

self.zip=zip

print "Bank name: ", Bank.\_\_name\_of\_bank # displays bank name

# getters

def getname(self):

return self.name

def getaddress(self):

print self.name + " and is located at: " +self.address +" ,"+ str(self.zip)

class Customer(Bank): # Inheriting from bank class

c\_count=0

def \_\_init\_\_(self,name,address,zip,balance):

Bank.\_\_init\_\_(self,name,address,zip)

self.balance=balance

Customer.c\_count += 1

def deposit(self,amount):

self.balance += amount

def withdraw(self,amount):

if self.amount>self.balance:

print "Your balance is not sufficient"

else:

self.balance -= amount

def getbalance(self):

return self.balance

def no\_customers(self):

print "the number of customers are: "+Customer.c\_count

def c\_display(self):

print "the name of customer is" +self.getname()

class Joint\_account(Customer,Bank): #multiple inheritance

def \_\_init\_\_(self,first\_person,second\_person):

super(Joint\_account,self).\_\_init\_\_() # Using super class

self.first\_person=first\_person

self.second\_person=second\_person

def get\_names(self):

print "the account names are: " +self.first\_person + " and " +self.second\_person

class Credit\_Card: # inheriting from customer class

def \_\_init\_\_(self,credit\_limit,name,address,zip):

self.credit\_limit=credit\_limit

def withdraw(self,amount): # method overriding

if amount>self.credit\_limit:

print "your credit limit exceeded"

else:

self.credit\_limit -= amount

def deposit(self,amount):

self.credit\_limit += amount

def get\_credit\_limit(self):

return self.credit\_limit

class Complaint:

complaint\_id=0

def \_\_init\_\_(self,name,id\_type,acc\_no): # cid is complaint id type and acc\_no is account number

self.name=name

self.id\_type=id\_type

self.acc\_no=acc\_no

def get\_cmp(self):

print "Hello " + self.name +" your complaint of id-type: " + str(self.id\_type) +\

" ,with account number: "+ \

str(self.acc\_no) + " is registered"

# instance for bank

b=Bank("bank of xyz","abc,Kansas City, Missouri",11111)

b.getname()

b.getaddress()

#creating instance for cutomer and showing 2000 deposit + intial balance 200 =2200

c=Customer("mani",4511,64110,200)

c.deposit(2000)

print c.getbalance()

# isntance for complaint

comp=Complaint("mani",15,"200123")

comp.get\_cmp()

#instance for credit card

c=Credit\_Card(3000,"mani","xxx,Kansas City,MO",64110)

c.withdraw(3100)

## Task-4

Using Numpy create random vector of size 15 having only Integers in the range 0 -20. Write a program to find the most frequent item/value in the vector list.

Screenshot of the code(Input and Output):



Code Snippet:

import numpy as np

# creating random number with max number as 5 and size=15

a = np.random.randint(5,size=15)

print a

# counting the most frequent element

counts = np.bincount(a)

print "the most frequent number is :" +str(np.argmax(counts)

## Limitations:

* For problem 3 we used many classes which is confusing. Instead we could have used 2 classes bank and customer class and implement the program.
* Using super class in multiple inheritance gives confusing regarding Python method resolution order (MRO). We can avoid  
  this.

## References: <https://stackoverflow.com/>