

**COMPUTER PROJECT**

Sandra Anto Vazhappilly

**ACKNOWLEDGEMENTS**

In the accomplishment of the completion of this project successfully, many people have bestowed upon me their blessings and their support. Therefore, I utilize this space to thank all the people who have been concerned with this project.

Primarily, I would like to thank the Almighty for allowing me to work on this project and complete it successfully. I would also like to thank my Computer teacher Mrs. SHEEBA ANSAR for her valuable guidance, suggestions and instructions throughout the course of this project.

My team in this project, ANKITA MENON, DEVIKA SUNIL, KARTHIKA PILLAI, and ANN KANNATH TONY have been of immense help to me. Their support and patience have served as major contributors towards the completion of this project and I would like to extend my heartfelt gratitude to them.

I would also like to thank my parents and classmates whose suggestions and support have helped and motivated me in various phases of this project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| Serial No | Contents | Page |
| 1 | PROJECT OVERVIEW | 5 |
| 2 | SYSTEM SPECIFICATION | 6 |
| 3 | DATA FILE DESIGN | 6 |
| 4 | ALGORITHM | 9 |
| 5 | PROGRAM CODE | 26 |
| 6 | OUTPUT | 99 |

**PROJECT OVERVIEW**

Today’s world is astoundingly dependent on computers. Many companies have a large amount of accounting and calculating to a daily basis. The larger the empire, the more difficult it is to keep track. Therefore, it is safe to say that most companies use computers to do the cercal work for them, guaranteeing accuracy and legibility.

Our project brings to sight an overview of an average working system of a Video library/CD shop. The code has been divided into 3 parts, Master Cassettes, Stock Cassettes and Customer Sales.

The code aims to help the manager of the shop to keep track of the CDs he usually sells, the CDs/Cassettes he has in stock using master and stock option.

In the Customer Sales code, he can cover a record of the purchase made by the customer, view a monthly sales report check the monthly sales report for each CD/Cassette and sort his customers mobile number wise.

We aim to show through our project the proceedings of a small shop and thereby, the intensity of the actual companies battling it out in today’s world.

***“Our first computers were born not out of greed or ego, but in the revolutionary spirit of helping common people rise above the most powerful institutions.”***

****

**SYSTEM SPECIFICATION**

1. Hardware Requirement

Processor: Pentium processor with windows XP/7/8

Memory: 1 GB

Disk Capacity: <1 GB

1. Software Requirement

Python 2.7

**DATA FILE DESIGN**

Data design is the first of the three design activities that are conducted during system development. The impact of data structure and procedural complexity leads to data design to have a profound influence on system quality.

All the data objects required by this system are listed below:

1. Master.dat

This database stores cassettes/CD data information for one time.

* Cast\_Code –Cassette/CD code - (1, 2, 3, etc.)
* Cast\_Name –Title of the Cassettes/CD
* Cast\_ Comp – Cassettes/CD company
* Cast\_Price – Price per Cassettes/CD

1. Cassettes/CD

The cassette.dat database is to be created on the basis of information collected from the Cassettes/CD company. The file receives multiple entries of same cassette code. It should contain information related to:

* Cast\_Code – Cassette/CD code - (1, 2, 3, etc.)
* Tot\_Cast – Total number of Cassettes/CD purchased
* Dd, mm, yy – Date of purchase

1. Balance.dat

* Cast\_Code – Cassette/CD code - (1, 2, 3, etc.)
* Cast\_Bal – Total number of Cassettes/CDs in balance
* Cast\_Price – Price per Cassette/CD
* dd, mm, yy – Balance date

1. Customer.dat

* Cast\_Code – Cassette/CD code – (1, 2, 3, etc.)
* C\_Name – Cassettes /CD purchased Customer’s name
* C\_Address – Customer’s address
* C\_MPhone – Customer’s Mobile number
* No\_of\_Cast – Number of Cassette/CDs purchased
* Dd, mm, yy – Casette/CD purchase date

The above data can be designed in the form of a relational database with course name as the key field. Also, it can be altered or modified according to Special need.

**MAIN ALGORITHM**

**Step 1:** Import os and, from pickle, import load,dump.

**Step 2:** Import datetime and import string.

**Step 3:** Store MFile as “Master.dat”,File1 as “Cassettes.dat”,File2 as “Balance.dat”,File3 as “Customer.dat” and Cdate as datetime.datetime.now()

**Step 4:** Create a class called Class\_Date.

**Step 5:** Initialize self.dd=Cdate.day,self.mm=Cdate.month,self.yy=Cdate.year

**Step 6:** Create class called Master.

**Step 7:** Create class Cassette.

**Step 8:** Create class Balance.

**Step 9:** Create class Customer.

**Step 10:** Create function to set the date as DD-MM-YYYY

**Step 11:** Create function to find a character month on against a month no.

**Step 12:** Set opt=’’,M=Master(),CS=Cassettes(),BL=Balance(),Cust=Customer()

**Step 13:** Print Menu as shown below:

**Video Library Main Menu**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 - >Master Cassettes/CDs

2 - >Stock Cassettes/CDs

3 ->Customer Sales

4 - >Exit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 14:** Get the choice from user and store it in opt.

**Step 15:** If opt=1 go to **Step**  16, if opt=2 go to **Step**  24,if opt=3, go to **Step**  35 if opt=4, go to **Step**  47 else go to **Step**  48.

**Step 16:** Take the user to Master Cassette Menu.

**Step 17:** Print Master Cassette Menu as shown below:

**Master Cassette Menu**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 - >Cassettes/CD stock Entry

2 - >View Cassettes/CDs

3 - >Back to main menu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 18:** Get the choice from the user and store it in ch.

**Step 19:** if ch=1 go to **Step**  20,if ch=2,go to **Step**  22, if ch=3, go to **Step**  13.

**Step 20:** M.Master\_Entry()

**Step 21:** Go to **Step**  17

**Step 22:** M.Master\_Display()

**Step 23:** Go to **Step**  17

**Step 24:** Take the user to Stock Cassettes Menu

**Step 25:** Print menu as shown below:

**Stock Cassette Menu**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 - >Cassette/CD Stock Entry

2 - >Display Cassette/CD

3 - >Stock/Balance Cassettes

4 - >Back to main menu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 26:** Get the choice from user and store it in ch.

**Step 27:** if ch=1, go to **Step**  28,if ch=2,go to **Step**  30,if ch=3 go to **Step**  32 else if ch=4 go to **Step**  13.

**Step 28:** CS.New\_Cassettes()

**Step 29:** Go to **Step**  25

**Step 30:** CS.Display\_Cassettes()

**Step 31:** Go to **Step**  25

**Step 32:** BL.Balance\_Cassettes()

**Step 34:** Go to **Step**  25

**Step 35:** Take the user to customer details.

**Step 36:** Print Customer Sales menu as shown below:

**Customer Sales Menu**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 - >Sales Entry

2 - >Monthly Sales Report

3 - >Code wise monthly sales

4 - >Customer mobile number wise

5 - >Back to main menu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 37:** Get the choice from user and store it in ch.

**Step 38:** if ch=1, go to **Step**  39, if ch=2, go to **Step**  41, if ch=3, go to **Step**  43, if ch=4 go to **Step**  45,else if ch=5 go to **Step**  13.

**Step 39:** Cust.Cassette\_Sale()

**Step 40:** Go to **Step**  36.

**Step 41:** Cust.MonthlySales\_Report()

**Step 42:** Go to **Step**  36

**Step 43:** Cust.CodeWiseMonthlySales\_Report()

**Step 44:** Go to **Step**  36

**Step 45:** Cust.CustomerWithMobileSearch()

**Step 46:** Go to **Step**  36.

**Step 47:** Print “Thank you and have a nice day” and go to **Step**  49.

**Step 48:** Print “Please enter the correct option” and go to **Step**  13.

**Step 49:** Stop.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLASS MASTER**

**Step 1:** Create master Class

**Step 2:** Initialize the following values:

self.Cast\_Code=0

self.Cast\_Name=’’

self.Cast\_Comp=’’

self.Cast\_Price=0

**Step 3:** create function Check\_Code and pass self and c code as arguments and create the following variables

MList=list()

TRec=list()

**Step 4:** open “Master.dat”, as Mobj in read mode

**Step 5:** extract Master.dat records

Check if entered c\_code is equal to the code of the first record set trec=Mrec then append the first record in the list.

In the list if code matches the record, break from loop and close mobj return Flag=true and Trec

**Step 6:** create function Master\_Entry and pass argument as self and store variable TRec=list()

Ch=’Y’

Print Add Master Cassette

**Step 7:** while ch=’Y’

Get cassette or CD code from the user and store it in variable self.Cast\_Code

**Step 8:** while len(str(self.Cast\_Code))<6 or len(str(self.Cast\_Code))>6:

Print “Please enter correct code!” and get the correct value from the user again.

**Step 9:** while True store Flag,TRec = self.Check\_Code(self.Cast\_Code)

**Step 10:** If Flag=False then get the Cassette/CD name, Company name, Individual Cassette/CD price from user and store it in self.Cast\_Name,self.Cast\_Comp,self.Cast\_Price

**Step 11:** Store N,C,P=1,1,1

**Step**  12: check if self.Cast\_Name==0 or len(self.Cast\_Name)>25.

If true, then store N=0.

**Step 13:** Check if self.Cast\_Comp==0 or len(self.Cast\_Comp)>25.

If true, then store C=0.

**Step 14:** Check if (self.Cast\_Price<=0):

If true then store P=0

**Step 15:** (while N=0 or C=0 or P=0) if any of these conditions are true,then check if self.Cast\_Name=0 or len(self.Cast\_Name)>25

**Step 16:** if condition is true then print “cassette/CD Name should not be greater than 25” and get Cassette/CD Name from user again and store N=1

**Step 17:**else check if self.Cast\_Comp=0 or len(self.Cast\_Comp)>25

**Step 18:** if condition is true, then print “Company name should not be greater than 25” and get Company Name from the user again and store C=1.

**Step 19:** else check self.Cast\_Price<=0

**Step 20:** if condition is true, then print Enter valid price for Cassette/CD and get the individual Cassette/CD price from user and store P=1.

**Step 21:** else if the condition is false open MFile in append mode as Mobj

**Step 22:** if it is not mobjthen print MFile,”is not created” else append data into a sequence object.

**Step 23:** else print “Code”,self.Cast\_Code,”is already in ‘Master.dat’ file”

**Step 24:** create another function Master\_Display passing self as variable

**Step 25:** if file is not MFile then print MFile,”the file does not exist”

**Step 26:** else open MFile in read mode and store it in Mobj

**Step 27:** print “Cassette/CD Master Report”

Print “=”\*100

Print “code”,”\tCassette/CD Name”,(“ ”\*(25-len(“Cassette/CD Name”))),”\tCompany Name”,(“ “\*(25-len(“Company Name”))),”\t\tPrice”

Print “=”\*100

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLASS CASSETTES**

**Step 1:** Create Class Cassettes

**Step 2:**initialize the following:

self.Cast\_Code = 0

self.Tot\_Cast = 0

self.dd = self. mm = self.yy = 0

**Step 3:** Create function New\_Cassettes

**Step 4:** Initialize the following:

M = Master()

B = Balance()

CDt =Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy =CDt.yy

**Step 5:** print “Add new stock cassettes/CD”

**Step 6:** Initialize ch as ‘Y’

**Step 7:** While ch = ‘Y’

Assign TRec as list() to store master record, and assign Flag as False to if Cast\_Code is in Master.dat or not

**Step 8:** Print “Date:%s-%s-%s”%(CDt.dd,Cdt.mm,CDt.yy)”

**Step 9:** While True:

self.Cast\_Code = int(input(“Cassette/CD Code:”))

For function call to check cassette/CD code in Master.dat

**Step 10**: assign Flag, TRec as M.Check\_Code(self.Cast\_Code)

**Step 11:** if flag==True:

self.Cast\_Name as TRec[1] for Title of the Cassette/CD

self.Cast\_Comp as TRec[2] for cassette/CD company

self.Cast\_Price as TRec[3] for cassette/CD price

**Step 12:** print “Cassette/Cd Name:”,self.Cast\_Name

print “Company Name:”,self.Cast\_Comp

print “Individual cassette/CD price:”,self.Cast\_Price

**Step 13:** while true

ask for input for self.Tot\_Cast

**Step**  14:

if self.Tot\_Cast is less than or equal to 0

then,print “Enter valid cassette/CD number”

else break from loop in **Step**  13

**Step 15:** ch=input to save record or not

if ch=’y’ or ch=’Y’

initialize CList as list()

and open File1 in append mode as Cobj

if not Cobj print File1 is not created

else append these data into a sequence object

(self.Cast\_Code)

(self.Tot\_Cast)

(self.dd)

(self.mm)

(self.yy)

**Step 16:** then dump into binary file

and print record saved

**Step 17:** ask for input to store more record or not

if yes continue

else break

**Step 18:** create a user defined function Master\_Display

**Step 19**: if not os.path.isfile(MFile) then,

print (MFile,”file does not exist”)

else assign

Mobj as open(MFile,’rb’)

print ”\nCassette/CD Master Report”

print ”=” \*70

print “Code”,”\t\tCassette/CD Name”,”\t\t\tCompany Name”,”\t\t\tPrice”

print ”=”\*120

**Step 20:** try:

while True assign

MRec = []

MRec as load(Mobj)

then print str(MRec[0]),”\t\t”,MRec[1],(“ “\*(25-len(MRec[1]))),”\t\t”,MRec[2],(“ “\*(25-len(MRec[2]))),”\t\t”,MRec[3]

except EOFError

print (“-“ \*120)

Mobj.close()

**CLASS BALANCE**

**Step 1**: create a class called Balance

**Step 2**: Initialize self.Cast\_Code = 0,self.Cast\_Bal = 0 ,self.Cast\_Price = 0,

self.dd = self.mm = self.yy = 0

**Step 3**: create a function called Give\_Balance and pass arguments self and C\_Code

**Step 4**: create a variable Tbalance = 0

**Step 5**: When File2 does not exit return False

**Step 6**: else create a list to extract record from Balance.dat and store it in Brec

**Step 7**: create variable Tbalance = 0

**Step 8**: open file File2 in read mode and store in Bobj

**Step 9**: Create function AddUpdateBalance and pass arguments self, Clist, CPrice

**Step 10**: create variable Cbalance = Balance.Give\_Balance(self,CList[0])

**Step 11**: if Cbalance == False add record for first time in Brec

**Step 12**: else if Cbalance >= 0

**Step 13**: open File2 in read mode and store it as Bobj

Open Temp.dat in write mode and store it as Bobj

**Step 14**: while True create a list to extract record from Balance.dat and store in Brec

**Step 15**: if CList[0]!= Brec[0].Write data in Tobj file

**Step 16**: else BRec[1] = Cbalance + CList[1]. Write data in Tobj file

**Step 17**: close files Tobj, Bobj

**Step 18**: remove Balance.dat

**Step 19**: rename Temp.dat as Balance.dat

**Step 20**: create function UpdateBalance and pass self, CList argument

**Step 21:** open (File2,’rb’) and store it as bobj and open Temp.dat in write mode and store it as Tobj

**Step 22**: while True create a list to extract record from Balance.dat and store in Brec

**Step 23**: if CList[0]!=BRec[0]. Write data in Tobj

**Step 24**: else BRec[1] = BRec[1] – Clist[4]

**Step 25**: close file Tobj and close file Bobj

**Step 26**: remove Balance.dat

**Step 27**: rename Temp.dat as Balance.dat

**Step 28**: print “Balance.dat updated”

**Step**  **29**: create another function Balance\_Cassettes and pass self as argument

**Step**  **30**: create object M = Master()

**Step**  **31**: if not File2 then print file does not exist **Step**  : else create variable TAmount = 0 and print Balance Stock Register (Cassette/CD), print “=” \*105

**Step**  **32**: Bobj = open File2in read mode and store in Bobj

**Step**  **33:** then print ”Code”,”\tCassette/CD Name”,(“ “\*(25-len(“Cassette/CD Name”))),”\tCompany Name”,(“ “\*(25-len(“Company Name”))), ”\tQuantity”,”\tPrice”,”\tAmount” print “-“ \*105

**Step**  **34**: while True create variable, BRec=[], BRec = load(Bobj),TRec = list(),Flag ,TRec = M.Check\_Code(BRec[0])

Step 35: if Flag is true then store Amount = =BRec[1] \* BRec[2],TAmount = TAmount + Amount and print BRec[0],”\t”, TRec[1],(“ “\*(25-len(TRec[1]))),”\t”, TRec[2],(“”\*(25-len(TRec[2]))),”\t”, BRec[1],”\t\t”, BRec[2],”\t”, Amount

Print “-“\*105 print”%s Total Amount:%s %.2f” % (‘’\*56,’’\*4,TAmount)

**Step**  **36**: close Bobj

**CLASS CUSTOMER**

**Step 1**: create class Customer

**Step**  **2**: initialize

self.Cast\_Code = 0 #cassette/CD code

self.C\_Name =” “ #customer name

self.C\_Address =” “ #Customer Address

self.C\_MPhone =0 #Customer mobile no

self.No\_Of\_Cast =0 #Number of Cassette/CD

self.dd = self.mm = self.yy = 0 #Sale date

**Step**  **3**: create a user defined function named Cassette\_Sale

**Step**  **4**: assign

M = Master()

B = Balance()

CDt = Cast\_Date()

self.dd = CDt.dd

self.mm = CDt.mm

self.yy = CDt.yy

Cbalance = 0

**Step**  **5:** print “Customer sales cassette/CD”

**Step 6**: initialize ch as ‘y’

**Step**  **7:** while ch == ’y’ assign TRec as list() for a temporary list to store master record

Flag as False, To check if Cast\_Code is in Master.dat or not

**Step 8**: print Date: %s-%s-%s” % (CDt.dd, CDt.mm, CDt.yy)

**Step 9**: self.Cast\_Code = int(input(“\nCassette/CD Code (Enter 6 digit code): “))

**Step 10:** assign Flag, TRec as M.Check\_Code(self.Cast\_Code)

and Cbalance as B.Give\_Balance(self.Cast\_Code)

**Step 11:** if flag == True

assign

self.Cast\_Name as TRec[1] ,Title of the cassette/CD

self.Cast\_Comp as TRec[2] ,cassette/CD company

self.Cast\_Price as TRec[3] , Price per cassette/Cd

**Step 12**: print “Cassette/CD Name:” ,self.Cast\_Name

print “Company Name :”,self.Cast\_Comp

print “Individual Cassette/CD price :”,self.Cast\_Price

print “\nEnter Customer Details”

**Step**  **13**: input the following

self.C\_Name

self.C\_Address

self.C\_MPhone

**Step**  **14**: input self.No\_Of\_Cast

**Step**  **15**: if (self.No\_Of\_Cast>Cbalance) then

print “Out of stock”

else break

**Step**  **16:** if the program has to continue

initialize CList and assign it as list()

**Step**  **17**: open File 1 in append mode as Cobj

**Step 18**: if not print File 1 is not created

**Step**  **19**: else append these into sequence object

CList.append(self.Cast\_Code)

CList.append(self.Tot\_Cast)

CList.append(self.dd)

CList.append(self.mm)

CList.append(self.yy)

**Step**  **20**: Dump (CList, Cobj)

**Step**   **21**: B.Add\_to\_File(self.Cast\_Code, self.Tot\_Cast, self.Cast\_Price, self.dd, self.mm, self.yy)

**Step**  **22**: print record saved

**Step**  **23**: create a user defined function Display\_Cassettes

**Step**  **24**: assign M = Master()

if not os.path.isfile(File1) then,

print (File1,”file does not exist”)

else

open Cobj in read mode

**Step**  **25**: print “\nCassette/CD entry Register”

print “=’ \*125

print “Code”,”\tCassette/CD Name”,(“ “\*(25-len(“Cassette/CD Name”))), “\tCompany Name”,(“ “\*(25-len(“Company Name”))),”\t\tQuantity”,”\tPrice”,”\t\\tDate”

print “-“ \* 125

**Step**  **26**: try:

while True:

assign CRec as[]

CRec as load(Cobj)

TRec aslist()

Flag, TRec as M.Check\_Code(CRec[0])

nDt as Set\_DateFormat(CRec[2], CRec[3], CRec[4])

**Step**   **27**: if (Flag is equal to True):

print CRec[0], “\t”, TRec[1], (“ “\*(25-len(TRec[1]))),”\t\”,TRec[2],(“ “\*(25-len(TRec[2]))), “\t\t”,CRec[1],”\t\t”, TRec[3], “\t\t”,nDt

refer Step 26

except EOFError and pass

print “-“ \*125

Cobj.close()

**PROGRAM CODE**

print'\*'\*54,"WELCOME TO BLUE RAY DIGITAL CASSETTES AND CD EXHIBITION","\*"\*54

print'\*'\*64,"CUSTOMER SATISFACTION IS OUR GOAL",'\*'\*64

import os

from pickle import load, dump

import datetime

import string

MFile ="Master.dat"

File1 = "cassettes.dat"

File2 = "Balance.dat"

File3 ="Customer.dat"

Cdate = datetime.datetime.now()                             #  Current date and time

                                                            # Class for date

class  Cast\_Date :

    def \_\_init\_\_(self):

        self.dd = Cdate.day

[self.mm](http://self.mm/) = Cdate.month

        self.yy = Cdate.year

class Master:

    def \_\_init\_\_(self):

        self.Cast\_Code = 0                              # cassette/CD code -(Like, 1, 2, 3 etc.)

        self.Cast\_Name = ""                             # Title of the cassette/CD

        self.Cast\_Comp = ""                             # cassette/CD company

        self.Cast\_Price = 0                             # Price per cassette/CD

    def Check\_Code(self, C\_Code):

        MList = list()

        TRec = list()

        Flag = False         # To check if Cast\_Code is in master.dat or not

        if os.path.isfile("Master.dat"):

            Mobj = open("Master.dat", 'rb')

            try:

                while True:

                    MRec = []# For extracting master.dat records

                    MRec = load(Mobj)

                    if (C\_Code == MRec[0]):

                        TRec = MRec

                    MList.append(MRec[0])

            except EOFError:

                pass

            for i in range(len(MList)):

                if (C\_Code == MList[i]):

                    Flag = True

                    break

            Mobj.close()

         # Flag for Master data entry and TRec for Cassette data entry

        return Flag, TRec

    # For Master data entry

    def Master\_Entry(self):

        TRec = list() # A temporary list to store master record

        print("\nAdd Master Cassette/CD")

        ch='Y'

        while ch=='Y':

            self.Cast\_Code = int(input("\nCassette/CD Code [Enter 6 digit code]: "))

            while len(str(self.Cast\_Code))<6 or len(str(self.Cast\_Code))>6:

                print "Please Enter correct code!"

                self.Cast\_Code = int(input("Cassette/CD Code [Enter 6 digit code ]: "))

            Flag, TRec = self.Check\_Code(self.Cast\_Code)

            if (Flag == False):

                ch=input("Enter the number of entries to be made:")

                for i in range (ch):

                    self.Cast\_Name = raw\_input("Cassette/CD Name :")

                    self.Cast\_Comp = raw\_input("Company Name:")

                    self.Cast\_Price = float(input("Individual Cassette/CD price :"))

                    N,C,P=1,1,1

                    if (self.Cast\_Name == 0 or len(self.Cast\_Name) > 25):

                        N=0

                    if (self.Cast\_Comp == 0 or len(self.Cast\_Comp) > 25):

                        C=0

                    if (self.Cast\_Price <= 0):

                        P=0

                    while N==0 or C==0 or P==0:

                        if (self.Cast\_Name == 0 or len(self.Cast\_Name) > 25):

                            print("Cassette/CD Name should not greater than 25")

                            self.Cast\_Name = raw\_input("Cassette/CD Name :")

                            N=1

                        elif (self.Cast\_Comp ==0 or len(self.Cast\_Comp)  > 25):

                            print("Company Name should not greater than 25")

                            self.Cast\_Comp = raw\_input("Company Name:")

                            C=1

                        elif (self.Cast\_Price <= 0):

                                print("Enter valid price for Cassette/CD")

                                self.Cast\_Price = float(input("Individual Cassette/CD price:"))

                                P=1

                    else:

                        with open(MFile, 'ab+') as Mobj:

                            if not Mobj:

                                print (MFile, "is not created")

                            else:

                                # Appends data into a sequence object

                                MList = list()

                                MList.append(self.Cast\_Code)

                                MList.append(self.Cast\_Name)

                                MList.append(self.Cast\_Comp)

                                MList.append(self.Cast\_Price)

                               # Write data into binary file

                                dump(MList, Mobj)

            else:

                print ("\ncode", self.Cast\_Code,"is already in 'Master.dat' file")

                ch = raw\_input("\n\nAdd new Cassette/CD code? <Y?N>: ")

                ch = ch.upper()

                if ch=='Y':

                    self.Cast\_Code =int(input("\nCassette/CD Code [Enter 6 digit code]: "))

                    continue

                else:

                    break

    def Master\_Display(self):

        if not os.path.isfile(MFile):

            print (MFile, "file does not exist")

        else:

            Mobj = open(MFile, 'rb')

            print "\nCassette/CD Master Report"

            print "=" \* 100

            print " Code", "\tCassette/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tPrice"

            print "-" \* 100

            try:

                while True:

                    MRec = []

                    MRec =load(Mobj)

                    print str(MRec[0]), "\t",MRec[1],(" "\*(25-len(MRec[1]))),"\t", MRec[2],(" "\*(25-len(MRec[2]))), "\t\t",MRec[3]

            except EOFError:

                pass

            print "-" \* 100

            Mobj.close()

class Cassettes:

    def \_\_init\_\_(self):

        self.Cast\_Code = 0

        self.Tot\_Cast = 0           # Total cassette/CD purchased

        self.dd = [self.mm](http://self.mm/) = self.yy = 0  # Cassette/CD purchase date

            # For cassette/CDs entry into the cassettes.dat data file

    def New\_Cassettes(self):

        M = Master()

        B = Balance()

        CDt = Cast\_Date()

        self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

        self.yy = CDt.yy

        TRec = list() #A temporary list to store master record

        print("Add New Stock cassette/CD");

        ch = 'Y'

        while ch == 'Y':

            Flag = False   # To check if Cast\_Code is in Master.dat or not

            print("\n Date: %s-%s-%s" %  (CDt.dd,CDt.mm, CDt.yy))

            while True:

                self.Cast\_Code = int(input("\nCassette/CD Code :"))

                   # Function call to check cassette/CD code in Master.dat

                Flag,TRec = M.Check\_Code(self.Cast\_Code)

                if (Flag == True):

                    self.Cast\_Name = TRec[1]       #title of the cassette/CD

                    self.Cast\_Comp = TRec[2]       #cassette/Cd company

                    self.Cast\_Price = TRec[3]      #Price per cassette/CD

                    print"Cassette/CD Name :",self.Cast\_Name

                    print"Company Name : ",self.Cast\_Comp

                    print"Individual Cassette/CD price: ",self.Cast\_Price

                    while True:

                        self.Tot\_Cast = int(input("Enter no of stock cassettes/CDs purchased :"))

                        if (self.Tot\_Cast <= 0):

                            print"Enter valid Cassette/Cd number"

                        else:

                            break

                    ch = raw\_input("\n Do you want to save the record <Y/N>: ")

                    if ch == 'y' or ch== 'Y':

                        CList = list()

                        with open(File1, 'ab') as Cobj:

                            if not Cobj:

                                print File1, "is not created"

                            else:

                                  # Appends data into a sequnce object

                                CList.append(self.Cast\_Code)

                                CList.append(self.Tot\_Cast)

                                CList.append(self.dd)

                                CList.append([self.mm](http://self.mm/))

                                CList.append(self.yy)

                                  #write data into the binary files

                                dump(CList, Cobj)   #B.Add\_to\_File(self.Cast\_Code, self.Tot\_Cast,self.Cast\_Price, self.dd, [self.mm](http://self.mm/), self.yy)

                                B.AddUpdateBalance(CList, self.Cast\_Price)

                                print"Record saved"

                            ch =raw\_input("\nStock more cassette/Cd record? <Y/N>: ")

                            ch = ch.upper()

                            if ch =='Y':

                                continue

                            else:

                                break

                else:

                        print "Please enter existing cassette no."

               # For cassettes/CDs entry into the cassettes.dat data file

    def SetDateFormat(self,d1, m1, y1):

        fDt = ''

        d11 = str(d1)

        m11 = str(m1)

        y11 = str(y1)

        if (len(d11)==1):

            d11 = '0'+d11

        if (len(m11)==1):

            m11 = '0'+m11

        fDt = d11+'-'+m11+'-'+y11

        return fDt

    def Display\_Cassettes(self):

        M = Master()

        if not os.path.isfile(File1):

            print (File1, "file does not exist ")

        else:

            Cobj = open(File1, 'rb')

            print "\nCassette/CD entry Register"

            print "=" \* 125

            print " Code", "\tCassettes/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tQuantity","\t\tPrice","\t\tDate"

            print "-" \* 125

            try:

                while True:

                    CRec = []

                    CRec = load(Cobj)

                    TRec = list()

                    Flag, TRec = M.Check\_Code(CRec[0])

                    nDt =self.SetDateFormat(CRec[2],CRec[3], CRec[4])

                    if (Flag == True):

                        print CRec[0],"\t", TRec[1],(" "\*(25-len(TRec[1]))),"\t",TRec[2],(" "\*(25-len(TRec[2]))), "\t\t  ",CRec[1],"\t\t      ", TRec[3],"\t\t",nDt

            except EOFError:

                pass

            print "-" \* 125

            Cobj.close()

              # Function to set the data as: DD-MM-YYYY

class Balance:

    def \_\_init\_\_(self):

                  # Instance attributes of Balance.dat data file

        self.Cast\_Code = 0           # cassette/Cd code to be balance

        self.Cast\_Price = 0          # Total numbert of cassettes/CDs in balance

        self.Cast\_Price = 0          #Unit price of cassettes/CDs on code wise

        self.dd = [self.mm](http://self.mm/) = self.yy = 0  # Balance date

    def Give\_Balance(self, C\_Code):

        Tbalance = 0

        if not os.path.isfile(File2):

                       # When file does not exist

            return False

        else:

            Brec = list()  # A list to extract record from Balance.dat

            Tbalance = 0

            Bobj = open(File2,'rb')

            try:

                while True:

                    BRec = load(Bobj)

                    if (C\_Code == BRec[0]):

                        Tbalance = BRec[1]   #E.g. Cast\_Bal

                        break

            except EOFError:

                    pass

            Bobj.close()

            return Tbalance

    def AddUpdateBalance(self, CList, CPrice):

                   # To know the balance cassette in 'Balance.daat'

        Cbalance = Balance.Give\_Balance(self, CList[0])

        if (Cbalance == False): # If file does not exist , add the record for firsttime

            BRec = list()

            with open(File2, 'ab') as Bobj:

                BRec.append(CList[0]) #Cast\_Code

                BRec.append(CList[1]) #Cast\_Bal

                BRec.append(CPrice) #Cast\_Price

                BRec.append(CList[2]) # Day

                BRec.append(CList[3]) # Month

                BRec.append(CList[4]) # Year

                dump(BRec, Bobj)

        elif (Cbalance >= 0):

            Bobj = open(File2, 'rb')

            Tobj = open("Temp.dat", 'wb')

            try:

                while True:

                    BRec = list()         # A list to extract record from Balance.dat

                    BRec = load(Bobj)

                    if (CList[0] != BRec[0]):

                                  # Write data into Temp.dat file

                        dump(BRec,Tobj)

                    else:

                        BRec[1] = Cbalance + CList[1]

                                  #self.Cast\_Bal = self.Cast\_Bal + Cbalance

                        dump(BRec, Tobj)

            except EOFError:

                pass

            Tobj.close()

            Bobj.close()

            os.remove("Balance.dat")

            os.rename("Temp.dat", "Balance.dat")

    def UpdateBalance(self, CList):

        Bobj = open(File2, 'rb')

        Tobj = open("Temp.dat", 'wb')

        try:

            while True:

                BRec = load(Bobj)

                if (CList[0] != BRec[0]):

                           # Write data into Temp.dat file

                    dump (BRec, Tobj)

                else:

                    BRec[1] = BRec[1] - CList[4]

                    dump(BRec, Tobj)

        except EOFError:

            pass

        Tobj.close()

        Bobj.close()

        os.remove("Balance.dat")

        os.rename("Temp.dat", "Balance.dat")

        print'Balance.dat updated'

    def Balance\_Cassettes(self):

        M = Master()

        if not os.path.isfile(File2):

            print File2, "file does not exist"

        else:

            TAmount = 0

            print "\nBalance Stock Register(Cassette/CD) "

            print "=" \* 128

            Bobj = open(File2, 'rb')

            print " Code","\tCassette/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tQuantity","\t\tPrice", "\t\tAmount"

            print "-" \* 128

            try:

                while True:

                    BRec = []

                    BRec = load(Bobj)

                    TRec = list()

                    Flag, TRec = M.Check\_Code(BRec[0])

                    if (Flag == True):

                        Amount = BRec[1] \* BRec[2]

                        TAmount = TAmount + Amount

                        print BRec[0],"\t", TRec[1], (" "\*(25-len(TRec[1]))),"\t",TRec[2],(" "\*(25-len(TRec[2]))),"\t\t  ",BRec[1],"\t\t\t", BRec[2],"\t\t",Amount

            except EOFError:

                pass

            print "-" \* 128

            print "%s Total Amount: %s %.2f" % (' ' \*56,'' \* 4 ,TAmount)

            Bobj.close()

class Customer:

    def \_\_init\_\_(self):

                               # Instance attributes of Customer.dat data file

        self.Cast\_Code = 0    # cassette/CD code

        self.C\_Name = ''      # Customer name

        self.C\_Addresss = ''  # Customer address

        self.C\_MPhone = 0     # Customer mobile no.

        self.No\_Of\_Cast = 0   # Number of Cassette/Cd

        self.dd = [self.mm](http://self.mm/) = self.yy = 0 # sale date

    def Cassette\_Sale(self):

        M =Master()

        B = Balance()

        CDt = Cast\_Date()

        self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

        self.yy = CDt.yy

        Cbalance = 0

        print"Customer sales cassette/CD"

        ch ='Y'

        while ch=='Y':

            TRec = list() # A temporary list to store master record

            Flag = False  # To check if Cast\_Code is in Master.dat or not

            print "Date: %s-%s-%s" % (CDt.dd, CDt.mm,CDt.yy)

            while True:

                self.Cast\_Code = int(input("\nCassette/CD Code (Enter 6 digit code): "))

                       # Function call to check cassette/Cd code in Master.dat

                Flag, TRec = M.Check\_Code(self.Cast\_Code)

                Cbalance = B.Give\_Balance(self.Cast\_Code)

                if (Flag == True):

                    self.Cast\_Name = TRec[1]   #Title of the cassette/CD

                    self.Cast\_Comp = TRec[2]   #cassette/CD company

                    self.Cast\_Price = TRec[3]   #Price per cassette/CD company

                    print "Cassette/CD Name :", self.Cast\_Name

                    print "Company Name :", self.Cast\_Comp

                    print "Individual Cassette/CD price : ",self.Cast\_Price

                    print "\nEnter Customer details",

                    self.C\_Name = raw\_input("Customer name: ").upper()

                    self.C\_Address = raw\_input("Custoer address: ")

                    self.C\_MPhone = raw\_input("Customer mobile no: ")

                    while True:

                        self.No\_Of\_Cast = int(input("\nEnter sales cassettes/CDs nos.: "))

                        if (self.No\_Of\_Cast > Cbalance):

                            print "Out of Stock"

                        else:

                            break

                    ch = raw\_input("\nSales confirm <Y/N>: ").upper()

                    if ch == 'Y':

                        CustList = list()

                        with open(File3, 'ab') as CustObj:

                            if not CustObj :

                                print File3, "is not Created"

                            else:

                                      # Appends data into a sequence object

                                CustList.append(self.Cast\_Code)

                                CustList.append(self.C\_Name)

                                CustList.append(self.C\_Address)

                                CustList.append(self.C\_MPhone)

                                CustList.append(self.No\_Of\_Cast)

                                CustList.append(self.dd)

                                CustList.append([self.mm](http://self.mm/))

                                CustList.append(self.yy)

                                B.UpdateBalance(CustList)

                                dump(CustList, CustObj)

                    ch=raw\_input("\nMore sale? <Y/N>:")

                    ch=ch.upper()

                    if ch!='Y':

                        break

            #Function to search individual customer on mobile no.

    def MonthlySales\_Report(self):

        M=Master()

        if not os.path.isfile(File2):

            print (File3, "file does not exist")

        else:

            monthNo = int(input('\nEnter month number:'))

            yearNo = int(input('\nEnter year:'))

            CDt = Cast\_Date()

            self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

            self.yy = CDt.yy

            D=Cassettes()

            if (monthNo <= 12 and monthNo <= [self.mm](http://self.mm/) and yearNo <= self.yy):

                MonthName = self.Month\_Name(monthNo)

                TAmount = 0

                        #Function called to set the date as DD-MM-YY

                nDt = D.SetDateFormat(self.dd, [self.mm](http://self.mm/), self.yy)

                print "\nCustomer Sales Status Report - Date:", nDt

                print "For the month of", MonthName, yearNo

                print "-" \* 160

                CustObj = open(File3, 'rb')

                print "Name","\t\t", "Mobile No.","\t\t" ,"Cassette/CD Cide and Name","\t\t", "Date","\t\t", "Qty", "\t\t", "Unit price", "\t\t","Amount"

                print "-"\*160

                try:

                    while True:

                        CustRec = []

                        CustRec = load(CustObj)

                        TRec = list()

                        Flag, TRec = M.Check\_Code(CustRec[0])

                        UPrice = TRec[3]

                        Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] #An additional 20% of Unit Price

                        nDt = D.SetDateFormat(CustRec[5], CustRec[6], CustRec[7])

                        TAmount+=Amount

                        if (monthNo == CustRec[6] and yearNo == CustRec[7]):

                            Clength = str(CustRec[0])+'-'+TRec[1]

                            nName = ''

                            for i in range(len(Clength)):   #Extracts only 24 characters

                                nName = nName + Clength[i]

                                if i == 23:

                                    break

                            print CustRec[1],"\t   ", CustRec[3],"\t\t        ", nName,"\t\t\t",nDt, "\t",CustRec[4], "\t\t",UPrice,"\t\t       ", Amount

                except EOFError:

                    pass

                    print "-" \* 160

                    print'Note. Amount is calculated as 20% extra on unit price.'

                    print ("%s Total Sales Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

                    CustObj.close()

            else:

                print "Month number and year is not valid"

                    #Function to display close wise monthly sales report.

    def CodeWiseMonthlySales\_Report(self):

        M=Master()

        D=Cassettes()

        TRec = list() #A temporary list to store master record

        Flag = False  #To check if Cast\_Code is in Master.dat or not

        if not os.path.isfile(File2):

            print (File3, "file does not exist")

        else:

            CCode = int(input("\nCassette/CD Code:"))

            monthNo = int(input('\nEnter month number:'))

            yearNo = int(input('\nEnter year:'))

            CDt = Cast\_Date()

            self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

            self.yy = CDt.yy

                    #Function to call to check cassette/CD code in Master.dat

            Flag, TRec = M.Check\_Code(CCode)

            if (monthNo <= 12 and  monthNo <= [self.mm](http://self.mm/) and yearNo <= self.yy and Flag == True):

                CName = TRec[1]  #Tittle of the casette/CD

                CComp = TRec[2]  #casette/CD company

                CPrice = TRec[3] #Price per cassette/CD

                        #Function Call for a character month

                MonthName = self.Month\_Name (monthNo)

                TAmount = 0

                        #Function Called to set date as DD-MM-YY

                nDt = D.SetDateFormat(self.dd, [self.mm](http://self.mm/), self.yy)

                print "\nCode wise Sales Report - Date:", nDt

                print "For the month of", MonthName, yearNo

                print "Cassette/CD Code:  %d  Name: %s" % (CCode, CName)

                print

                print "-" \* 120

                CustObj = open(File3, 'rb')

                print "Name","\t\t", "Mobile No.","\t\t" , "Date","\t\t", "Qty", "\t\t","\t\t", "Unit price", "\t\t","Amount"

                print "-" \* 120

                ctr = 0

                try:

                    while True:

                        CustRec = []

                        CustRec = load(CustObj)

                        TRec = list()

                        Flag, TRec = M.Check\_Code(CustRec[0])

                        UPrice = TRec[3]

                        Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] #An additional 20% of Unit Price

                        nDt = D.SetDateFormat(CustRec[5], CustRec[6], CustRec[7])

                        TAmount+=Amount

                        if (monthNo == CustRec[6] and yearNo == CustRec[7]) and (CCode == CustRec[0]):

                            ctr += 1

                            print CustRec[1],"\t\t\t", CustRec[3],"\t\t\t", nDt, CustRec[4],"\t\t\t", UPrice,"\t\t\t", Amount

                except EOFError:

                    pass

                print "-" \* 120

                if (ctr == 0):

                    print 'No record found on such Code No. , Month and Year'

                else:

                    print'Note. Amount is calculated as 20% extra on unit price.'

                    print ("%s Total Sales Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

                CustObj.close()

            else:

                print "Either Code not found or Month no. and year is not valid"

                 #Function to search individual customer on mobile no.

    def Month\_Name(self,mNo):

        self.mDict = {1:'January', 2:'February', 3:'March', 4:'April', 5:'May', 6:'June', 7:'July', 8:'August', 9:'September', 10:'October', 11:'November', 12:'December'}

        self.nName = ''

        for key, value in self.mDict.items():

            if (key == mNo):

                self.nName = value

                break

        return self.nName

while True:

    M=Master()

    CS=Cassettes()

    BL=Balance()

    Cust=Customer()

    print

    print "\n Video Library Main Menu"

    print "-" \* 50

    print "       1 - >  Master Cassettes/CDs"

    print "       2 - >  Stock Cassettes/CDs"

    print "       3 - >  Customer Sales"

    print "       4 - >  Exit"

    print "-" \* 50

    opt = input("Enter your choice(1,2,3 or 4):")

    if opt==1:

        ch=0

        while True:

            print

            print "\n\tMaster Cassettes Menu"

            print "-" \* 50

            print "       1 - >  Cassettes/CDs Stock Entry"

            print "       2 - >  View Cassettes/CDs"

            print "       3 - >  Back to Main Menu"

            print "-" \* 50

            ch = input("Enter your choice:")

            if ch==1:

                M.Master\_Entry()

            elif ch==2:

                M.Master\_Display()

            elif ch==3:

                break

    elif opt==2:

        ch=0

        while True:

            print

            print "\n\tStock Cassettes Menu"

            print "-" \* 50

            print "       1 - >  Cassettes/CDs Stock Entry"

            print "       2 - >  Display Cassettes/CDs"

            print "       3 - >  Stock/Balance Cassettes"

            print "       4 - >  Back to Main Menu"

            print "-" \* 50

            ch = input("Enter your choice:")

            if ch==1:

                CS.New\_Cassettes()

            elif ch==2:

                CS.Display\_Cassettes()

            elif ch==3:

                BL.Balance\_Cassettes()

            elif ch==4:

                break

    elif opt==3:

        ch=0

        while True:

            print

            print "\n\tCustomer Sales Menu"

            print "-" \* 50

            print "       1 - >  Sales Entry"

            print "       2 - >  Monthly Sales Report"

            print "       3 - >  Code Wise Monthly Sales"

            print "       4 - >  Back to Main Menu"

            print "-" \* 50

            ch = int(raw\_input("Enter your choice:"))

            if ch==1:

                Cust.Cassette\_Sale()

            elif ch==2:

                Cust.MonthlySales\_Report()

            elif ch==3:

                Cust.CodeWiseMonthlySales\_Report()

            elif ch==4:

                break

    elif opt==4:

        print "Thank you and have a nice day"

        break

    else:

        print "Please enter correct option!"

import os

from pickle import load, dump

import datetime

import string

MFile ="Master.dat"

File1 = "cassettes.dat"

File2 = "Balance.dat"

File3 ="Customer.dat"

Cdate = datetime.datetime.now()                             #  Current date and time

                                                            # Class for date

class  Cast\_Date :

    def \_\_init\_\_(self):

        self.dd = Cdate.day

[self.mm](http://self.mm/) = Cdate.month

        self.yy = Cdate.year

class Master:

    def \_\_init\_\_(self):

        self.Cast\_Code = 0                              # cassette/CD code -(Like, 1, 2, 3 etc.)

        self.Cast\_Name = ""                             # Title of the cassette/CD

        self.Cast\_Comp = ""                             # cassette/CD company

        self.Cast\_Price = 0                             # Price per cassette/CD

    def Check\_Code(self, C\_Code):

        MList = list()

        TRec = list()

        Flag = False         # To check if Cast\_Code is in master.dat or not

        if os.path.isfile("Master.dat"):

            Mobj = open("Master.dat", 'rb')

            try:

                while True:

                    MRec = []# For extracting master.dat records

                    MRec = load(Mobj)

                    if (C\_Code == MRec[0]):

                        TRec = MRec

                    MList.append(MRec[0])

            except EOFError:

                pass

            for i in range(len(MList)):

                if (C\_Code == MList[i]):

                    Flag = True

                    break

            Mobj.close()

         # Flag for Master data entry and TRec for Cassette data entry

        return Flag, TRec

    # For Master data entry

    def Master\_Entry(self):

        TRec = list() # A temporary list to store master record

        print("\nAdd Master Cassette/CD")

        ch='Y'

        while ch=='Y':

            self.Cast\_Code = int(input("\nCassette/CD Code [Enter 6 digit code]: "))

            while len(str(self.Cast\_Code))<6 or len(str(self.Cast\_Code))>6:

                print "Please Enter correct code!"

                self.Cast\_Code = int(input("Cassette/CD Code [Enter 6 digit code ]: "))

            Flag, TRec = self.Check\_Code(self.Cast\_Code)

            if (Flag == False):

                ch=input("Enter the number of entries to be made:")

                for i in range (ch):

                    self.Cast\_Name = raw\_input("Cassette/CD Name :")

                    self.Cast\_Comp = raw\_input("Company Name:")

                    self.Cast\_Price = float(input("Individual Cassette/CD price :"))

                    N,C,P=1,1,1

                    if (self.Cast\_Name == 0 or len(self.Cast\_Name) > 25):

                        N=0

                    if (self.Cast\_Comp == 0 or len(self.Cast\_Comp) > 25):

                        C=0

                    if (self.Cast\_Price <= 0):

                        P=0

                    while N==0 or C==0 or P==0:

                        if (self.Cast\_Name == 0 or len(self.Cast\_Name) > 25):

                            print("Cassette/CD Name should not greater than 25")

                            self.Cast\_Name = raw\_input("Cassette/CD Name :")

                            N=1

                        elif (self.Cast\_Comp ==0 or len(self.Cast\_Comp)  > 25):

                            print("Company Name should not greater than 25")

                            self.Cast\_Comp = raw\_input("Company Name:")

                            C=1

                        elif (self.Cast\_Price <= 0):

                                print("Enter valid price for Cassette/CD")

                                self.Cast\_Price = float(input("Individual Cassette/CD price:"))

                                P=1

                    else:

                        with open(MFile, 'ab+') as Mobj:

                            if not Mobj:

                                print (MFile, "is not created")

                            else:

                                # Appends data into a sequence object

                                MList = list()

                                MList.append(self.Cast\_Code)

                                MList.append(self.Cast\_Name)

                                MList.append(self.Cast\_Comp)

                                MList.append(self.Cast\_Price)

                               # Write data into binary file

                                dump(MList, Mobj)

            else:

                print ("\ncode", self.Cast\_Code,"is already in 'Master.dat' file")

                ch = raw\_input("\n\nAdd new Cassette/CD code? <Y?N>: ")

                ch = ch.upper()

                if ch=='Y':

                    self.Cast\_Code =int(input("\nCassette/CD Code [Enter 6 digit code]: "))

                    continue

                else:

                    break

    def Master\_Display(self):

        if not os.path.isfile(MFile):

            print (MFile, "file does not exist")

        else:

            Mobj = open(MFile, 'rb')

            print "\nCassette/CD Master Report"

            print "=" \* 100

            print " Code", "\tCassette/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tPrice"

            print "-" \* 100

            try:

                while True:

                    MRec = []

                    MRec =load(Mobj)

                    print str(MRec[0]), "\t",MRec[1],(" "\*(25-len(MRec[1]))),"\t", MRec[2],(" "\*(25-len(MRec[2]))), "\t\t",MRec[3]

            except EOFError:

                pass

            print "-" \* 100

            Mobj.close()

class Cassettes:

    def \_\_init\_\_(self):

        self.Cast\_Code = 0

        self.Tot\_Cast = 0           # Total cassette/CD purchased

        self.dd = [self.mm](http://self.mm/) = self.yy = 0  # Cassette/CD purchase date

            # For cassette/CDs entry into the cassettes.dat data file

    def New\_Cassettes(self):

        M = Master()

        B = Balance()

        CDt = Cast\_Date()

        self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

        self.yy = CDt.yy

        TRec = list() #A temporary list to store master record

        print("Add New Stock cassette/CD");

        ch = 'Y'

        while ch == 'Y':

            Flag = False   # To check if Cast\_Code is in Master.dat or not

            print("\n Date: %s-%s-%s" %  (CDt.dd,CDt.mm, CDt.yy))

            while True:

                self.Cast\_Code = int(input("\nCassette/CD Code :"))

                   # Function call to check cassette/CD code in Master.dat

                Flag,TRec = M.Check\_Code(self.Cast\_Code)

                if (Flag == True):

                    self.Cast\_Name = TRec[1]       #title of the cassette/CD

                    self.Cast\_Comp = TRec[2]       #cassette/Cd company

                    self.Cast\_Price = TRec[3]      #Price per cassette/CD

                    print"Cassette/CD Name :",self.Cast\_Name

                    print"Company Name : ",self.Cast\_Comp

                    print"Individual Cassette/CD price: ",self.Cast\_Price

                    while True:

                        self.Tot\_Cast = int(input("Enter no of stock cassettes/CDs purchased :"))

                        if (self.Tot\_Cast <= 0):

                            print"Enter valid Cassette/Cd number"

                        else:

                            break

                    ch = raw\_input("\n Do you want to save the record <Y/N>: ")

                    if ch == 'y' or ch== 'Y':

                        CList = list()

                        with open(File1, 'ab') as Cobj:

                            if not Cobj:

                                print File1, "is not created"

                            else:

                                  # Appends data into a sequnce object

                                CList.append(self.Cast\_Code)

                                CList.append(self.Tot\_Cast)

                                CList.append(self.dd)

                                CList.append([self.mm](http://self.mm/))

                                CList.append(self.yy)

                                  #write data into the binary files

                                dump(CList, Cobj)   #B.Add\_to\_File(self.Cast\_Code, self.Tot\_Cast,self.Cast\_Price, self.dd, [self.mm](http://self.mm/), self.yy)

                                B.AddUpdateBalance(CList, self.Cast\_Price)

                                print"Record saved"

                            ch =raw\_input("\nStock more cassette/Cd record? <Y/N>: ")

                            ch = ch.upper()

                            if ch =='Y':

                                continue

                            else:

                                break

                else:

                        print "Please enter existing cassette no."

               # For cassettes/CDs entry into the cassettes.dat data file

    def SetDateFormat(Self,d1, m1, y1):

        fDt = ''

        d11 = str(d1)

        m11 = str(m1)

        y11 = str(y1)

        if (len(d11)==1):

            d11 = '0'+d11

        if (len(m11)==1):

            m11 = '0'+m11

        fDt = d11+'-'+m11+'-'+y11

        return fDt

    def Display\_Cassettes(self):

        M = Master()

        if not os.path.isfile(File1):

            print (File1, "file does not exist ")

        else:

            Cobj = open(File1, 'rb')

            print "\nCassette/CD entry Register"

            print "=" \* 125

            print " Code", "\tCassettes/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tQuantity","\t\tPrice","\t\tDate"

            print "-" \* 125

            try:

                while True:

                    CRec = []

                    CRec = load(Cobj)

                    TRec = list()

                    Flag, TRec = M.Check\_Code(CRec[0])

                    nDt =self.SetDateFormat(CRec[2],CRec[3], CRec[4])

                    if (Flag == True):

                        print CRec[0],"\t", TRec[1],(" "\*(25-len(TRec[1]))),"\t",TRec[2],(" "\*(25-len(TRec[2])))< "\t\t",CRec[1],"\t\t", TRec[3],"\t\t",nDt

            except EOFError:

                pass

            print "-" \* 125

            Cobj.close()

              # Function to set the data as: DD-MM-YYYY

class Balance:

    def \_\_init\_\_(self):

                  # Instance attributes of Balance.dat data file

        self.Cast\_Code = 0           # cassette/Cd code to be balance

        self.Cast\_Price = 0          # Total numbert of cassettes/CDs in balance

        self.Cast\_Price = 0          #Unit price of cassettes/CDs on code wise

        self.dd = [self.mm](http://self.mm/) = self.yy = 0  # Balance date

    def Give\_Balance(self, C\_Code):

        Tbalance = 0

        if not os.path.isfile(File2):

                       # When file does not exist

            return False

        else:

            Brec = list()  # A list to extract record from Balance.dat

            Tbalance = 0

            Bobj = open(File2,'rb')

            try:

                while True:

                    BRec = load(Bobj)

                    if (C\_Code == BRec[0]):

                        Tbalance = BRec[1]   #E.g. Cast\_Bal

                        break

            except EOFError:

                    pass

                    Bobj.close()

    def AddUpdateBalance(self, CList, CPrice):

                   # To know the balance cassette in 'Balance.daat'

        Cbalance = Balance.Give\_Balance(self, CList[0])

        if (Cbalance == False): # If file does not exist , add the record for firsttime

            BRec = list()

            with open(File2, 'ab') as Bobj:

                BRec.append(CList[0]) #Cast\_Code

                BRec.append(CList[1]) #Cast\_Bal

                BRec.append(CPrice) #Cast\_Price

                BRec.append(CList[2]) # Day

                BRec.append(CList[3]) # Month

                BRec.append(CList[4]) # Year

                dump(BRec, Bobj)

        elif (Cbalance >= 0):

            Bobj = open(File2, 'rb')

            Tobj = open("Temp.dat", 'wb')

            try:

                while True:

                    BRec = list()         # A list to extract record from Balance.dat

                    BRec = load(Bobj)

                    if (CList[0] != BRec[0]):

                                  # Write data into Temp.dat file

                        dump(BRec,Tobj)

                    else:

                        BRec[1] = Cbalance + CList[1]

                                  #self.Cast\_Bal = self.Cast\_Bal + Cbalance

                        dump(BRec, Tobj)

            except EOFError:

                pass

            Tobj.close()

            Bobj.close()

            os.remove("Balance.dat")

            os.rename("Temp.dat", "Balance.dat")

    def UpdateBalance(self, Clist):

        Bobj = open(File2, 'rb')

        Tobj = open("Temp.dat", 'wb')

        try:

            while True:

                BRec = load(Bobj)

                if (Clist[0] != BRec[0]):

                           # Write data into Temp.dat file

                    dump (BRec, Tobj)

                else:

                    BRec[1] = BRec[1] - CList[4]

                    dump(BRec, Tobj)

        except EOFError:

            pass

        Tobj.close()

        Bobj.close()

        os.remove("Balance.dat")

        os.rename("Temp.dat", "Balance.dat")

        print'Balance.dat updated'

    def Balance\_Cassettes(self):

        M = Master()

        if not os.path.isfile(File2):

            print File2, "file does not exist"

        else:

            TAmount = 0

            print "\nBalance Stock Register(Cassette/CD) "

            print "=" \* 105

            Bobj = open(File2, 'rb')

            print " Code","\tCassette/CD Name",(" "\*(25-len("Cassette/CD Name"))), "\tCompany Name",(" "\*(25-len("Company Name"))), "\t\tQuantity","\t\tPrice", "\t\tAmount"

            print "-" \* 105

            try:

                while True:

                    BRec = []

                    BRec = load(Bobj)

                    TRec = list()

                    Flag, TRec = M.Check\_Code(BRec[0])

                    if (Flag == True):

                        Amount = BRec[1] \* BRec[2]

                        TAmount = TAmount \* Amount

                        print BRec[0],"\t", TRec[1], (" "\*(25-len(TRec[1]))),"\t",TRec[2],(" "\*(25-len(TRec[2])))< "\t",BRec[1],"\t\t", BRec[2],"\t",Amount

            except EOFError:

                pass

            print "-" \* 105

            print "%s Total Amount: %s %.2f" % (' ' \*56,'' \* 4, TAmount)

            Bobj.close()

class Customer:

    def \_\_init\_\_(self):

                               # Instance attributes of Customer.dat data file

        self.Cast\_Code = 0    # cassette/CD code

        self.C\_Name = ''      # Customer name

        self.C\_Addresss = ''  # Customer address

        self.C\_MPhone = 0     # Customer mobile no.

        self.No\_Of\_Cast = 0   # Number of Cassette/Cd

        self.dd = [self.mm](http://self.mm/) = self.yy = 0 # sale date

    def Cassette\_Sale(self):

        M =Master()

        B = Balance()

        CDt = Cast\_Date()

        self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

        self.yy = CDt.yy

        Cbalance = 0

        print"Customer sales cassette/CD"

        ch ='Y'

        while ch=='Y':

            TRec = list() # A temporary list to store master record

            Flag = False  # To check if Cast\_Code is in Master.dat or not

            print "Date: %s-%s-%s" % (CDt.dd, CDt.mm,CDt.yy)

            while True:

                self.Cast\_Code = int(input("\nCassette/CD Code (Enter 6 digit code): "))

                       # Function call to check cassette/Cd code in Master.dat

                Flag, TRec = M.Check\_Code(self.Cast\_Code)

                Cbalance = B.Give\_Balance(self.Cast\_Code)

                if (Flag == True):

                    self.Cast\_Name = TRec[1]   #Title of the cassette/CD

                    self.Cast\_Comp = TRec[2]   #cassette/CD company

                    self.Cast\_Price = TRec[3]   #Price per cassette/CD company

                    print "Cassette/CD Name :", self.Cast\_Name

                    print "Company Name :", self.Cast\_Comp

                    print "Individual Cassette/CD price : ",self.Cast\_Price

                    print "\nEnter Customer details",

                    self.C\_Name = raw\_input("Customer name: ").upper()

                    self.C\_Address = raw\_input("Custoer address: ")

                    self.C\_MPhone = raw\_input("Customer mobile no: ")

                    while True:

                        self.No\_Of\_Cast = int(input("\nEnter sales cassettes/CDs nos.: "))

                        if (self.No\_Of\_Cast > Cbalance):

                            print "Out of Stock"

                        else:

                            break

                    ch = raw\_input("\nSales confirm <Y/N>: ").upper()

                    if ch == 'Y':

                        CustList = list()

                        with open(File3, 'ab') as CustObj:

                            if not CustObj :

                                print File3, "is not Created"

                            else:

                                      # Appends data into a sequence object

                                CustList.append(self.Cast\_Code)

                                CustList.append(self.C\_Name)

                                CustList.append(self.C\_Address)

                                CustList.append(self.C\_MPHone)

                                CustList.append(self.No\_of\_Cast)

                                CustList.append(self.dd)

                                CustList.append([self.mm](http://self.mm/))

                                CustList.append(self.yy)

                                B.UpdateBalance(CustList)

                                dump(CustList, CustObj)

                    ch=raw\_input("\nMore sale? <Y/N>:")

                    ch=ch.upper()

                    if ch!='Y':

                        break

            #Function to search individual customer on mobile no.

    def Return\_CustomerName(self, Mno):

        M=Master()

        CName=''

        if not os.path.isfile(File2):

            print File3, "file does not exist"

        else:

            CustObj=open(File3, 'rb')

            try:

                while True:

                    CustRec = [ ]

                    CustRec = load(CustObj)

                    if Mno == CustRec[3]:

                        CName = CustRec[" "]

                        break

            except EOFError:

                pass

            CustObj.close()

        return CName

            #Function to display Sales report for a particular month in a calender year.

    def MonthlySales\_Report(self):

        M=Master()

        if not os.path.isfile(File2):

            print (File3, "file does not exist")

        else:

            monthNo = int(input('\nEnter month number:'))

            yearNo = int(input('\nEnter year:'))

            CDt = Cast\_Date()

            self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

            self.yy = CDt.yy

            D=Cassettes()

            if (monthNo <= 12 and monthNo <= [self.mm](http://self.mm/) and yearNo <= self.yy):

                        #Function call for a character month

                MonthName=Month\_Name(monthNo)

                TAmount = 0

                        #Function called to set the date as DD-MM-YY

                nDt =D.SetDateFormat(self.dd, [self.mm](http://self.mm/), self.yy)

                print "\nCustomer Sales Status Report - Date:", nDt

                print "For the month of", MonthName, yearNo

                print "-" \* 100

                CustObj = open(File3, 'rb')

                print "{0:<20} {1:<12} {2:<25} {3:^10} {4:>5} {5:>12} {6:>8}". format("Name", "Mobile No.", "Cassette/CD Cide and Name", "Date", "Qty", "Unit price", "Amount")

                print "-" \* 100

                try:

                    while True:

                        CustRec = []

                        CustRec = load(CustObj)

                        TRec = list()

                        Flag, TRec = M.Check\_Code(CustRec[0])

                        UPrice = TRec[3]

                        Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] #An additional 20% of Unit Price

                        nDt = D.SetDateFormat(CustRec[5], CustRec[6], CustRec[7])

                        TAmount+=Amount

                        if (monthNo == CustRec[6] and yearNo == CustRec[7]):

                            Clength = str(CustRec[0])+'-'+TRec[1]

                            nName = ''

                            for i in range(len(Clength)):   #Extracts only 24 characters

                                nName = nName + Clength[i]

                                if i == 23:

                                    break

                                print "{0:20} {1:<12} {2:<25} {3:>10} {4:>5.0f} {5:>12.2f} {6:>8.2f}". format(CustRec[1], CustRec[3], nName, nDt, CustRec[4], UPrice, Amount)

                except EOFError:

                    pass

                    print "-" \* 100

                    print'Note. Amount is calculated as 20% extra on unit price.'

                    print ("%s Total Sales Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

                    CustObj.close()

            else:

                print "Month number and year is not valid"

                    #Function to display close wise monthly sales report.

    def CodeWiseMonthlySales\_Report(self):

        M=Master()

        TRec = list() #A temporary list to store master record

        Flag = False  #To check if Cast\_Code is in Master.dat or not

        if not os.path.isfile(File2):

            print (File3, "file does not exist")

        else:

            CCode = int(input("\nCassette/CD Code:"))

            monthNo = int(input('\nEnter month number:'))

            yearNo = int(input('\nEnter year:'))

            CDt = Cast\_Date()

            self.dd = CDt.dd

[self.mm](http://self.mm/) = CDt.mm

            self.yy = CDt.yy

                    #Function to call to check cassette/CD code in Master.dat

            Flag, TRec = M.Check\_Code(CCode)

            if (monthNo <= 12 and  monthNo <= [self.mm](http://self.mm/) and yearNo <= self.yy and Flag == True):

                CName = TRec[1]  #Tittle of the casette/CD

                CComp = TRec[2]  #casette/CD company

                CPrice = TRec[3] #Price per cassette/CD

                        #Function Call for a character month

                MonthName = Month\_Name (monthNo)

                TAmount = 0

                        #Function Called to set date as DD-MM-YY

                nDt = Set\_DateFormat(self.dd, [self.mm](http://self.mm/), self.yy)

                print "\nCode wise Sales Report - Date:", nDt

                print "For the month of", MonthName, yearNo

                print "Cassette/CD Code:  %d  Name: %s" (CCode, CName)

                print

                print "-" \* 77

                CustObj = open(File3, 'rb')

                print "{0:<20} {1:<12} {2:^10} {3:>5} {4:>12} {5:>8}". format("Name", "Mobile No.", "Cassette/CD Cide and Name", "Date", "Qty", "Unit price", "Amount")

                print "-" \* 77

                ctr = 0

                try:

                    while True:

                        CustRec = []

                        CustRec = load(CustObj)

                        TRec = list()

                        Flag, TRec = M.Check\_Code(CustRec[0])

                        UPrice = TRec[3]

                        Amount = (UPrice + (UPrice \* 0.20)) \* CustRec[4] #An additional 20% of Unit Price

                        nDt = Set\_DateFormat(CustRec[5], CustRec[6], CustRec[7])

                        TAmount+=Amount

                        if (monthNo == CustRec[6] and yearNo == CustRec[7]) and (CCode == CustRec[0]):

                            ctr += 1

                            print "{0:20} {1:<12} {2:>10} {3:>5.0f} {4:>12.2f} {5:>8.2f}". format(CustRec[1], CustRec[3], nDt, CustRec[4], UPrice, Amount)

                except EOFError:

                    pass

                print "-" \* 77

                if (ctr == 0):

                    print 'No record found on such Code No. , Month and Year'

                else:

                    print'Note. Amount is calculated as 20% extra on unit price.'

                    print ("%s Total Sales Amount: %s %.2f" % (' ' \* 50, ' ' \* 4, TAmount))

                CustObj.close()

            else:

                print "Either Code not found or Month no. and year is not valid"

                 #Function to search individual customer on mobile no.

    def Month\_Name(self,mNo):

        mDict = {1:'January', 2:'February', 3:'March', 4:'April', 5:'May', 6:'June', 7:'July', 8:'August', 9:'September', 10:'October', 11:'November', 12:'December'}

        nName = ''

        for key, value in mDict.items():

            if (key == mNo):

                mName = value

                break

        return mName

while True:

    M=Master()

    CS=Cassettes()

    BL=Balance()

    Cust=Customer()

    print

    print "\n Video Library Main Menu"

    print "-" \* 50

    print "       1 - >  Master Cassettes/CDs"

    print "       2 - >  Stock Cassettes/CDs"

    print "       3 - >  Customer Sales"

    print "       4 - >  Exit"

    print "-" \* 50

    opt = input("Enter your choice(1,2,3 or 4):")

    if opt==1:

        ch=0

        while True:

            print

            print "\n\tMaster Cassettes Menu"

            print "-" \* 50

            print "       1 - >  Cassettes/CDs Stock Entry"

            print "       2 - >  View Cassettes/CDs"

            print "       3 - >  Back to Main Menu"

            print "-" \* 50

            ch = input("Enter your choice:")

            if ch==1:

                M.Master\_Entry()

            elif ch==2:

                M.Master\_Display()

            elif ch==3:

                break

    elif opt==2:

        ch=0

        while True:

            print

            print "\n\tStock Cassettes Menu"

            print "-" \* 50

            print "       1 - >  Cassettes/CDs Stock Entry"

            print "       2 - >  Display Cassettes/CDs"

            print "       3 - >  Stock/Balance Cassettes"

            print "       4 - >  Back to Main Menu"

            print "-" \* 50

            ch = input("Enter your choice:")

            if ch==1:

                CS.New\_Cassettes()

            elif ch==2:

                CS.Display\_Cassettes()

            elif ch==3:

                BL.Balance\_Cassettes()

            elif ch==4:

                break

    elif opt==3:

        ch=0

        while True:

            print "\n\tCustomer Sales Menu"

            print "-" \* 50

            print "       1 - >  Sales Entry"

            print "       2 - >  Monthly Sales Report"

            print "       3 - >  Code Wise Monthly Sales"

            print "       4 - >  Back to Main Menu"

            print "-" \* 50

            ch = int(raw\_input("Enter your choice:"))

            if ch==1:

                Cust.Cassette\_Sale()

            elif ch==2:

                Cust.MonthlySales\_Report()

            elif ch==3:

                Cust.CodeWiseMonthlySales\_Report()

            elif ch==4:

                    break

    elif opt==4:

        print "Thank you and have a nice day"

        break

    else:

        print "Please enter correct option!"

**OUTPUT**





























