Library Management Project

=================================

Arghyadip Chakraborty

Sherlock Inc.

from textwrap import wrap

from pickle import load,dump

from os import makedirs

from time import localtime

from datetime import datetime

class Book:

AccNo=""

Title=""

Author=""

Publisher=""

Price=0

Edition=0

Pages=0

DOP=""

Issued=False

Active=False

def \_\_init\_\_(self,AN,Ti,Au,Pb,Pr,Ed,Pg,DP):

self.AccNo=AN

self.Title=Ti

self.Author=Au

self.Publisher=Pb

self.Price=Pr

self.Edition=Ed

self.Pages=Pg

self.DOP=DP

self.Issued=False

self.Active=True

def modify(self,Ti,Au,Pb,Pr,Ed,Pg):

self.Title=Ti if Ti else self.Title

self.Author=Au if Au else self.Author

self.Publisher=Pb if Pb else self.Publisher

self.Price=Pr if Pr else self.Price

self.Edition=Ed if Ed else self.Edition

self.Pages=Pg if Pg else self.Pages

class Member:

MNo=""

MName=""

MAddress=""

MDOB=""

MMobNo=""

MDOJ=""

MBIssued=False

MActive=False

def \_\_init\_\_(self,MN,Nm,Ad,DB,Mob,DJ):

self.MNo=MN

self.MName=Nm

self.MAddress=Ad

self.MDOB=DB

self.MMobNo=Mob

self.MDOJ=DJ

self.MBIssued=False

self.MActive=True

def modify(self,Nm,Ad,DB,Mob):

self.MName=Nm if Nm else self.MName

self.MAddress=Ad if Ad else self.MAddress

self.MDOB=DB if DB else self.DOB

self.MMobNo=Mob if Mob else self.MMobNo

class Issue:

IssNo=""

BAccNo=""

MNo=""

DOI=""

DOR=""

Fine=0

def \_\_init\_\_(self,IN,Acc,MN,DI):

self.IssNo=IN

self.BAccNo=Acc

self.MNo=MN

self.DOI=DI

self.DOR=""

self.Fine=0

def Return(self,DR):

self.DOR=DR

self.Calc\_Fine()

def Calc\_Fine(self):

d1=datetime(\*(map(int,self.DOI.split('/'))[::-1]))

d2=datetime(\*(map(int,self.DOR.split('/'))[::-1]))

dif=str(d2-d1)

day=0

try:

day=int(dif[:dif.find(' ')])

except:

day=0

self.Fine=(day-14)\*2 if day>14 else 0

class Library:

pth=""

def \_\_init\_\_(self,pt="F:/"):

self.pth=pt+'Library/'

for x in ('Books/','Members/','Issues/'):

try:

makedirs(self.pth+x)

except:

pass

def AH(self,No,Md='rb'):

ext=No[0]

if ext not in "BMI" or ext=='':

return None

de=dict(zip(('B','M','I'),('Books/','Members/','Issues/')))

tf=None

try:

tf=open(self.pth+de[ext]+"%s.dat"%No,Md)

except:

pass

return tf

def NNo(self,ext='',minn=1):

nno=minn

while True:

tf=self.AH(ext+str(nno))

if not tf:

yield ext+str(nno)

else:

nno+=1

tf.close()

def Book\_Add(self):

try:

NAcc=self.NNo('B')

Acc=next(NAcc)

print "Enter Accession No: %s"%Acc

Ti=raw\_input("Enter Title: ")

Au=raw\_input("Enter Author: ")

Pb=raw\_input("Enter Publisher: ")

Pr=input("Enter Price: Rs ")

Ed=input("Enter Edition: ")

Pg=input("Enter No of Pages: ")

DP='/'.join(map(str,list(localtime())[2::-1]))

if Ti and Au and Pb and Pr and Ed and Pg and DP:

bk=Book(Acc,Ti,Au,Pb,Pr,Ed,Pg,DP)

tf=self.AH(Acc,'wb+')

dump(bk,tf)

tf.close()

print "Book has been Added..."

else:

print "Sorry! Not Enough Details to Add Book!"

print "Please Try Again..."

except:

print "Sorry! Book Could not be Added!"

print "Please Try Again..."

def Book\_Remove(self):

try:

Acc='B'+raw\_input("Enter Accession No: B")

tf=self.AH(Acc)

if not tf:

print "Book doesn't Exists..."

return

bk=load(tf)

tf.close()

if not bk.Active:

print "Book has been Already Removed..."

return

elif bk.Issued:

print "Book is Issued to Some Member..."

return

bk.Active=False

tf=self.AH(Acc,'wb+')

dump(bk,tf)

tf.close()

print "Book has been Removed..."

except:

print "Sorry! Book Could not be Removed!"

print "Please Try Again..."

def Book\_Modify(self):

try:

Acc='B'+raw\_input("Enter Accession No: B")

tf=self.AH(Acc)

if not tf:

print "Book doesn't Exists..."

return

bk=load(tf)

tf.close()

if not bk.Active:

print "Book has been Removed..."

return

elif bk.Issued:

print "Book is Issued to Some Member..."

return

Ti=raw\_input("Enter Title: ")

Au=raw\_input("Enter Author: ")

Pb=raw\_input("Enter Publisher: ")

Pr=raw\_input("Enter Price: Rs ")

Pr=0 if not Pr else int(Pr)

Ed=raw\_input("Enter Edition: ")

Ed=0 if not Ed else int(Ed)

Pg=raw\_input("Enter No of Pages: ")

Pg=0 if not Pg else int(Pg)

bk.modify(Ti,Au,Pb,Pr,Ed,Pg)

tf=self.AH(Acc,'wb+')

dump(bk,tf)

tf.close()

print "Book has been Modified..."

except:

print "Sorry! Book Could not be Modified!"

print "Please Try Again..."

def Mem\_Add(self):

try:

NMN=self.NNo('M')

MNo=next(NMN)

print "Enter Member No: %s"%MNo

Nm=raw\_input("Enter Name: ")

Ad=raw\_input("Enter Address: ")

DB=raw\_input("Enter Date of Birth: ")

Mob=raw\_input("Enter Mobile No: ")

DJ='/'.join(map(str,list(localtime())[2::-1]))

if Nm and Ad and DB and Mob and DJ:

mem=Member(MNo,Nm,Ad,DB,Mob,DJ)

tf=self.AH(MNo,'wb+')

dump(mem,tf)

tf.close()

print "Member has been Added..."

else:

print "Sorry! Not Enough Details to Add Member!"

print "Please Try Again..."

except:

print "Sorry! Member Could not be Added!"

print "Please Try Again..."

def Mem\_Remove(self):

try:

MNo='M'+raw\_input("Enter Member No: M")

tf=self.AH(MNo)

if not tf:

print "Member doesn't Exists..."

return

mem=load(tf)

tf.close()

if not mem.MActive:

print "Member has been Already Removed..."

return

elif mem.MBIssued:

print "Member is Issued Some Book..."

return

mem.MActive=False

tf=self.AH(MNo,'wb+')

dump(mem,tf)

tf.close()

print "Member has been Removed"

except:

print "Sorry! Member Could not be Removed!"

print "Please Try Again..."

def Mem\_Modify(self):

try:

MNo='M'+raw\_input("Enter Member No: M")

tf=self.AH(MNo)

if not tf:

print "Member doesn't Exists..."

return

mem=load(tf)

tf.close()

if not mem.MActive:

print "Member has been Removed..."

return

elif mem.MBIssued:

print "Member is Issued Some Book..."

return

Nm=raw\_input("Enter Name: ")

Ad=raw\_input("Enter Address: ")

DB=raw\_input("Enter Date of Birth: ")

Mob=raw\_input("Enter Mobile No: ")

mem.modify(Nm,Ad,DB,Mob)

tf=self.AH(MNo,'wb+')

dump(mem,tf)

tf.close()

print "Member has been Modified..."

except:

print "Sorry! Member Could not be Modified!"

print "Please Try Again..."

def Book\_Issue(self):

try:

NIN=self.NNo('I')

IN=next(NIN)

print "Enter Issue No: %s"%IN

Acc='B'+raw\_input("Enter Accession No: B")

tf=self.AH(Acc)

if not tf:

print "Book doesn't Exists..."

return

bk=load(tf)

tf.close()

if not bk.Active:

print "Book has been Already Removed..."

return

elif bk.Issued:

print "Book is Issued to Some Member..."

return

MNo='M'+raw\_input("Enter Member No: M")

tf=self.AH(MNo)

if not tf:

print "Member doesn't Exists..."

return

mem=load(tf)

tf.close()

if not mem.MActive:

print "Member has been Removed..."

return

elif mem.MBIssued:

print "Member is Issued Some Book..."

return

DI='/'.join(map(str,list(localtime())[2::-1]))

isu=Issue(IN,bk.AccNo,mem.MNo,DI)

bk.Issued=True

mem.MBIssued=True

tf=self.AH(IN,'wb+')

dump(isu,tf)

tf.close()

tf=self.AH(bk.AccNo,'wb+')

dump(bk,tf)

tf.close()

tf=self.AH(mem.MNo,'wb+')

dump(mem,tf)

tf.close()

print "Book has been Issued..."

except:

print "Sorry! Book Could not be Issued!"

print "Please Try Again..."

def Book\_Return(self):

try:

IN='I'+raw\_input("Enter Issue No: I")

tf=self.AH(IN)

if not tf:

print "Issue doesn't Exits..."

return

isu=load(tf)

tf.close()

DR='/'.join(map(str,list(localtime())[2::-1]))

isu.Return(DR)

if isu.Fine>0:

print "Fine: Rs%s"%isu.Fine

tf=self.AH(IN,'wb+')

dump(isu,tf);tf.close()

tf=self.AH(isu.BAccNo)

bk=load(tf)

tf.close()

bk.Issued=False

tf=self.AH(isu.BAccNo,'wb+')

dump(bk,tf);tf.close()

tf=self.AH(isu.MNo)

mem=load(tf)

tf.close()

mem.MBIssued=False

tf=self.AH(isu.MNo,'wb+')

dump(mem,tf)

tf.close()

print "Book has been Returned..."

except:

print "Sorry! Book Could not be Returned!"

print "Please Try Again..."

def Get\_Books(self):

bkl=list()

Acc=1

while True:

tf=self.AH('B'+str(Acc))

if not tf:

break

bk=load(tf)

bkl.append(bk)

tf.close()

Acc+=1

return bkl

def Get\_Members(self):

meml=list()

MNo=1

while True:

tf=self.AH('M'+str(MNo))

if not tf:

break

mem=load(tf)

meml.append(mem)

tf.close()

MNo+=1

return meml

def Get\_Issues(self):

isul=list()

IN=1

while True:

tf=self.AH('I'+str(IN))

if not tf:

break

isu=load(tf)

isul.append(isu)

tf.close()

IN+=1

return isul

lib=None

def LibMang():

print '\*'\*79

print ('\*'\*5+" WELCOME TO SHERLOCK'S LIBRARY "+'\*'\*5).center(79)

print '-'\*79 df={1:Book\_Maint,2:Mem\_Maint,3:Book\_IR,4:Report\_Port,5:Search\_Port}

ch='1'

while ch in "12345":

print'''

------------------------------------

|\_\_\_\_\_::MAIN MENU::\_\_\_\_\_\_|

| Enter 1.Book Maintenance |

| Enter 2.Member Maintenance |

| Enter 3.Book Issue/Return |

| Enter 4.Report Portal |

| Enter 5.Search Portal |

| Enter 6.Exit |

------------------------------------'''

ch=''

while ch not in "123456" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch!='6':

ch=df[int(ch)]()

print '-'\*79

print ('\*'\*5+" THANK YOU "+'\*'\*5).center(79)

print '\*'\*79

def Book\_Maint():

print '~'\*79

df={1:lib.Book\_Add,2:lib.Book\_Remove,3:lib.Book\_Modify}

ch='1'

while ch in "123":

print'''

----------------------------------

|\_::BOOK MAINTENANCE::\_|

| Enter 1.Add Book |

| Enter 2.Remove Book |

| Enter 3.Modify Book |

| Enter 4.Go Back |

| Enter 5.Exit |

----------------------------------'''

ch=''

while ch not in "12345" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch in "123":

df[int(ch)]()

print '~'\*79

return '6' if ch=='5' else '1'

def Mem\_Maint():

print '~'\*79

df={1:lib.Mem\_Add,2:lib.Mem\_Remove,3:lib.Mem\_Modify}

ch='1'

while ch in "123":

print'''

-------------------------------------

|\_::MEMBER MAINTENANCE::\_|

| Enter 1.Add Member |

| Enter 2.Remove Member |

| Enter 3.Modify Member |

| Enter 4.Go Back |

| Enter 5.Exit |

-------------------------------------'''

ch=''

while ch not in "12345" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch in "123":

df[int(ch)]()

print '~'\*79

return '6' if ch=='5' else '1'

def Book\_IR():

print '~'\*79

df={1:lib.Book\_Issue,2:lib.Book\_Return,3:Search\_Port}

ch='1'

while ch in "123":

print'''

----------------------------------

|\_::BOOK ISSUE/RETURN::\_|

| Enter 1.Issue Book |

| Enter 2.Return Book |

| Enter 3.Search Portal |

| Enter 4.Go Back |

| Enter 5.Exit |

----------------------------------'''

ch=''

while ch not in "12345" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch in "12":

df[int(ch)]()

elif ch=='3':

ch=df[int(ch)]()

ch='5' if ch=='6' else ch

print '~'\*79

return '6' if ch=='5' else '1'

def Report\_Port():

print '~'\*79

df={1:Books\_Report,2:Members\_Report,3:Issues\_Report}

ch='1'

while ch in "123":

print'''

--------------------------------

|\_\_::REPORT PORTAL::\_\_\_|

| Enter 1.Books Report |

| Enter 2.Members Report |

| Enter 3.Issues Report |

| Enter 4.Go Back |

| Enter 5.Exit |

--------------------------------'''

ch=''

while ch not in "12345" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch in "123":

df[int(ch)]()

print '~'\*79

return '6' if ch=='5' else '1'

def Scale(l,x):

if x>=len(l):

return ''

return l[x]

def Display(sz,hd,lt,la=[]):

st=""

for x in range(len(sz)):

st+="{%d:^%s}|"%(x,sz[x])

st=st[:-1]

print

print '='\*79

l=[]

for x in range(len(hd)):

l.append(wrap(hd[x],width=sz[x]))

mxl=max(map(len,l))

for x in range(mxl):

pt=tuple(map(Scale,l,[x]\*len(sz)))

print st.format(\*pt)

print '='\*79

pos=0;n=1

while n<st.count('^'):

pos=st.index('^',pos+1)

if n in la:

st=st[:pos]+'<'+st[pos+1:]

n+=1

for pl in lt:

l=[]

for x in range(len(pl)):

l.append(wrap(pl[x],width=sz[x]))

mxl=max(map(len,l))

for x in range(mxl):

pt=tuple(map(Scale,l,[x]\*len(sz)))

print st.format(\*pt)

print '.'\*79

print '='\*79

def Books\_Display(bkl):

sz=(4,16,12,10,4,4,4,10,3,3)

hd=("Acc No","Title","Author","Publisher","Prce(Rs)","Edition","Pages","Date of Purchase","Issued","Active")

YN={True:"Yes",False:"No"}

lt=[]

for bk in bkl:

bl=[]

bl.append(bk.AccNo)

bl.append(bk.Title)

bl.append(bk.Author)

bl.append(bk.Publisher)

bl.append(str(bk.Price))

bl.append(str(bk.Edition))

bl.append(str(bk.Pages))

bl.append(bk.DOP)

bl.append(YN[bk.Issued])

bl.append(YN[bk.Active])

lt.append(tuple(bl))

Display(sz,hd,lt,[2,3,4])

def Books\_Report(): Books\_Display(lib.Get\_Books())

def Members\_Display(meml):

sz=(4,20,12,10,10,10,3,3)

hd=("Mem No","Name","Address","Date of Birth","Mobile No","Date of Joining","Issued","Active")

YN={True:"Yes",False:"No"}

lt=[]

for mem in meml:

ml=[]

ml.append(mem.MNo)

ml.append(mem.MName)

ml.append(mem.MAddress)

ml.append(mem.MDOB)

ml.append(mem.MMobNo)

ml.append(mem.MDOJ)

ml.append(YN[mem.MBIssued])

ml.append(YN[mem.MActive])

lt.append(tuple(ml))

Display(sz,hd,lt,[2,3])

def Members\_Report(): Members\_Display(lib.Get\_Members())

def Issues\_Display(isul):

sz=(8,10,8,18,19,11)

hd=("Iss No","BAcc No","Mem No","Date of Issue","Date of Return","Fine(Rs)")

YN={True:"Yes",False:"No"}

lt=[]

for isu in isul:

il=[]

il.append(isu.IssNo)

il.append(isu.BAccNo)

il.append(isu.MNo)

il.append(isu.DOI)

il.append(isu.DOR)

il.append(str(isu.Fine))

if il[-2]=='': il[-2]='-'

if il[-1]=='0': il[-1]='-'

lt.append(tuple(il))

Display(sz,hd,lt)

def Issues\_Report(): Issues\_Display(lib.Get\_Issues())

def Search\_Port():

print '~'\*79

df={1:Search\_Book,2:Search\_Member,3:Search\_Issue}

ch='1'

while ch in "123":

print'''

------------------------------

|\_\_::SEARCH PORTAL::\_\_|

| Enter 1.Search Book |

| Enter 2.Search Member |

| Enter 3.Search Issue |

| Enter 4.Go Back |

| Enter 5.Exit |

-------------------------------'''

ch=''

while ch not in "12345" or not ch:

ch=raw\_input("Enter Choice: ")

print

if ch in "123":

df[int(ch)]()

print '~'\*79

return '6' if ch=='5' else '1'

def Search\_Book():

bkl=lib.Get\_Books()

Acc='B'+raw\_input("Enter Accession No: B")

Ti=raw\_input("Enter Title: ")

Au=raw\_input("Enter Author: ")

Pb=raw\_input("Enter Publisher: ")

Pr=raw\_input("Enter Price: Rs ")

Pr=0 if not Pr else int(Pr)

Ed=raw\_input("Enter Edition: ")

Ed=0 if not Ed else int(Ed)

Pg=raw\_input("Enter No of Pages: ")

Pg=0 if not Pg else int(Pg)

fl=[]

for bk in bkl:

match=1

match\*=int(not Acc or Acc in bk.AccNo)

match\*=int(not Ti or Ti in bk.Title)

match\*=int(not Au or Au in bk.Author)

match\*=int(not Pb or Pb in bk.Publisher)

match\*=int(not Pr or Pr==bk.Price)

match\*=int(not Ed or Ed==bk.Edition)

match\*=int(not Pg or Pg==bk.Pages)

if match:

fl.append(bk)

Books\_Display(fl)

def Search\_Member():

meml=lib.Get\_Members()

MNo='M'+raw\_input("Enter Member No: M")

Nm=raw\_input("Enter Name: ")

Ad=raw\_input("Enter Address: ")

DB=raw\_input("Enter Date of Birth: ")

Mob=raw\_input("Enter Mobile No: ")

fl=[]

for mem in meml:

match=1

match\*=int(not MNo or MNo in mem.MNo)

match\*=int(not Nm or Nm in mem.MName)

match\*=int(not Ad or Ad in mem.MAddress)

match\*=int(not DB or DB in mem.MDOB)

match\*=int(not Mob or Mob in mem.MMobNo)

if match:

fl.append(mem)

Members\_Display(fl)

def Search\_Issue():

isul=lib.Get\_Issues()

IN='I'+raw\_input("Enter Issue No: I")

Acc='B'+raw\_input("Enter Accession No: B")

MNo='M'+raw\_input("Enter Member No: M")

DI=raw\_input("Enter Date of Issue: ")

DR=raw\_input("Enter Date of Return: ")

Fn=raw\_input("Enter Minimum Fine Amount: Rs")

Fn=0 if not Fn else int(Fn)

fl=[]

for isu in isul:

match=1

match\*=int(not IN or IN in isu.IssNo)

match\*=int(not Acc or Acc in isu.BAccNo)

match\*=int(not MNo or MNo in isu.MNo)

match\*=int(not DI or DI==isu.DOI)

match\*=int(not DR or DR==isu.DOR)

match\*=int(Fn<=isu.Fine)

if match:

fl.append(isu)

Issues\_Display(fl)

#Main

lib=Library()

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

LibMang()