**ACKNOWLEDGEMENT**

First of all I would like to extend my sincere gratitude towards the CBSE for giving us students an opportunity to work on such projects that help us in working with Python in a practical way. I also express my special thanks to my Computer Science teacher, Mr. Jagadeesan, whose guidance and teaching immensely helped me in completing this project. Secondly, I thank my team mates (Alisha and Vedha) for their valuable contribution towards this project. Lastly I would like to thank my parents and friends for their never ending support.

**OVERVIEW**

**AIM OF THE PROJECT**

The aim of the project is to create a home management system for HOPE ORPHANAGE that would enable them to add or delete member, track their personal details, update their details, and give feedback which would help them to maintain their home records properly.

**FILES USED**

Hope.Add1.CSV

Hope.Add2.CSV

Hope.feedback.CSV

Hope.marks.CSV

**CLASSES AND METHODS USED**

CLASSES

1. class Add

In this class there are a set of functions for adding child and adult members to their respective files.

2. class Existence

This class contains a set of functions for viewing, updating, and deleting details of existing members.

3. class Track

This class helps in tracking of details of all members by the maintenance department (admin staff)

4. class Feedback

This class serves as a platform for the maintenance department to communicate details pertaining to a member.

5. class Bind

This is a derived class that inherits the properties of all the previously defined classes, binding the methods in one master unit

FUNCTIONS**:**

1. Under class Add

a) \_\_init\_\_(): Constructor function

b) addChild(): writes the details of child

members into the respective file.

c) addAdult(): writes the details of adult

members into the respective file.

2. Under class Existence

a) View(): Displays details of existing members

b) Upadatechild(): Updates the values of certain

arguments of a child member

c) Updateadult() : Updates the values of certain

arguments of an adult member

d) Deletechild() : Deletes all the data of a

child member

e) Deleteadult() : Deletes all the data of an

adult member.

3. Under class Track

a) track(): Helps in accessing the data of

particular member.

4. Under class Feedback

a) \_\_init\_\_() : Constructor function

b) inp() : Helps the admin staff to enter

and store the feedback of a

child.

c) marks() : Helps the admin staff to enter

and store the marks of a

child.

d) displayinp() : Displays the stored details of

feedback.

e) displaymarks(): Displays the stored details of

marks

5. Under class Bind

Contains all the previously mentioned functions of the classes mentioned above since this is a derived class.

a) \_\_init\_\_()

b) addChild()

c) addAdult()

d) query()

e) updatechild()

f) updateadult()

g) deletechild()

h) deleteadult()

i) track()

j) inputfeedback()

k) inputmarks()

l) feedback()

m) marks()

**BLOCK DIAGRAM**

**\_SOURCE CODE\_**

import os

class Add(object):

#Set of functions for adding child and adult members to their respective files.

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

#Constructor function: Initializes the arguments of current

object.

self.name=None

self.id=0

self.dob=None

self.age=0

self.sex=None

self.rel=None

self.adp=0

self.state=None

self.work=None

self.edu=None

self.hobby=None

def addChild(self):

#Checks existence of file, generates identity number, writes details of child members into the file.

if not os.path.isfile('Hope.Add1.CSV'):

self.id=1

else:

wFile1=open('Hope.Add1.CSV','r')

L=wFile1.readlines()

self.id=1+len(L)

wFile1.close()

self.name=input("Name of the member: ")

self.dob=input("Enter the DOB(dd.mm.yyyy): ")

self.age=int(input("Age of the member: "))

self.sex=input("Male/Female: ")

self.rel=input("Name of the relative: ")

self.adp=int(input("Mobile/Phone no.: "))

self.state=input("Native state: ")

wFile1=open('Hope.Add1.CSV','a')

wFile1.write(self.name+','+(str(self.id))+','+self.dob+','+(str(self.age))+','+self.sex+','+self.rel+','+(str(self.adp))+','+self.state+'\n')

wFile1.close()

import os

class Add(object):

#Set of functions for adding child and adult members to their respective files.

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

#Constructor function: Initialises the arguments of current

object.

self.name=None

self.id=0

self.dob=None

self.age=0

self.sex=None

self.rel=None

self.adp=0

self.state=None

self.work=None

self.edu=None

self.hobby=None

def addChild(self):

#Checks existence of file, generates identity number, writes details of

child members into the file.

if not os.path.isfile('Hope.Add1.CSV'):

self.id=1

else:

wFile1=open('Hope.Add1.CSV','r')

L=wFile1.readlines()

self.id=1+len(L)

wFile1.close()

self.name=raw\_input("Name of the member: ")

self.dob=raw\_input("Enter the DOB(dd.mm.yyyy): ")

self.age=int(input("Age of the member: "))

self.sex=raw\_input("Male/Female: ")

self.rel=raw\_input("Name of the relative: ")

self.adp=int(input("Mobile/Phone no.: "))

self.state=raw\_input("Native state: ")

wFile1=open('Hope.Add1.CSV','a')

wFile1.write(self.name+','+(str(self.id))+','+self.dob+','+(str(self.age))+','+self.sex+','+self.rel+','+(str(self.adp))+','+self.state+'\n')

wFile1.close()

print"Data recorded successfully"

print"Data recorded successfully)

def addAdult(self):

#Checks existence of file, generates identity number, writes details of

adult members into the file.

if not os.path.isfile('Hope.Add2.CSV'):

self.id=1001

else:

wFile2=open('Hope.Add2.CSV','r')

L=wFile2.readlines()

self.id=1001+len(L)

wFile2.close()

self.name=raw\_input("Name of the member: ")

self.dob=raw\_input("Enter the DOB (dd.mm.yyyy): ")

self.age=int(input("Age of the member: "))

self.sex=raw\_input("Male/Female: ")

self.rel=raw\_input("Name of the relative: ")

self.adp=int(input("Mobile/Phone no.: "))

self.state=raw\_input("Native state: ")

self.work=raw\_input("Enter member's occupation: ")

self.edu=raw\_input("Enter member's academic qualifications: ")

self.hobby=raw\_input("Enter member's hobbies: ")

wFile2 = open('Hope.Add2.CSV','a')

wFile2.write(self.name+','+(str(self.id)+','+self.dob+','+(str(self.age))+','+self.sex+','+self.rel+','+(str(self.adp))+','+self.state+','+self.work+','+self.edu+','+self.hobby+'\n'))

wFile2.close()

print"Data recorded successfully"

import os

class Existence(object):

#Set of functions for viewing, updating and deleting details of existing

members.

def View(self):

#Displays details of existing members.

wFile1=open('Hope.Add1.CSV','r')

child\_list = wFile1.readlines()

wFile1.close()

name=raw\_input("Enter the name of the member whose details

you want to view: ")

if not os.path.exists('Hope.Add1.CSV'):

print"File not found"

return

for record in child\_list:

L = record.rstrip('\n').split(',')

if str(name) == str(L[0]):

found=True

ide=(L[1])

dob=(L[2])

age=(L[3])

sex=(L[4])

rel=(L[5])

adp=(L[6])

state=(L[7])

x = [ide,name,dob,age,sex,rel,adp,state]

print" "

print" Identity number: ",x[0]

print" Name of child : ",x[1]

print" Date of birth : ",x[2]

print" Age of child : ",x[3]

print" Sex of child : ",x[4]

print" Relations : ",x[5]

print" Phone number : ",x[6]

print" Native state : ",x[7]

wFile2=open('Hope.Add2.CSV','r')

adult\_list = wFile2.readlines()

wFile2.close()

if not os.path.exists('Hope.Add2.CSV'):

print"File not found"

return

for record in adult\_list:

L = record.rstrip('\n').split(',')

if name == (L[0]):

found=True

ide=(L[1])

dob=(L[2])

age=(L[3])

sex=(L[4])

rel=(L[5])

adp=(L[6])

state=(L[7])

work=(L[8])

edu=(L[9])

hobby=(L[10])

x = [ide,name,dob,age,sex,rel,adp,state,work,edu,hobby]

print" "

print" Identity number : ",x[0]

print" Name of adult : ",x[1]

print" Date of birth : ",x[2]

print" Age of adult : ",x[3]

print" Sex of adult : ",x[4]

print" Relations : ",x[5]

print" Phone number : ",x[6]

print" Native state : ",x[7]

print" Occupation : ",x[8]

print" Academic Qualification: ",x[9]

print" Interests and hobbies : ",x[10]

def Updatechild(self):

#Offers platform for updation of the values of certain arguments of

a child member.

if not os.path.exists('Hope.Add1.CSV'):

print"File not found..."

return

rfile = open('Hope.Add1.CSV','r')

tfile = open('Temp.CSV','w')

name = raw\_input("Enter the name to be updated: ")

found = False

for record in rfile:

L = record.split(',')

if name == str(L[0]):

found = True

age = int(input("Enter the age: "))

rel = raw\_input("Enter the relations: ")

phno = int(input("Enter the phone number: "))

tfile.write(L[0]+','+(str(L[1]))+','+L[2]+','+(str(age))+','+L[4]+','+rel+','+(str(phno))+','+L[7])

else:

tfile.write(record)

rfile.close()

tfile.close()

if found:

print"Member ",name," updated successfully..."

os.remove('Hope.Add1.CSV')

os.rename('Temp.CSV','Hope.Add1.CSV')

else:

print"Member ",name," not found..."

def Updateadult(self):

#Offers platform for updation of the values of certain arguments of

an adult member.

if not os.path.exists('Hope.Add2.CSV'):

print"File not found..."

return

rfile = open('Hope.Add2.CSV','r')

tfile = open('Temp.CSV','w')

name = raw\_input("Enter the name to be updated: ")

found = False

for record in rfile:

L = record.split(',')

if name == str(L[0]):

found = True

age = int(input("Enter the age: "))

rel = raw\_input("Enter the relations: ")

phno = int(input("Enter the phone number: "))

occ = raw\_input("Enter occupation: ")

acaq = raw\_input("Enter academic qualifications: ")

inth = raw\_input("Enter interests and hobbies: ")

tfile.write(L[0]+','+(str(L[1]))+','+L[2]+','+(str(age))+','+L[4]+','+rel+','+(str(phno))+','+L[7]+','+occ+','+acaq+','+inth)

else:

tfile.write(record)

rfile.close()

tfile.close()

if found:

print"Member ",name," updated successfully..."

os.remove('Hope.Add2.CSV')

os.rename('Temp.CSV','Hope.Add2.CSV')

else:

print"Member ",name," not found..."

def Deletechild(self):

#Offers platform for deletion of all data of a child member.

if not os.path.exists('Hope.Add1.CSV'):

print"File not found."

return

rfile = open('Hope.Add1.CSV','r')

tfile = open('Temp.CSV','w')

name = raw\_input("Enter the name you wish to delete: ")

found = False

for record in rfile:

L = record.split(',')

if name == str(L[0]):

found = True

else:

tfile.write(record)

rfile.close()

tfile.close()

if found:

print"Member ",name," deleted successfully..."

os.remove('Hope.Add1.CSV')

os.rename('Temp.CSV','Hope.Add1.CSV')

else:

print"Member ",name," not found..."

def Deleteadult(self):

#Offers platform for deletion of all data of an adult member.

if not os.path.exists('Hope.Add2.CSV'):

print"File not found."

return

rfile = open('Hope.Add2.CSV','r')

tfile = open('Temp.CSV','w')

name = raw\_input("Enter the name you wish to delete: ")

found = False

for record in rfile:

L = record.split(',')

if name == str(L[0]):

found = True

else:

tfile.write(record)

rfile.close()

tfile.close()

if found:

print"Member ",name," deleted successfully..."

os.remove('Hope.Add2.CSV')

os.rename('Temp.CSV','Hope.Add2.CSV')

else:

print"Member ",name," not found..."

import os

class Track:

#Helps in the tracking of details of all members by the maintenance

department(admin staff).

def track(self):

#Helps maintenance department access data of a particular

member.

wFile1=open('Hope.Add1.CSV','r')

name=raw\_input("Enter the name of the member: ")

if not os.path.exists('Hope.Add1.CSV'):

print"File not found"

return

for record in wFile1:

L=record.split(',')

if name == (L[0]):

found=True

dob=(L[1])

age=(L[2])

sex=(L[3])

rel=(L[4])

adp=(L[5])

state=(L[6])

entry=(L[7])

print[name,dob,age,sex,rel,adp,state,entry]

wFile1.close()

wFile2=open('Hope.Add2.CSV','r')

if not os.path.exists('Hope.Add2.CSV'):

print"File not found"

return

for record in wFile2:

L=record.split(',')

if name == (L[0]):

found=True

name=(L[0])

dob=(L[1])

age=(L[2])

sex=(L[3])

rel=(L[4])

adp=(L[5])

state=(L[6])

work=(L[7])

edu=(L[8])

hobby=(L[9])

entry=(L[10])

print[name,dob,age,sex,rel,adp,state,work,edu,hobby,entry]

wFile2.close()

import os

class Feedback:

#Serves as a platform for the maintenance department to communicate

details pertaining to a child member to his or her guardian.

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

#Constructor function: Initialises the arguments of current object.

self.name=None

self.imp=None

self.app=None

self.sugg=None

self.comm=None

self.rate=None

self.math=None

self.lang=None

self.art=None

self.sci=None

def inp(self):

#Platform for admin staff: Entering and storing feedback of a child.

self.name=raw\_input("Name of the student: ")

self.app=raw\_input("Appreciation: ")

self.sugg=raw\_input("Suggestion: ")

self.comm=raw\_input("Comment: ")

self.rate=raw\_input("Rating of the student: ")

wFile1=open('Hope.feedback.CSV','a')

wFile1.write(self.name+','+self.app+','+self.sugg+','+\

self.comm+','+self.rate+'\n')

wFile1.close()

print "Data recorded successfully"

def marks(self):

#Platform for admin staff: Entering and storing marks of a child.

self.name = raw\_input("Enter student name: ")

self.math = raw\_input("Marks scored in maths: ")

self.lang = raw\_input("Marks scored in languages: ")

self.art = raw\_input("Marks scored in art: ")

self.sci = raw\_input("Marks scored in science: ")

wFile1=open('Hope.marks.CSV','a')

wFile1.write(self.name+','+self.math+','+self.lang+','+\

self.art+','+self.sci+'\n')

wFile1.close()

print "Data recorded successfully"

def displayinp(self):

#Platform for customer: Viewing stored details of feedback of a

child

wFile1=open('Hope.feedback.CSV','r')

thelist = wFile1.readlines()

wFile1.close()

name=raw\_input("Enter the name of the member whose details

you want to view: ")

if not os.path.exists('Hope.feedback.CSV'):

print"File not found"

return

for record in thelist:

L = record.rstrip('\n').split(',')

if str(name) == str(L[0]):

found=True

app=(L[1])

sugg=(L[2])

comm=(L[3])

rate=(L[4])

x = [name,app,sugg,comm,rate]

print"STUDENT FEEDBACK REPORT "

print" Name of child : ",x[0]

print" Appreciation : ",x[1]

print" Suggestions : ",x[2]

print" Comments : ",x[3]

print" Rate out of 10 : ",x[4]

def displaymarks(self):

#Platform for customer: Viewing stored details of marks of a child.

wFile1=open('Hope.marks.CSV','r')

thelist = wFile1.readlines()

wFile1.close()

name=raw\_input("Enter the name of the member whose marks you

want to view: ")

if not os.path.exists('Hope.marks.CSV'):

print"File not found"

return

for record in thelist:

L = record.rstrip('\n').split(',')

if str(name) == str(L[0]):

found=True

math=(L[1])

lang=(L[2])

art=(L[3])

sci=(L[4])

x = [name,math,lang,art,sci]

print"STUDENT FEEDBACK REPORT "

print" Name of child : ",x[0]

print" Maths marks : ",x[1]

print" Language marks : ",x[2]

print" Art marks : ",x[3]

print" Science marks : ",x[4]

import os

class Bind(Add,Existence,Track,Feedback):

#Derived class: Inherits the properties of all the previously defined classes,

binding the methods in one master unit

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

Add.\_\_init\_\_(self)

def addChild(self):

Add.addChild(self)

def addAdult(self):

Add.addAdult(self)

def query(self):

Existence.View(self)

def updatechild(self):

Existence.Updatechild(self)

def updateadult(self):

Existence.Updateadult(self)

def deletechild(self):

Existence.Deletechild(self)

def deleteadult(self):

Existence.Deleteadult(self)

def track(self):

Track.track(self)

def inputfeedback(self):

Feedback.inp(self)

def inputmarks(self):

Feedback.marks(self)

def feedback(self):

Feedback.displayinp(self)

def marks(self):

Feedback.displaymarks(self)

#Main environment

b = Bind()

print("WELCOME TO HOPE ORPHANAGE...HOMAGE MANAGEMENT SYSTEM")

print("")

print("INTRODUCTION")

print("HOPE Orphanage is a global charity that serves the poor in \

more than 20 states. It is a registered in some other states.\

HOPE Orphanage began its work in India in 2000. Its program \

has expanded from mobile clinic. The Vision of the Orphanage is \

“Bringing Hope....Changing Lives”. HOME Management system \

is an online aid for registration of orphans.")

print("")

while True:

print("1. If you are a guardian, enter 1...")

print("2. If you are an admin member, enter 2...")

print("3. If you wish to exit the system, enter 3...")

c = int(input("Enter your choice: "))

if c==1:

print("Welcome dear visitor, please select from the following...")

print("1. To register a new ward, enter 1...")

print("2. To view details of an existing ward, enter 2")

print("3. To update details of an existing ward, enter 3...")

print("4. To delete details of an existing ward, enter 4...")

print("5. To view the feedback/marks of your ward(child), enter 5...")

print("6. To exit the system, enter 6...")

ch = int(input("Enter your choice: "))

if ch==1:

print("If you want to register a child(age<18), enter 1...")

print("If you want to register an adult(age>=18), enter 2...")

ch1 =int(input("Enter your choice: "))

if ch1==1:

b.addChild()

elif ch1==2:

b.addAdult()

else:

print("Invalid Entry...try again!")

elif ch==2:

b.query()

elif ch==3:

print("If ward in question is a child, enter 1...")

print("If ward in question is an adult, enter 2...")

ch2 = int(input("Enter your choice: "))

if ch2==1:

b.updatechild()

elif ch2==2:

b.updateadult()

else:

print("Invalid Entry...try again!")

elif ch==4:

print("If ward in question is a child, enter 1...")

print("If ward in question is an adult, enter 2...")

ch3 = int(input("Enter your choice: "))

if ch3==1:

b.deletechild()

elif ch3==2:

b.deleteadult()

else:

print("Invalid Entry...try again!")

elif ch==5:

print("To view the feedback report of your child, enter 1...")

print("To view the academic progress/marks of your child, enter

2...")

ch4 = int(input("Enter your choice: "))

if ch4==1:

b.feedback()

elif ch4==2:

b.marks()

else:

print("Invalid Entry...try again!")

elif ch==6:

print("Exiting HOPE Orphanage...THANKYOU FOR VISITING!!")

break

else:

print("Invalid Entry...try again!")

elif c==2:

p = raw\_input("Enter the admin password: ")

if str(p)=="adminalka":

print("Welcome to admin...")

print("1. To track member details, enter 1...")

print("2. To input a certain child's feedback, enter 2...")

print("3. To input a certain child's marks, enter 3...")

print("4. To exit from system, enter 4...")

ce = int(input("Enter your choice: "))

if ce==1:

b.track()

elif ce==2:

b.inputfeedback()

elif ce==3:

b.inputmarks()

elif ce==4:

print("Exiting HOPE Orphanage...")

break

else:

print("Invalid Entry...try again!")

else:

print("Wrong password...retry!")

elif c==3:

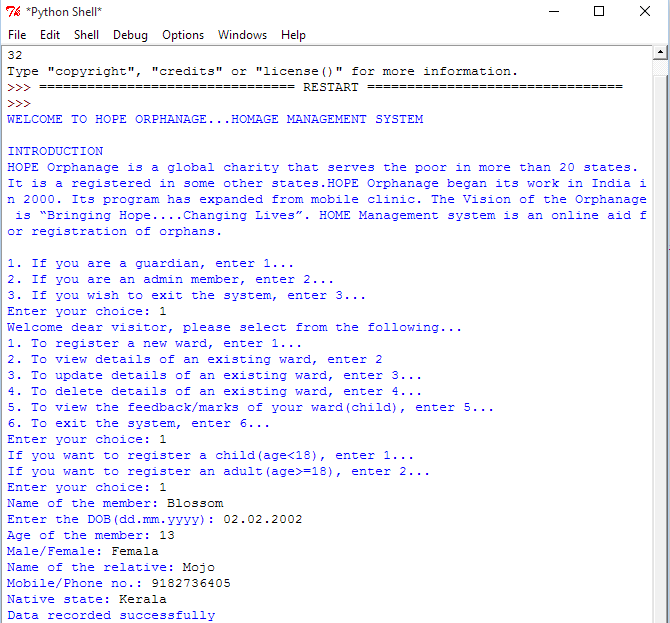
print("Exiting HOPE Orphanage...THANKYOU FOR VISITING!!")

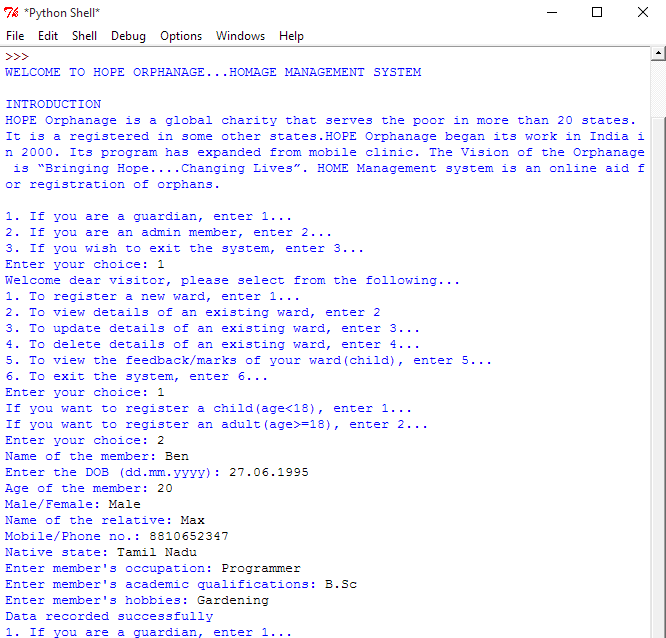
break

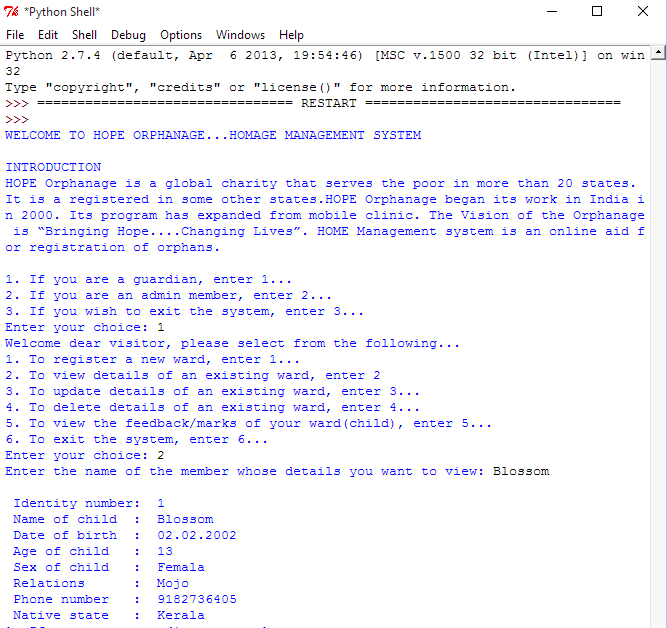
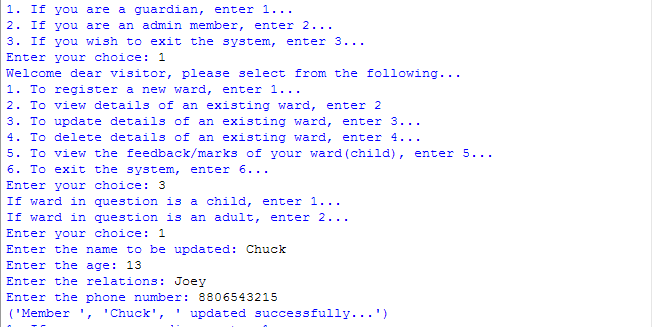
else:

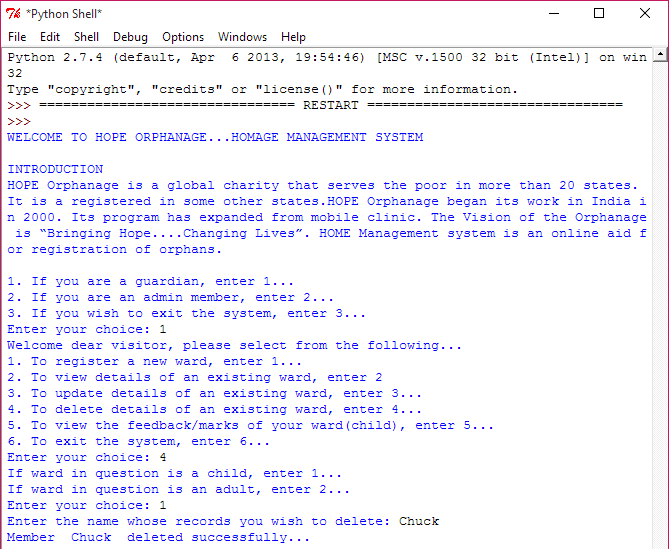
print("Invalid Entry...try again!")

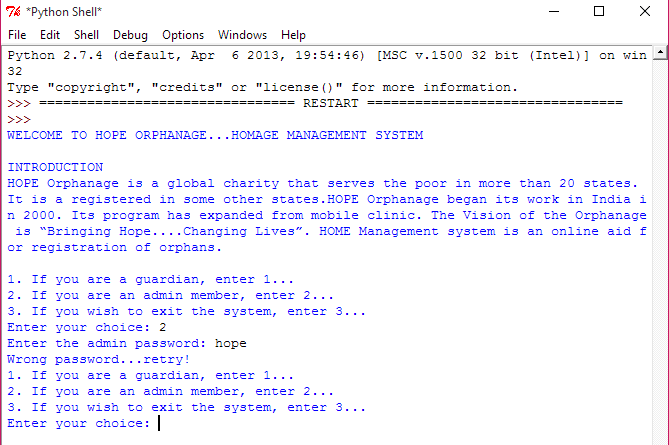
**\_SCREENSHOTS\_**

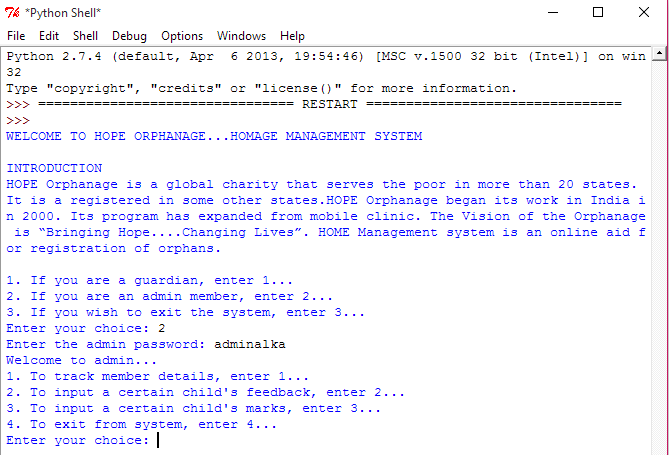
****

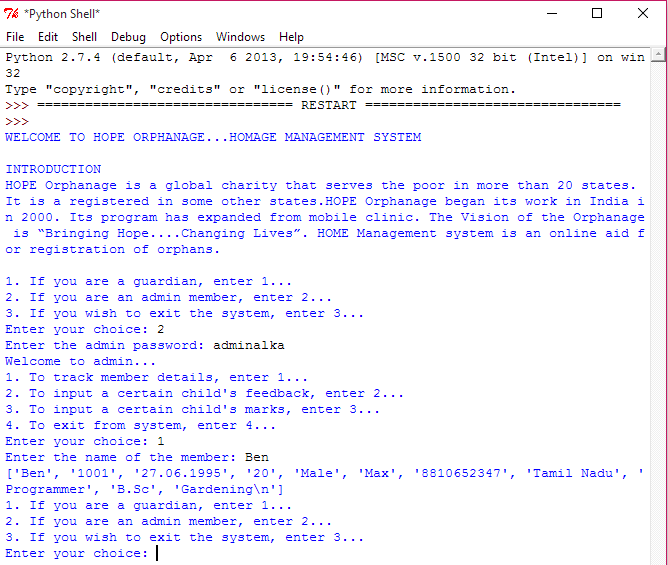
****

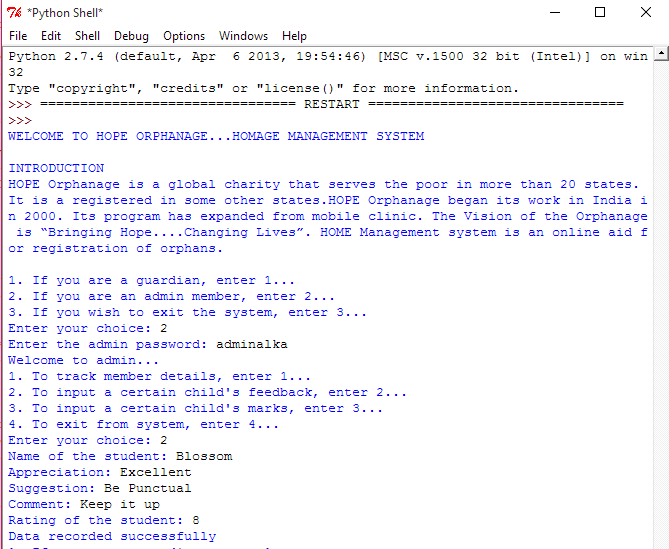
****

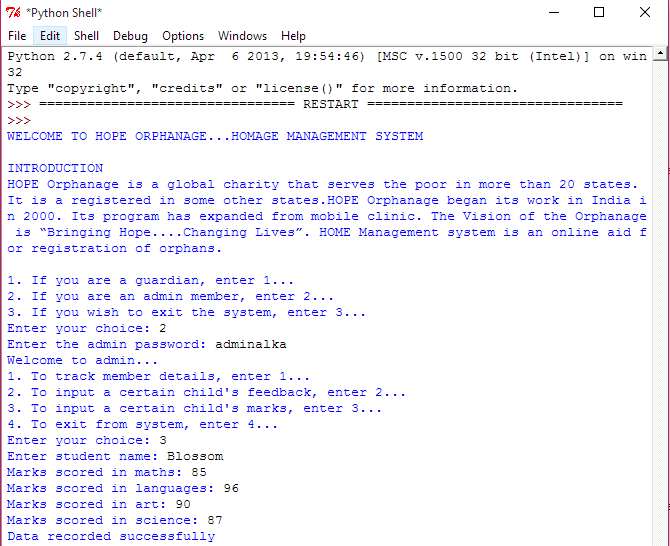
****

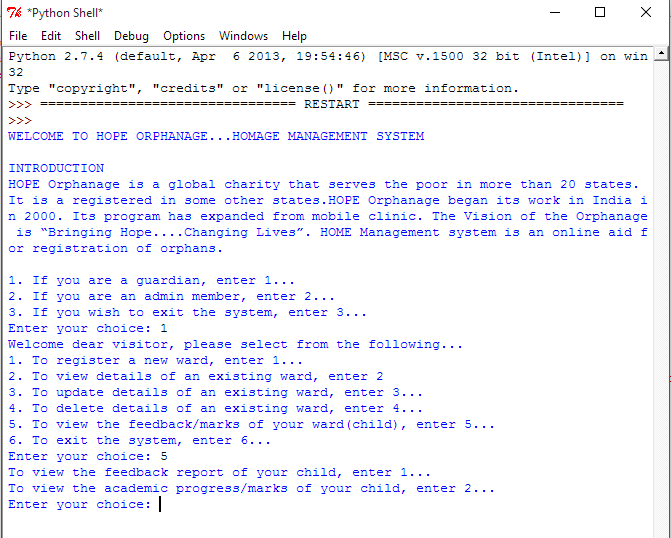
****

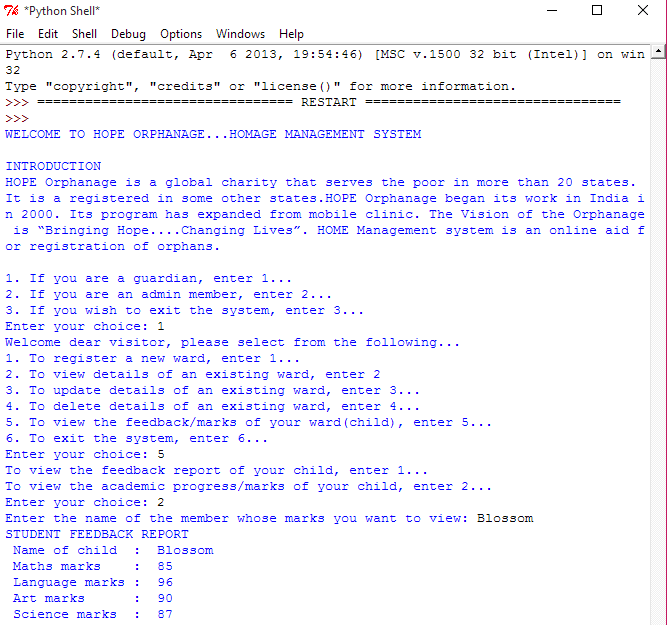
****

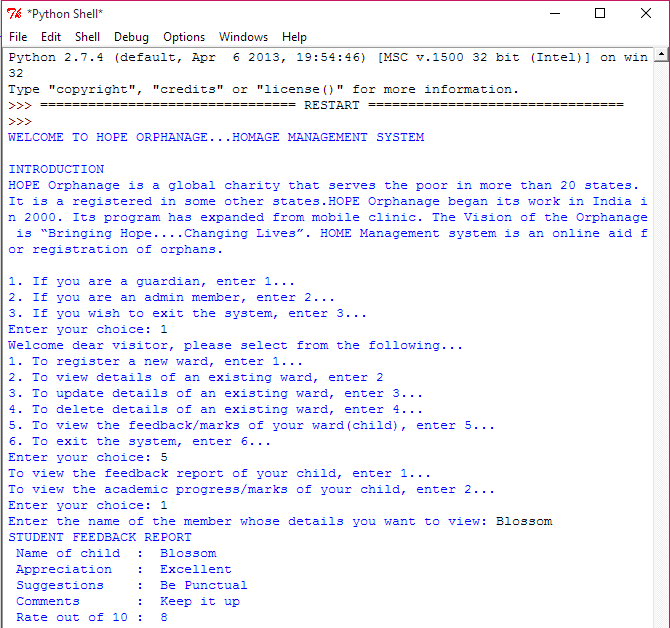
****

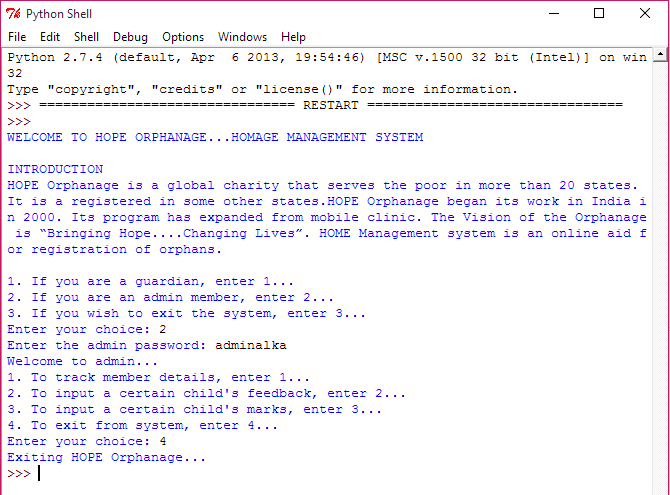
****

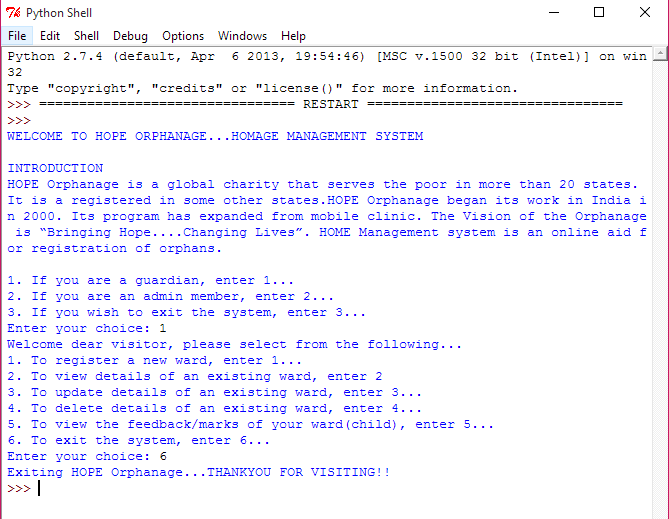
****

****

****

****

****

****

**BIBLIOGRAPHY**

1. Python Programming: An Introduction to

Computer Science by John Zelle

2. Python in a Nutshell by Alex Martelli