01\_

MAX = 3

def enQueue(*Queue*):

*if* (len(Queue) == MAX):

        print("Queue overflow")

*else*:

        item = int(input("Enter item to enqueue: "))

        Queue.append(item)

def dispQueue(*Queue*):

*if*(len(Queue) == 0):

        print("Queue Empty")

*else*:

*for* item *in* Queue:

            print(item, *end*='\t')

def deQueue(*Queue*):

*if*(len(Queue) == 0):

        print("Queue Underflow")

*else*:

*return* Queue.pop(0)

def qFront(*Queue*):

*if*(len(Queue) == 0):

        print("Queue Empty")

*else*:

*return* Queue[0]

def qRear(*Queue*):

*if*(len(Queue) == 0):

        print("Queue Empty")

*else*:

*return* Queue[len(Queue)-1]

def main():

    Queue=[]

*while* True:

        print("1.EnQueue\t2.DeQueue\t3.DispQueue\t4.QFront\t5.QRear\t6.Exit")

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

*if*(ch == 1):

            enQueue(Queue)

*elif*(ch == 2):

            print("Dequeued item is: ", deQueue(Queue))

*elif*(ch == 3):

            dispQueue(Queue)

*elif*(ch == 4):

            print("Queue front is: ", qFront(Queue))

*elif*(ch == 5):

            print("Queue rear is: ", qRear(Queue))

*elif*(ch == 6):

            exit()

*else*:

            print("Invalid input, please try again")

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

    main()

02\_

def add(*Courses*):

    code,cname,fname,regs = input("Enter Code,Coursename,faculty,no\_of\_regs: ").split(',')

    regs = int(regs)

*if* code *in* *Courses*:

        print("Duplicates are not allowed")

*else*:

*Courses*[code] = [cname,fname,regs]

def dispAll(*Courses*):

*if* len(*Courses*) == 0:

        print("Courses dictionary is empty")

*else*:

*for* code,details *in* *Courses*.items():

            print(code,details[0],details[1],details[2])

def disp(*Courses*):

    ccode = input("Enter course code to display: ")

*if* ccode *in* *Courses*:

        print(ccode, *Courses*[ccode])

*else*:

        print("Course not present")

def highest(*Courses*):

*if* len(*Courses*) == 0:

        print("Courses dictionary is empty")

*return*

    max = 0

*for* code,details *in* *Courses*.items():

*if* details[2] > max:

            max = details[2]

    hCourses = []

*for* code,details *in* *Courses*.items():

*if* details[2] == max:

            hCourses.append([code,details])

    print("Course with highest registration:\n", hCourses)

def main():

    Courses = {}

*while* True:

        opt = int(input("1: CourseAdd 2: DisplayCourse 3: DispAll 4: Highest 5:Exit \nEnter your optiion:"))

*if* opt == 1:

            add(Courses)

*elif* opt == 2:

            disp(Courses)

*elif* opt == 3:

            dispAll(Courses)

*elif* opt == 4:

            highest(Courses)

*elif* opt == 5:

            exit()

*else*:

            print("Invalid option")

*break*

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

  main()

03\_

*import* csv

def read(*n*):

    books=[]

*for* i *in* range(*n*) :

        no,title,author,price=input("Enter number title author and price: ").split(',')

        price=float(price)

        books.append([no,title,author,price])

*with* open("Books.csv","w",*newline*='') *as* f:

        w=csv.writer(f)

        w.writerows(books)

    print( "Books are: ",books)

def AuthorSearch():

    A=input("Enter author to search: ")

    flag=0

*with* open("Books.csv") *as* f:

        r=csv.reader(f)

*for* book *in* r:

*if* A== book[2]:

                flag=1

                print("Book Details are: ",book[0],book[1],book[2],book[3])

*if* flag==0:

            print("Book not found ")

def TitleSearch():

    A=input("Enter title to search: ")

    flag=0

*with* open("Books.csv") *as* f:

        r=csv.reader(f)

*for* book *in* r:

*if* A== book[1]:

                flag=1

                print("Book Details are: ",book[0],book[1],book[2],book[3])

*if* flag==0:

            print("Book not found ")

def PriceSearch():

*try*:

        price=float(input("Enter price to search the books under that price2,Wit,Des Machale,500: "))

*if* price<=0:

*raise* ValueError("Invalid Price")

*else*:

            flag=0

*with* open("Books.csv") *as* f:

                r=csv.reader(f)

*for* book *in* r:

*if* price > float(book[3]) :

                        flag=1

                        print("Book Details are: ",book[0],book[1],book[2],book[3])

*if* flag==0:

                    print("Book not found ")

*except* ValueError *as* e:

        print("ValueError",e,type(e))

def main():

    n=int(input("Enter the number of books: "))

    read(n)

*while* True:

        opt=int(input("\n1: Author Search  2:Price Search  3:Title Search  4:Exit\nEnter the option: "))

*if* opt==1:

            AuthorSearch()

*elif* opt==2:

            PriceSearch()

*elif* opt==3:

            TitleSearch()

*else*:

*break*

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

    main()

04\_

*# termwork\_4*

class Person:

    def \_\_init\_\_(*self*, *fname*, *lname*, *email*):

*self*.fname = *fname*

*self*.lname = *lname*

*self*.email = *email*

    def disp(*self*):

*# print("Person:")*

        print(*self*.fname, *self*.lname, *self*.email)

class Customer(Person):

    def \_\_init\_\_(*self*, *fname*, *lname*, *email*, *no*):

        super().\_\_init\_\_(*fname*, *lname*, *email*)  *# calling person's init*

*self*.no = *no*

    def disp(*self*):

        Person.disp(*self*)

        print("Customer No: ", *self*.no)

class Employee(Person):

    def \_\_init\_\_(*self*, *fname*, *lname*, *email*, *pan*):

        super().\_\_init\_\_(*fname*, *lname*, *email*)  *# calling person's init*

*self*.pan = *pan*

    def disp(*self*):

        Person.disp(*self*)

        print("Pan No: ", *self*.pan)

def show(*ob*):

*if* isinstance(*ob*, Customer):

        print("Customer: ")

*ob*.disp()

*elif* isinstance(*ob*, Employee):

        print("Employee: ")

*ob*.disp()

*elif* isinstance(*ob*, Person):

        print("Person: ")

*ob*.disp()

def main():

    choice = 'y'

*while* choice == 'y':

        fname, lname, email = input("Enter first name, last name and email: ").split()

        print("1:Customer 2:Employee 3:Person")

        opt = int(input("Enter your option: "))

*if* opt == 1:

            no = int(input("Enter the customer no: "))

            c = Customer(fname, lname, email, no)

            show(c)

*elif* opt == 2:

            pan = input("Enter the employee pan: ")

            e = Employee(fname, lname, email, pan)

            show(e)

*elif* opt == 3:

            p = Person(fname, lname, email)

            show(p)

*else*:

            print("Invalid option")

        choice = input("Continue(y/n)?: ")

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

    main()

05\_

*import* sqlite3

*from* contextlib *import* closing

def dbConnect(*file*):

*try*:

        conn = sqlite3.connect(*file*)

*return* conn

*except*:

        print("Dbconnect error")

def dbInsert(*conn*):

    n = int(input("Enter n:"))

*with* closing(*conn*.cursor()) *as* C:

*for* i *in* range(n):

            pid, n, q, p = input("Enter id,name,qty and price: ").split()

            pid = int(pid)

            q = int(q)

            p = float(p)

            query = "insert into products values(?,?,?,?)"

*try*:

                C.execute(query, (pid, n, q, p))

*except*:

                print("DbInsert error")

*conn*.commit()

def dbDisplay(*conn*):

*with* closing(*conn*.cursor()) *as* C:

        query = "Select \* from products"

*try*:

            C.execute(query)

            result =  C.fetchall()

            print("Products:")

*for* line *in* result:

                print(line[0], line[1], line[2], line[3])

*except*:

            print("DbDisplay error")

def dbDelete(*conn*):

    pid = int(input("Enter prodId to delete: "))

*with* closing(*conn*.cursor()) *as* C:

        query = "delete from products where productid=?"

*try*:

            C.execute(query,(pid,))

*conn*.commit()

*except*:

            print("Dbdelete error")

def dbUpdate(*conn*):

*with* closing(*conn*.cursor()) *as* C:

        query = "Update products set price=price+(price\*0.1) where price<50"

*try*:

            C.execute(query)

*conn*.commit()

*except*:

            print("DbUpdate error")

def dbDispLTforty(*conn*):

*with* closing(*conn*.cursor()) *as* C:

        query = "select \* from products where quantity<40"

*try*:

            C.execute(query)

            result =  C.fetchall()

*for* line *in* result:

                print(line[0], line[1], line[2], line[3])

*except*:

            print("DbDisplay less than 40 error")

def main():

    conn = dbConnect("D:\\Srushti\\GIT\\4th sem\\Python\\PROGRAMS\\products.sqlite")

    dbInsert(conn)

    dbDisplay(conn)

    dbDelete(conn)

    print("Products after deleting")

    dbDisplay(conn)

    dbUpdate(conn)

    print("Products after update")

    dbDisplay(conn)

    print("Products whose qty<40:")

    dbDispLTforty(conn)

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

    main()

06\_

*from* tkinter *import* \*

class MyCalc:

    def \_\_init\_\_(*self*,*root*):

        frame1 = Frame(*root*,*bg*="yellow")

        frame1.pack(*fill*=X)

*self*.lbl1 = Label(frame1,*text*="Enter 1st num: ",*fg*="blue",*bg* = "yellow", *font* = (16))

*self*.lbl1.pack(*side* = LEFT, *padx*=25, *pady*=25)

*self*.tbox1 = Entry(frame1,*font*=("arial",12,"bold"),*bg*="white")

*self*.tbox1.pack(*side* = LEFT, *padx*=25, *pady*=25)

        frame2 = Frame(*root*)

        frame2.pack(*fill*=X)

*self*.lbl2 = Label(frame2,*text*="Enter 2nd num: ",*fg*="blue",*bg* = "yellow",*font* = (16))

*self*.lbl2.pack(*side* = LEFT, *padx*=25, *pady*=25)

*self*.tbox2 = Entry(frame2,*font*=("arial",12,"bold"),*bg*="white")

*self*.tbox2.pack(*side* = LEFT, *padx*=25, *pady*=25)

        frame3 = Frame(*root*)

        frame3.pack(*fill*=X)

*self*.lbl3 = Label(frame3,*text*="Result is:           ",*fg*="blue",*bg* = "yellow",*font* = (16))

*self*.lbl3.pack(*side* = LEFT, *padx*=25, *pady*=25)

*self*.tbox3 = Entry(frame3,*font*=("arial",12,"bold"),*bg*="white")

*self*.tbox3.pack(*side* = LEFT, *padx*=25, *pady*=25)

        frame4 = Frame(*root*)

        frame4.pack(*fill*=BOTH)

*self*.btnAdd = Button(frame4,*text*="Sum",*command* = *self*.Add,*font*=("arial",12,"bold"),*bg*="light blue")

*self*.btnAdd.pack(*side* = LEFT, *padx*=25, *pady*=20)

*self*.btnAdd = Button(frame4,*text*="Diff",*command* = *self*.Diff,*font*=("arial",12,"bold"),*bg*="light blue")

*self*.btnAdd.pack(*side* = LEFT, *padx*=25, *pady*=20)

*self*.btnProd = Button(frame4,*text*="Prod",*command* = *self*.Prod,*font*=("arial",12,"bold"),*bg*="light blue")

*self*.btnProd.pack(*side* = LEFT, *padx*=25, *pady*=20)

*self*.btnProd = Button(frame4,*text*="Div",*command* = *self*.Div,*font*=("arial",12,"bold"),*bg*="light blue")

*self*.btnProd.pack(*side* = LEFT, *padx*=25, *pady*=20)

    def Add(*self*):

*self*.tbox3.delete(0,END)

        num1= int(*self*.tbox1.get())

        num2= int(*self*.tbox2.get())

*self*.tbox3.insert(END, str(num1+num2))

    def Diff(*self*):

*self*.tbox3.delete(0,END)

        num1= int(*self*.tbox1.get())

        num2= int(*self*.tbox2.get())

*self*.tbox3.insert(END, str(num1- num2))

    def Prod(*self*):

*self*.tbox3.delete(0,END)

        num1= int(*self*.tbox1.get())

        num2= int(*self*.tbox2.get())

*self*.tbox3.insert(END, str(num1\* num2))

    def Div(*self*):

*self*.tbox3.delete(0,END)

        num1= int(*self*.tbox1.get())

        num2= int(*self*.tbox2.get())

*self*.tbox3.insert(END, str(num1/ num2))

def main():

    root = Tk()

    root.title("Calculator")

    root.geometry("400x400")

    root.config(*bg*="white")

    myCalc = MyCalc(root)

    root.mainloop()

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

    main()

07\_

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

    myFile = open("Termwork7.txt")

    allLines = myFile.read().splitlines()

    avgMarks = 0

    maxAvg = 0

*for* line *in* allLines:

        (name,m1,m2,m3,m4,usn)=map(str,line.split(","))

        avgMarks = (int(m1)+int(m2)+int(m3))/3

        print(usn, "\t",m1,"\t",m2,"\t",m3,"\t",name)

        print()

*if*(avgMarks > maxAvg):

            maxAvg = avgMarks

print("Max class %5.2f" %maxAvg)

08\_

*import* pandas *as* pd

*import* numpy *as* np

myData = pd.read\_csv("Termwork8.csv")

print(myData.to\_string())

print(".....table columns......")

print("===============================")

*for* col *in* myData.columns:

    print(col+"\t\t",*end*="")

colName = input("Enter col name: ")

print(myData.loc[0:10,colName].unique())

print(myData.groupby(colName)[colName].count())

print("Total rows in DFrame: "+str(myData.shape[0]))