**VANASTHALI PUBLIC**

**SCHOOL**



**Computer Science Program File**

INDEX

|  |  |  |
| --- | --- | --- |
| **SR NO .** | **PROGRAMS** | **PAGE**  **NO.** |
| **1.** | **Program to calculate simple interest using a function interest() that can receive principal amount , time and rate and returns calculated simple interest. Do specify default values for rate and time as 10% and 2 years respectively.** |  |
| **2.** | **Given a dictionary with values list, extract key whose value has most unique values.** |  |
| **3.** | **Write a python program that displays a first three rows fetched from student table of MySQL database “supermarket”.** |  |
| **4.** | **Write a program to print the following pattern** |  |
| **5.** | **Define a function to check if the elements in the first half of a tuple are sorted in ascending order or not.** |  |
| **6.** | **Program to count the frequency of a list of elements using a dictionary** |  |
| **7.** | **Write a program to create a dictionary with the roll number, name and marks of n students in a class and display the names of students who have marks above 75.** |  |
| **8.** | **Read a text file x.txt and built y.txt that should be reverse of x. txt** |  |
| **9.** | **Program to sort a sequence using insertion sort.** |  |
| **10.** | **Program to sort a list using bubble sort.** |  |
| **11.** | **Write a program that inputs main string and then creates an encrypted string by embedding a short symbol based string after each character . The program should also be able to produce the decrypted string from encrypted string.** |  |

|  |  |  |
| --- | --- | --- |
| **12.** | **Define a function having a text file name as it's argument. The function should return a Dictionary having key as line number of text file and values list of two integers [number of uppercase alphabets and number of lowercase alphabet in each line.** |  |
| **13.** | **Write a program to read a text file line by line and display each word separated by a ‘#’.** |  |
| **14.** | **Write a program to read a text file and display the count of vowels and consonants in the file.** |  |
| **15.** | **Write a program to get student data (roll no, name and marks) from user and write onto a binary file. The program should be able to get data from the user and write onto the file as long as the user wants.** |  |
| **16.** | **Write a program to append student records to file created in previous program, by getting data from user.** |  |
| **17.** | **Write a program to open file Stu.dat and search for records with roll numbers as 46 or 48 . If found display the records.** |  |
| **18.** | **Python Program to implement stack operations.** |  |
| **19.** | **Define a function reading few lines from the user in a list until an empty line is given as input and count the lines contains a word cat in it.** |  |
| **20.** | **Write a program to read following details of sport’s performance (sport , competitions, prizes-won) of your school and store into a csv file delimited with tab character .** |  |

**Q. Program to calculate simple interest using a function interest() that can receive principal amount , time and rate and returns calculated simple interest. Do specify default values for rate and time as 10% and 2 years respectively.**

**CODE:-**

def interest(principal,time=2,rate=0.10):

return principal\*rate\*time

#\_\_main\_\_

Prin=float(input('Enter principal amount:'))

print("Simple interest with default ROI and time value is:")

Si1=interest(Prin)

print("Rs.",Si1)

Roi=float(input("Enter rate of interest:"))

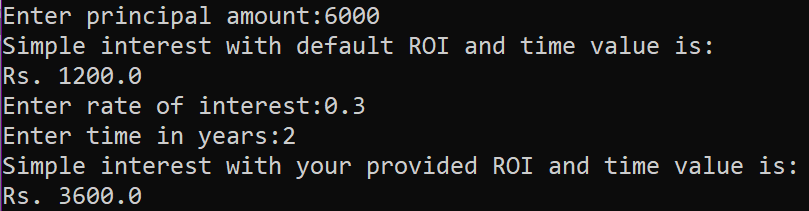
time=int(input("Enter time in years:"))

print("Simple interest with your provided ROI and time value is:")

Si2=interest(Prin,time,Roi)

print("Rs.",Si2)

**OUTPUT:-**



**Q. Given a dictionary with values list, extract key whose value has most unique values.**

**CODE:-**

test\_dict = {"Gfg" : [5, 7, 5, 4, 5],

"HANG" : [6, 7, 4, 3, 3],

"Best" : [9, 9, 6, 5, 5]}

print("The original dictionary is : " + str(test\_dict))

max\_val=0

max\_key=None

for sub in test\_dict:

if len(set(test\_dict[sub]))>max\_val:

max\_val=len(set(test\_dict[sub]))

max\_key=sub

print("Key with maximum unique values : "+ str(max\_key))

**OUTPUT:**

****

**Q. Define a function having a variable as its argument and check whether the string is palindrome or not.**

**CODE:-**

def pallindrome(string):

newstr=""

for w in range (-1,-len(string)-1,-1):

newstr=newstr+string[w]

if newstr==string:

print(string,"is a pallindrome")

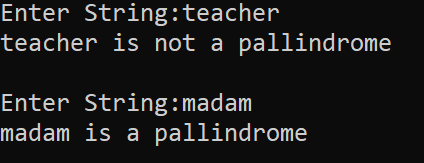
else:

print(string,"is not a pallindrome")

var=input("Enter String:")

pallindrome(var)

**OUTPUT:-**



**Q. Define a function to input a number and check whether the given number is Armstrong number or not .**

**(Note: If a 3 digit number is equal to the sum of the cubes of its each digit , then it is an Armstrong number )**

**CODE:-**

def armstrong():

num=int(input("Enter a 3 digit number:"))

su=0

for w in str(num):

su=su+int(w)\*\*3

if num==su:

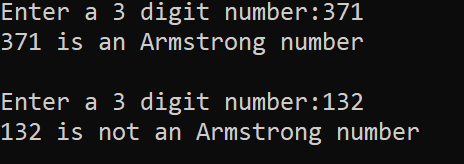
print(num,"is an Armstrong number")

else:

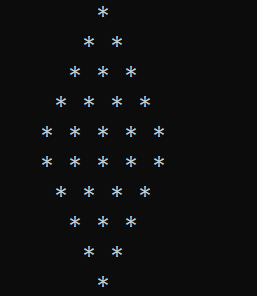
print(num,"is not an Armstrong number")

armstrong()

**OUTPUT:-**



**Q. Write a program to print the following pattern :**



**CODE:-**

n=5 #for number of lines

# upper half

for i in range(1,n+1):

print(" "," "\*(n-i)+"\* "\*i)

# lower half

for i in range(n,0,-1):

print(" "," "\*(n-i)+"\* "\*i)

**Q. Program to find frequencies of all elements of a list .Also , print the list of unique elements in the list and duplicate elements in the given list.**

**CODE:-**

lst=eval(input("Enter list:"))

lstc=list(lst)

uniq=[]

dupl=[]

for element in lst:

c=0

for w in lst:

if element==w:

c=c+1

if c==1:

uniq.append(element)

print('Element',element,'frequency',c)

for q in lstc:

if lst.count(q)==1:

pass

else:

for w in range(len(lstc)-1):

if lstc[w]==q:

print('Element',q,'frequency',lstc.count(q))

dupl.append(q)

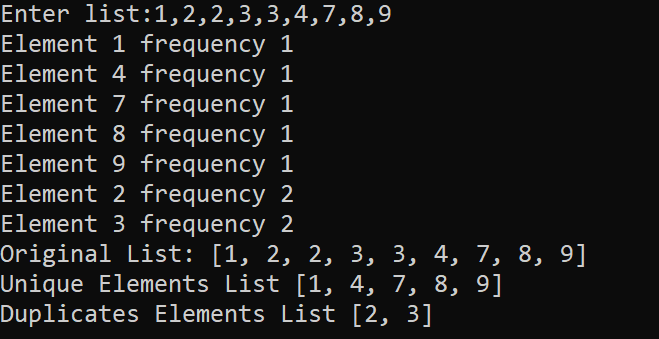
lstc.pop(w)

print('Original List:',list(lst))

print('Unique Elements List',uniq)

print('Duplicates Elements List',dupl)

**OUTPUT**



**Q. Define a function to check if the elements in the first half of a tuple are sorted in ascending order or not.**

**CODE:-**

tup = eval(input("Enter a tuple:"))

ln = len(tup)

mid = ln//2

if ln % 2 == 1:

mid = mid + 1

half = tup[:mid]

if sorted(half) == list(tup[:mid]):

print("First half is sorted")

else:

print("First half is not sorted" def tuplehalf():

tup=eval(input("Enter a tuple:"))

ln=len(tup)

mid=ln//2

if ln%2==1:

mid=mid+1

half=tup[:mid]

if sorted(half)==list(tup[:mid]):

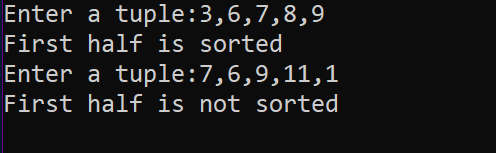
print("First half is sorted")

else:

print("First half is not sorted")

tuplehalf()

**OUTPUT:-**



**Q. Program to count the frequency of a list of elements using a dictionary.**

**CODE:-**

sentence="This is a super idea This idea will change the idea of learning"

words=sentence.split()

d={}

for one in words:

key=one

if key not in d:

count=words.count(key)

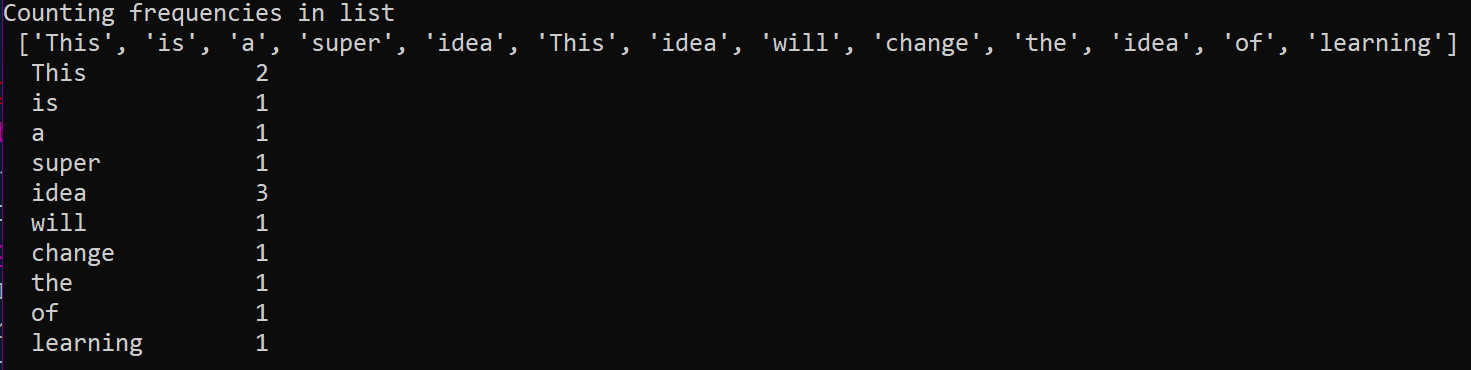
d[key]=count

print("Counting frequencies in list \n",words)

for w in d:

print(" "+w.ljust(15),str(d[w]).ljust(3))

**OUTPUT:-**



**Q. Write a program to create a dictionary with the roll number, name and marks of n students in a class and display the names of students who have marks above 75.**

**CODE:-**

n=int(input("How many Students?"))

stu={}

for i in range(1,n+1):

print()

print("Enter details of student",(i))

rollno=int(input("Roll number:"))

name=input("Name:")

marks=float(input("Marks:"))

d={"Rollno" :rollno,"Name":name,"Marks": marks}

key="Stu"+str(i)

stu[key]=d

print("Students with marks > 75 are:")

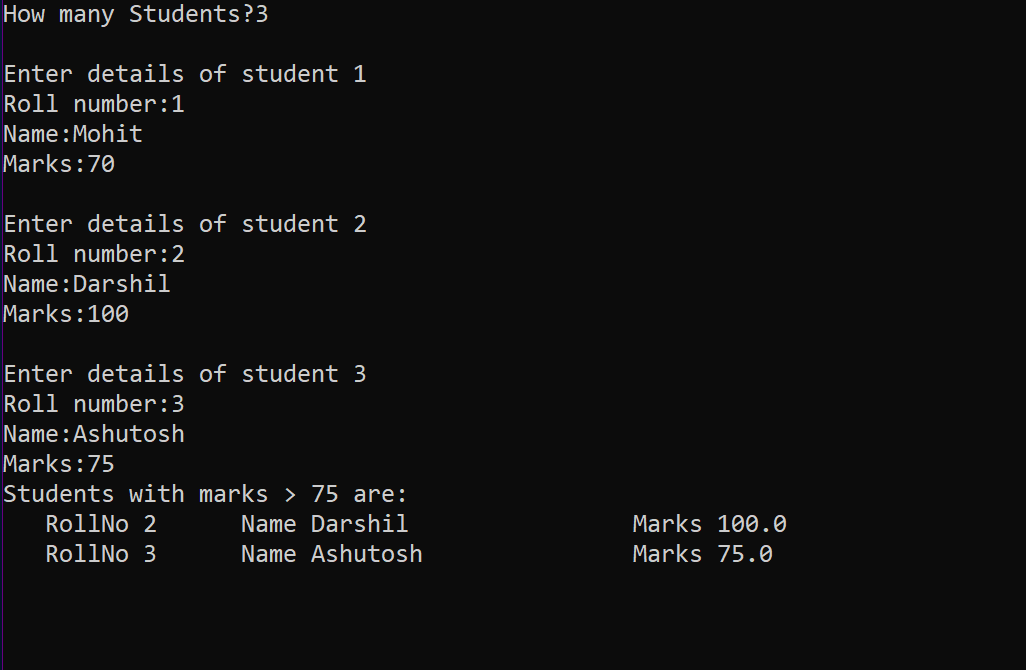
for i in range(1,n+1):

key="Stu"+str(i)

if stu[key]["Marks"]>=75:

print(" RollNo".ljust(8),str(stu[key]["Rollno"]).ljust(3)," Name".ljust(6),str(stu[key]["Name"]).ljust(20)," Marks".ljust(7),str(stu[key]["Marks"]).ljust(20))

**OUTPUT:-**



**Q. Read a text file x.txt and built y.txt that should be reverse of x. txt**

**CODE:-**

def reverse(x,y):

fx=open(x,"r")

fy=open(y,"w+")

k=fx.readlines()

for w in range(-1,-len(k)-1,-1):

x=k[w].rstrip("\n")

line=x

newlin=""

for i in range (-1,-len(line)-1,-1):

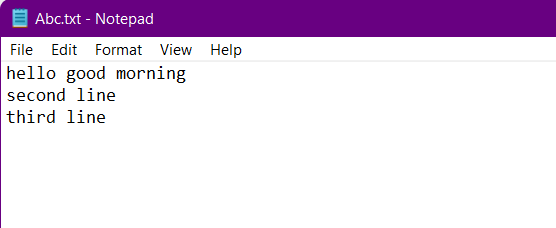
newlin=newlin+line[i]

print(newlin)

fy.write(newlin+"\n")

reverse("Abc.txt","NewAbc.txt")

**OUTPUT:-**





**Q. Program to sort a sequence using insertion sort.**

**CODE:-**

aList = [15,6,13,22,3,52,2]

print("Original list is:",aList)

for i in range(1,len(aList)):

key=aList[i]

j=i-1

while j>=0 and key<aList[j]:

aList[j+1]=aList[j]

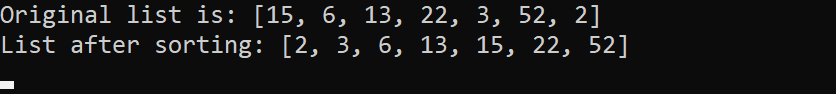
j=j-1

else:

aList[j+1] = key

print("List after sorting:",aList)

**OUTPUT:-**



**Q. Program to sort a list using bubble sort.**

**CODE:-**

alist = [15,6,13,22,3,52,2]

print("original list is:",alist)

n=len(alist)

for i in range(n):

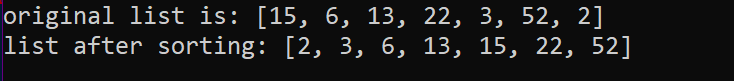
for j in range(0,n-i-1):

if alist[j]>alist[j+1]:

alist[j],alist[j+1]=alist[j+1],alist[j]

print("list after sorting:",alist)

**OUTPUT:-**



**Q. Write a program that inputs main string and then creates an encrypted string by embedding a short symbol based string after each character . The program should also be able to produce the decrypted string from encrypted string.**

**CODE:-**

def encrypt(sttr,enkey):

return enkey.join(sttr)

def decrypt(sttr,enkey):

return sttr.split(enkey)

#-main

mainstring = input("enter main string:")

encryptstr = input("enter encryption key:")

enstr=encrypt(mainstring,encryptstr)

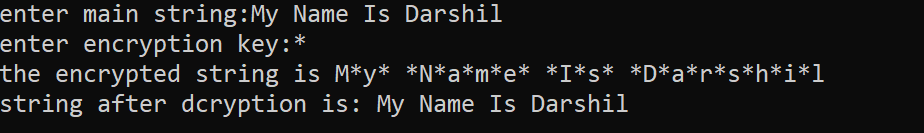
delst=decrypt(enstr,encryptstr)

destr="".join(delst)

print("the encrypted string is",enstr)

print("string after dcryption is:",destr)

**OUTPUT:-**

****

**Q. Define a function having a text file name as it's argument. The function should return a Dictionary having key as line number of text file and values list of two integers [number of upper case alphabets and number of lower case alphabet ] in each line.**

**CODE:-**

def fx(file):

z=0

di={}

f=open(file,"r")

lst=f.readlines()

for line in lst:

z=z+1

uc=0

lc=0

for word in line:

if word.isupper():

uc=uc+1

elif word.islower():

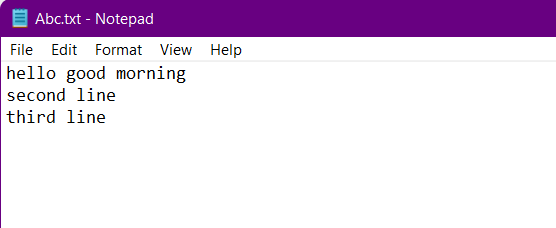
lc=lc+1

di[z]=[lc,uc]

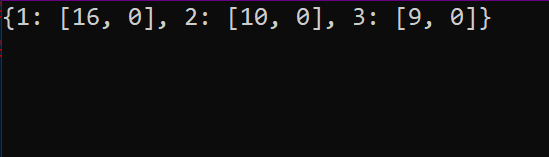
return di

y=fx("Abc.txt")

print(y)



**OUTPUT:-**



**Q. Write a program to read a text file line by line and display each word separated by a ‘#’.**

**CODE:-**

myfile=open("answer.txt","r")

line=" "

while line:

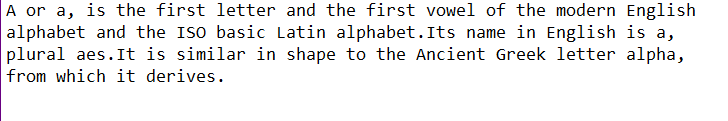
line=myfile.readline()

for word in line.split():

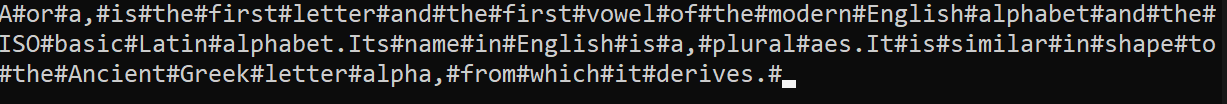
print(word,end='#')

myfile.close()

**FILE CONTENTS**



**OUTPUT:-**



**Q. Write a program to read a text file and display the count of vowels and consonants in the file.**

**CODE:-**

myfile = open("answer.txt","r")

ch=" "

vcount=0

ccount=0

while ch:

ch=myfile.read(1)

if ch in['a','A','e','E','i','I','o','O','u','U']:

vcount=vcount+1

else:

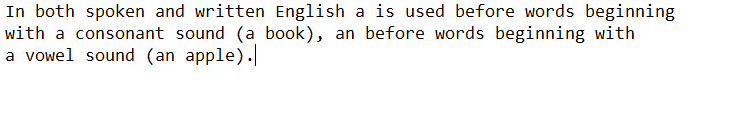
ccount=ccount +1

print("vowels in the file:",vcount)

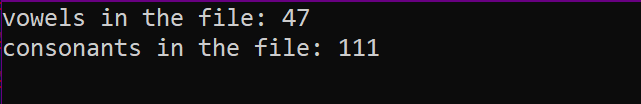
print("consonants in the file:",ccount)

myfile.close()

**FILE CONTENTS**



**OUTPUT:-**



**Q. Write a program to get student data(roll no. , name and marks) from user and write onto a binary file. The program should be able to get data from the user and write onto the file as long as the user wants.**

**CODE:-**

import pickle

stu={}

stufile=open('Stu.dat', 'wb')

ans='y'

while ans=='y' or ans='Y':

rno=int(input("Enter roll number : "))

name=input("Enter name :")

marks=float(input("Enter marks : "))

stu['Rollno']=rno

stu['Name']=name

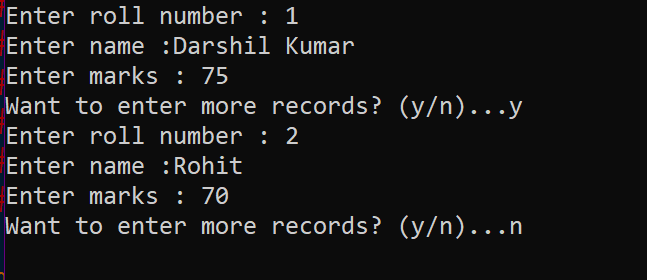
stu['Marks']=marks

pickle.dump(stu, stufile)

ans=input("Want to enter more records? (y/n)...")

stufile.close()

**OUTPUT:-**

****

**Q. Write a program to append student records to file created in previous program, by getting data from user.**

**CODE:-**

import pickle

stu={ }

stufile=open('Stu.dat', 'ab')

ans='y'

while ans=='y' or ans=='Y':

rno=int(input("Enter roll number :"))

name=input("Enter name :")

marks=float( input("Enter marks: "))

stu['Rollno']=rno

stu['Name']=name

stu['Marks']=marks

conf=input('Confirm(y/n):')

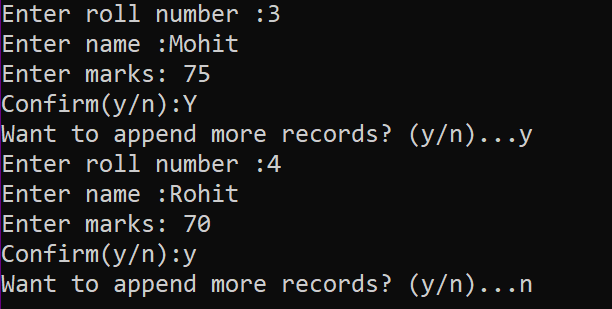
if conf=='Y' or conf=='Y':

pickle.dump(stu,stufile)

ans = input("Want to append more records? (y/n)...")

stufile.close()

**OUTPUT:-**



**Q. Write a program to open file Stu.dat and search for records with roll numbers as 46 or 48 . If found display the records.**

**CODE:-**

import pickle

stu={}

found=False

fin=open('Stu.dat', 'rb')

searchkeys=[46, 48]

try:

print("Searching in File Stu.dat ...")

while True :

stu=pickle.load(fin)

if stu['Rollno'] in searchkeys:

print(stu)

found=True

except EOFError:

if found==False :

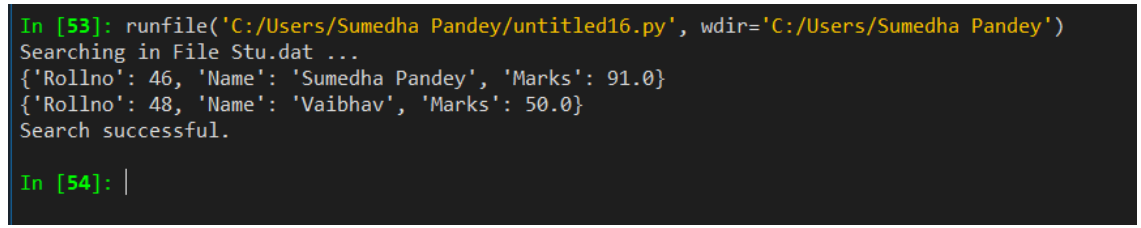
print("No such records found in the file")

else:

print("Search successful.")

fin.close()

**OUTPUT:-**



**Q. Define a function reading few lines from the user in a list until an empty line is given as input and count the lines contains a word cat in it.**

def fx():

k=[]

while True:

x=input("Enter a Line:")

if x=="":

break

else:

k.append(x)

print(k)

c=0

for w in k:

y=w.split()

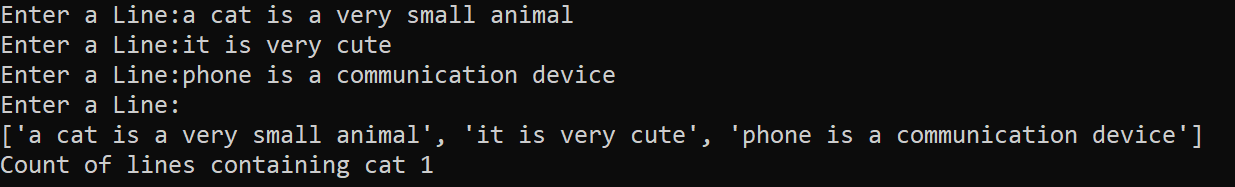
if "cat" in y:

c=c+1

print("Count of lines containing cat",c)

fx()

**OUTPUT:-**



**Q. Write a program to read following details of sport’s performance (sport , competitions, prizes-won) of your school and store into a csv file delimited with tab character .**

**CODE:-**

import csv

fh=open("Sport.csv","w")

writer=csv.writer(fh,delimiter='\t')

writer.writerow(['Sport','Competitions','Prizes won'])

ans='y'

i=1

while ans=='y':

print("Record", i)

sport=input("Sport name:")

comp=int(input("No. of competitions participated :"))

prizes=int(input("Prizes won:"))

srec=[sport,comp,prizes] # create sequence of user data

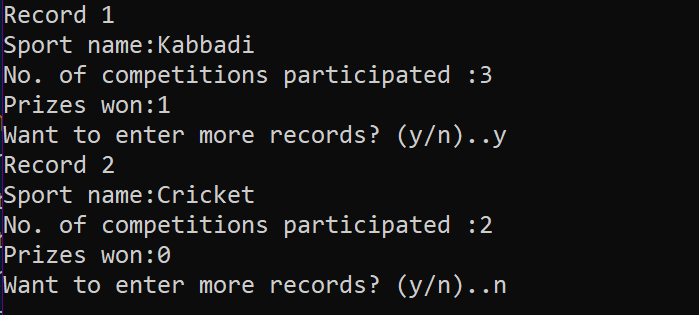
writer.writerow(srec)

i=1+1

ans=input("Want to enter more records? (y/n)..")

fh.close()

**OUTPUT:-**



INDEX

|  |  |  |
| --- | --- | --- |
| **SR NO** | **PROGRAMS** | **PAGE**  **NO.** |
| **1.** | **Write a python program that displays a first three rows fetched from student table of MySQL database “supermarket”.** |  |
| **2.** | **Write a program to build a table named as stationary with attributes Id, name, price, company Id is primary key Add records until user want and display same** |  |
| **3.** | **Write a python databases connectivity program that delete records from sample table of database sample**. |  |
| **4.** | **Define a function named as db to create a new database named as school.** |  |
| **5.** | **Define a function table() to create a new table named as class\_12 with attributes Name, Class, Roll\_no,section and Gender in database named as school.** |  |
| **6.** | **D.efine a function insert to add the details of three students in the table class\_12 in database school** |  |
| **7.** | **Define a function named as update() to update the records of the students taken from user of table class\_12 in database school.** |  |
| **8.** | **Define a function del() to delete the records of the table class\_12 of database school.** |  |
| **9.** | **Define a function display() to display all the records of the table class\_12 of database school.** |  |
| **10.** | **Write a python program to display the records of the student of sec A of table class\_12 in database school.** |  |
| **11.** | **Write a python program to delete the records of the students of sec A from table class\_12 in database school and then display the content of the table .** |  |
| **12.** | **Write a python program that display the first three rows fetched from class\_12 table of MYSQL database school** |  |
| **13.** | **Write a python program that deletes records from class\_12 table of database school that have gender male and then display the contents of the table.** |  |
| **14.** | **Python program to implement stack operations.** |  |
| **15.** | **Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user.Display the content of the Stack along with the largest odd number in the Stack** |  |
| **16.** | **Write a menu driven program that has functions PushS(lst) and PopS(lst) for performing Push and Pop operations with a stack of List containing integers.** |  |
| **17.** | **Write a menu driven program that has functions Make Push (package) and MakePop(package) to add a new Package and delete & Package from a List of Package Description, considering them to act as push and pop operations of the Stack** |  |
| **18.** | **Write a program to implement a stack for these book details(Bookno, book name). That is now each item node of the stack contains two types of information-a bookno and its name.Just implement Push and display operations.** |  |
| **19.** | **Write a program to perform insert and delete operations on a Queue containing Members details as given in the following definition of itemnode.** |  |
| **20.** | **Write a function in Python,INSERTQ(arr,data) and DELETEQ(Arr) for performing insertion and deletion operations in a Queue. arr is the list used for implementing queue and data is the value to be inserted.** |  |
| **21.** | **SQL Queries** |  |

**Q. Write a python program that displays a first three rows fetched from student table of MySQL database “supermarket”.**

**CODE:-**

import mysql.connector as con

c=con.connect(host="localhost",user="root",passwd="123456",database="supermarket")

if c.is\_connected() == False:

print('Error connecting to MySQL database')

crsr=c.cursor()

crsr.execute("select \* from product")

data=crsr.fetchmany(3)

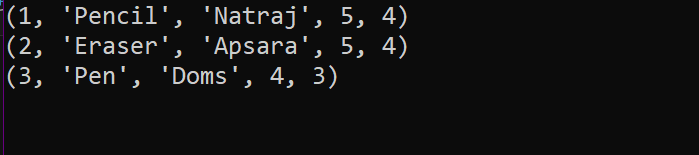
count=crsr.rowcount

for row in data :

print(row)

c.close()

**OUTPUT:-**



**Q.Write a program to build a table named as stationary with attributes Id, name, price, company Id is primary key Add records until user want and display same**

**CODE:-**

z=0

import mysql.connector as con

while z==0:

try:

pswd=input("Enter Password:")

dbobj=con.connect(host="localhost",user="root",password=pswd,charset="utf8")

z=1

except:

print("Wrong Password")

crsr=dbobj.cursor()

def db():

crsr.execute("CREATE DATABASE IF NOT EXISTS SAMPLE;")

crsr.execute("commit")

crsr.execute("USE SAMPLE;")

crsr.execute("commit")

crsr.execute("CREATE TABLE IF NOT EXISTS STATIONARY(Id int Primary key,Name char(20),Price int,Company char(20));")

crsr.execute("commit")

db()

def que():

y=0

print("Enter ID as 0 to exit")

while y==0:

try:

i=int(input("Enter ID:"))

if i==0:

y=1

else:

name=input("Enter Name:")

price=int(input("Enter Price:"))

company=input("Enter Company:")

lst=(str(i),name,str(price),company)

print(lst)

confirm=input("Confirm(Y/N):")

if confirm=="Y" or "y":

crsr.execute("INSERT INTO STATIONARY VALUES ('{}','{}','{}','{}')".format(i,name,price,company))

crsr.execute("commit")

except:

print("Some ERROR")

def dis():

print()

print("-"\*85)

crsr.execute("DESC STATIONARY;")

recs=crsr.fetchall()

print("%3s %-10s %-5s %5s"%(recs[0][0],recs[1][0],recs[2][0],recs[3][0]))

crsr.execute("select \* from STATIONARY;")

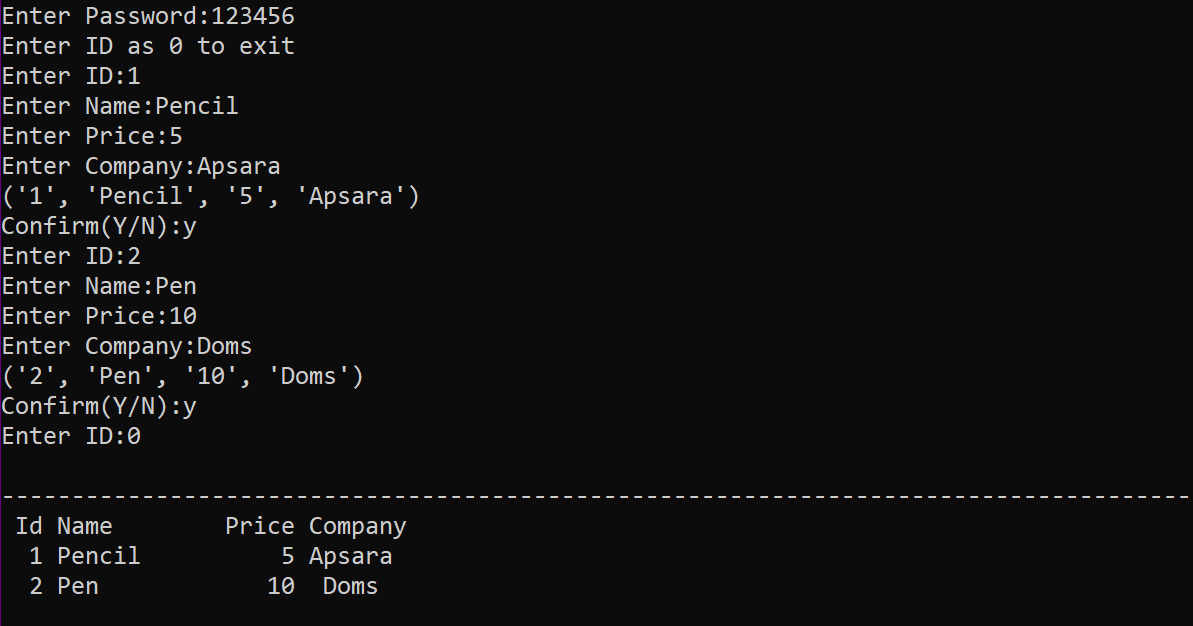
recs=crsr.fetchall()

for rec in recs:

print("%3s %-10s %5s %5s"%(rec[0],rec[1],rec[2],rec[3]))

que()

dis()

**OUTPUT:-** 

**Q.Write a python databases connectivity program that delete records from sample table of database sample** .

**CODE:-**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='sample',charset='utf8')

if con.is\_connected():

print('Successfully connected')

cursor=con.cursor()

print('Displaying records before deleting..............')

cursor.execute('select\*from stationary')

data=cursor.fetchall()

for a in data:

print(a)

nme=input('Enter the name for deleting the record :-')

que='delete from stationary where name="%s"'%(nme)

cursor.execute(que)

print()

cursor.execute('select\*from stationary')

data=cursor.fetchall()

print('Displaying records after deleting.......')

for w in data :

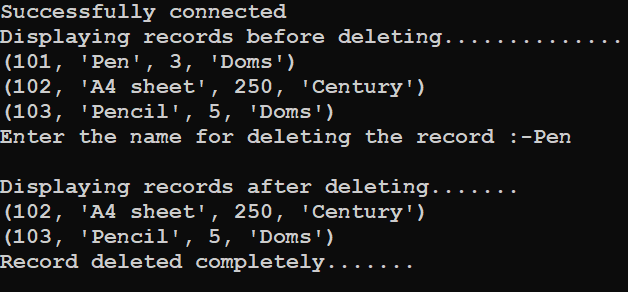
print(w)

con.commit()

con.close()

print('Record deleted completely.......')

**OUTPUT :-**



**Q.** **Define a function named as db to create a new database named as school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def db():

que='create database if not exists school'

cursor.execute(que)

print('Database created...')

db()

**OUTPUT**:-



**Q. Define a function table() to create a new table named as class\_12 with attributes Name, Class, Roll\_no,section and Gender in database named as school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def tb():

cursor.execute('create table if not exists class\_12(Rno int primary key,Name varchar(15),Class int,Sec varchar(10),Gender varchar(15))')

print('Table is created...')

tb()

**OUTPUT:-**



**Q.Define a function insert to add the details of three students in the table class\_12 in database school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def insert():

try:

for w in range(3):

print()

print('Entering the',w+1,'record')

rno=int(input('Enter rno: '))

name = input('Enter name:')

Class=int(input('Enter class:'))

sec=input('Enter section:')

gender=input('Enter Gender: ')

query='Insert into class\_12 values({},"{}",{},"{}","{}")'.format(rno,name,Class,sec,gender)

cursor.execute(query)

con.commit()

con.close()

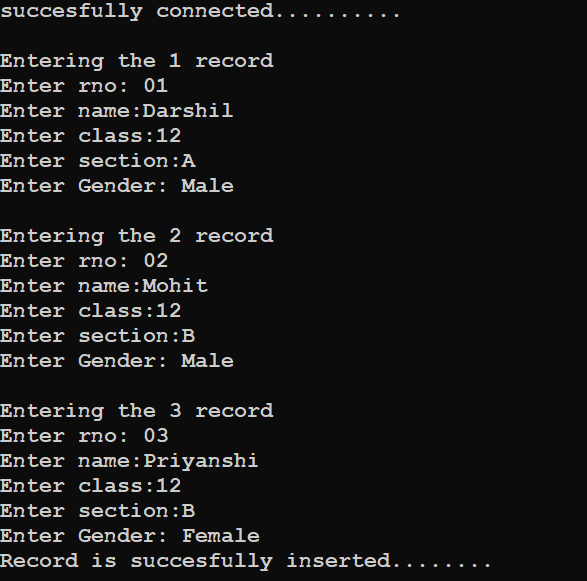
print('Record is succesfully inserted........')

except:

print("Some Error")

insert()

**OUTPUT**



**Q. Define a function named as update() to update the records of the students taken from user of table class\_12 in database school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def update():

rno=int(input('Enter roll number:'))

cursor.execute('desc class\_12')

recs=cursor.fetchall()

print(recs[0][0].ljust(8),recs[1][0].ljust(20),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),sep='')

cursor.execute('select \* from class\_12 where rno={}'.format(rno))

w=cursor.fetchall()

for rec in w:

print(str(rec[0]).ljust(8),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

print('''

1)Name

2)Class

3)Gender

4)Section''')

print()

ch=int(input('Enter Here:'))

if ch==1:

name=input('Enter new name :')

cursor.execute('update class\_12 set name="%s" where rno=%d'%(name,rno))

con.commit()

print('Record updated........')

elif ch==2:

clas=int(input('Enter new class :'))

cursor.execute('update class\_12 set class=%d where rno=%d'%(clas,rno))

con.commit()

print('Record updated........')

elif ch==3:

gender=input('Enter the new Gender :')

cursor.execute('update class\_12 set gender="%s" where rno=%d'%(gender,rno))

con.commit()

print('Record updated.........')

elif ch==4:

sec=input('Enter new Section :')

cursor.execute('update class\_12 set sec="%s" where rno=%d'%(sec,rno))

con.commit()

print('Record updated........')

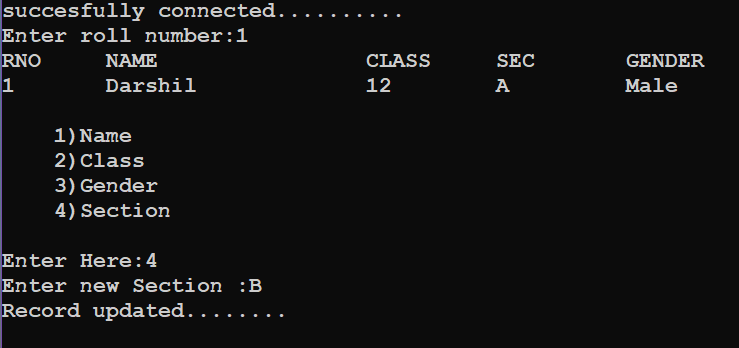
else:

print('WRONG CHOICE !!!!!!!!')

con.close()

update()

**OUTPUT:-**



**Q.Define a function del() to delete the records of the table class\_12 of database school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def delete():

while True:

print('''

1)Delete all

2)Delete with Rno''')

ch=int(input("Enter Choice:"))

if ch==1:

que='Delete from class\_12'

cursor.execute(que)

elif ch==2:

rn=int(input('Enter Roll number: '))

query='delete from class\_12 where rno={}'.format(rn)

cursor.execute(query)

else:

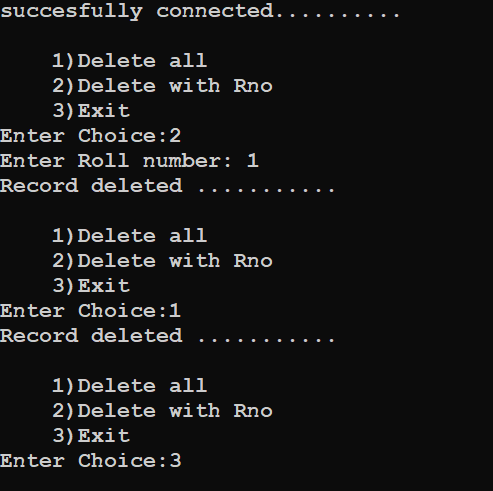
print('WRONG CHOICE.....')

con.commit()

print('Record deleted ...........')

delete()

**OUTPUT**

****

**Q. Define a function display() to display all the records of the table class\_12 of database school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

def display():

try:

cursor.execute('desc class\_12')

recs=cursor.fetchall()

print(recs[0][0].ljust(6),recs[1][0].ljust(15),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),\

sep='')

Que='select \* from class\_12'

cursor.execute(Que)

w=cursor.fetchall()

for rec in w:

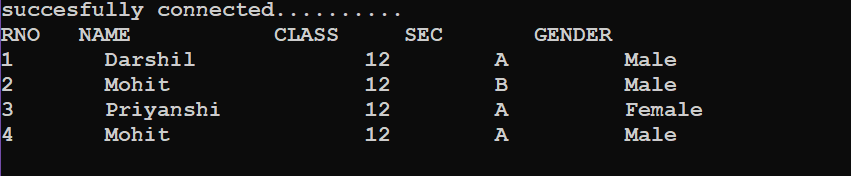
print(str(rec[0]).ljust(8),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

except Exception as error:

print('Error :- ',error)

display()

**OUTPUT**



**Q. Write a python program to display the records of the student of sec A of table class\_12 in database school.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

print()

print('Records of the student of section A :- ')

print()

cursor.execute('desc class\_12')

recs=cursor.fetchall()

print(recs[0][0].ljust(6),recs[1][0].ljust(15),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),sep='')

que='select \* from class\_12 where sec="A" '

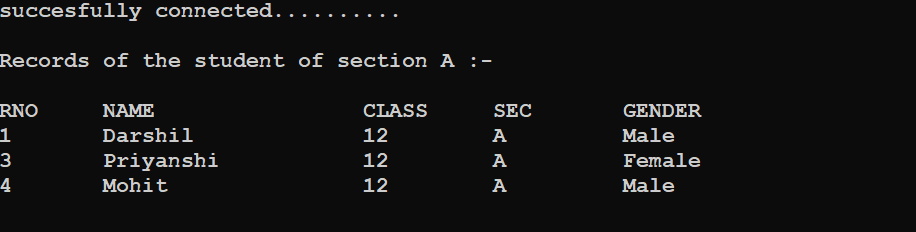
cursor.execute(que)

dat=cursor.fetchall()

for rec in dat:

print(str(rec[0]).ljust(8),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

**OUTPUT**



**Q.Write a python program to delete the records of the students of sec A from table class\_12 in database school and then display the content of the table .**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

cursor=con.cursor()

print()

que='delete from class\_12 where sec="A" '

cursor.execute(que)

con.commit()

cursor.execute('desc class\_12')

recs=cursor.fetchall()

print(recs[0][0].ljust(6),recs[1][0].ljust(20),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),sep='')

que='select \* from class\_12 '

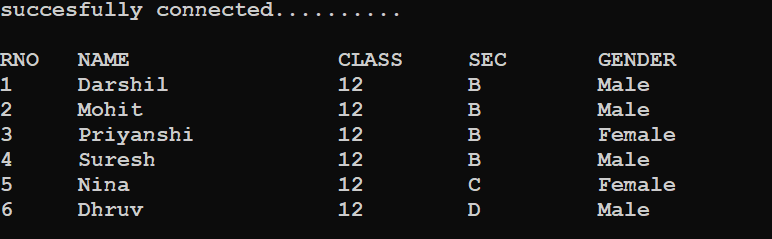
cursor.execute(que)

dat=cursor.fetchall()

for rec in dat:

print(str(rec[0]).ljust(6),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

**OUTPUT**



**Q. Write a python program that display the first three rows fetched from class\_12 table of MYSQL database school**.

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected..........')

crsr=con.cursor()

print()

crsr.execute('desc class\_12')

recs=crsr.fetchall()

print(recs[0][0].ljust(6),recs[1][0].ljust(20),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),sep='')

que='select \* from class\_12 '

crsr.execute(que)

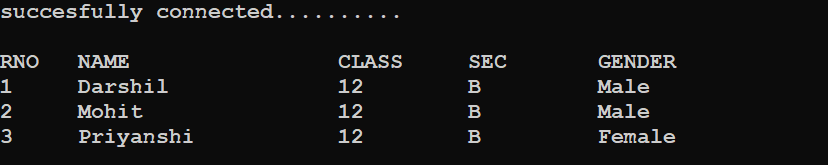
dat=crsr.fetchmany(3)

for rec in dat:

print(str(rec[0]).ljust(6),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

con.close()

**OUTPUT**



**Q. Write a python program that deletes records from class\_12 table of database school that have gender male and then display the contents of the table.**

**CODE**

import mysql.connector as c

con=c.connect(host='localhost',user='root',password='',database='school',charset='utf8')

if con.is\_connected():

print('succesfully connected')

cursor=con.cursor()

que='delete from class\_12 where gender="Male" '

cursor.execute(que)

con.commit()

print('DISPLAYING THE RECORDS……')

print()

cursor.execute('desc class\_12')

recs=cursor.fetchall()

print(recs[0][0].ljust(8),recs[1][0].ljust(20),recs[2][0].ljust(10),recs[3][0].ljust(10),recs[4][0].ljust(10),sep='')

Que='select \* from class\_12'

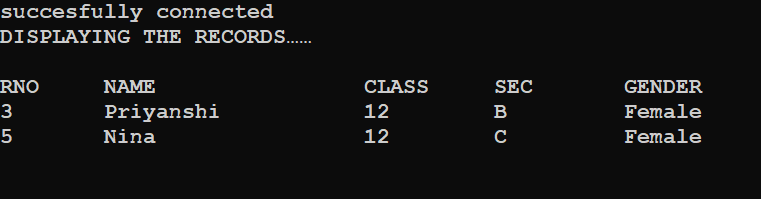
cursor.execute(Que)

w=cursor.fetchall()

for rec in w:

print(str(rec[0]).ljust(8),rec[1].ljust(20),str(rec[2]).ljust(10),rec[3].ljust(10),rec[4].ljust(10),sep='')

**OUTPUT**



**Q. Python program to implement stack operations.**

**CODE:-**

def push(s,x):

global top

s.append(x)

top=len(s)-1

def pop(s):

global top

if len(s)==0:

print("Underflow")

else:

x=s.pop()

print("poped ",x)

if len(s)==0:

top=None

else:

top=len(s)-1

def display(s):

global top

if len(s)==0:

print("stack is empty")

else:

print("Stack elements....")

for a in range (top,-1,-1):

print(s[a])

stack=[]

top=None

while True:

print("\nStack operations")

print("1.Push")

print("2.Pop")

print("3.Display")

print("4.Exit")

print('top is',top)

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

if ch==1:

item=int(input("Enter data : "))

push(stack,item)

elif ch==2:

pop(stack)

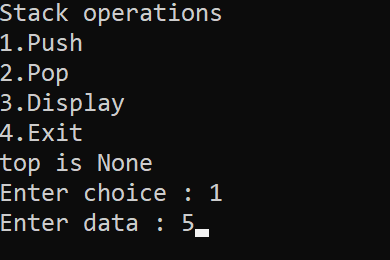
elif ch==3:

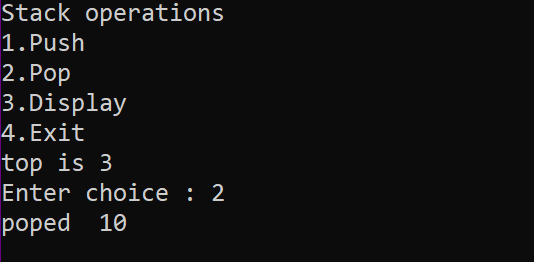
display(stack)

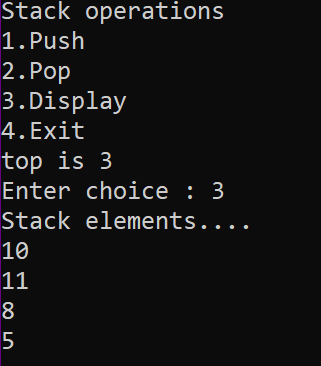
else:

break

**OUTPUT:-**





****

**Q. Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user.Display the content of the Stack along with the largest odd number in the Stack**

**CODE:-**

def push(stack, item):

stack.append(item)

def pop(stack):

if stack==[]:

return

return stack.pop()

def oddStack(num):

if num%2==1:

push(stack,num)

def GetLargest(stack):

elem=pop(stack)

large=elem

while elem!=None:

if large<elem:

large=elem

elem=pop(stack)

return large

n=int(input("how many numbers? "))

stack=[] #empty stack

large= -99

for i in range(n):

number=int(input("Enter number: "))

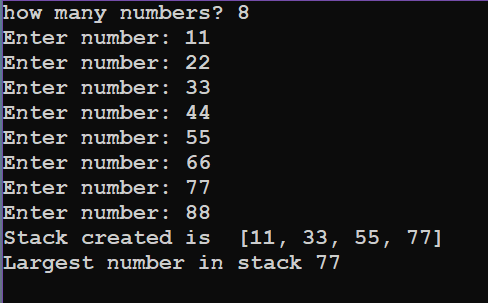
oddStack(number)

print("Stack created is ", stack)

large=GetLargest(stack)

print("Largest number in stack",large)

**OUTPUT:-**



**Q. Write a menu driven program that has functions PushS(lst) and PopS(lst)**

**for performing Push and Pop operations with a stack of List containing integers.**

**CODE:-**

def PushS(lst):

n= int(input("Enter integer:"))

lst.append(n)

def PopS(lst):

if lst==[]:

print("Stack is empty--UNDERFLOW!")

else:

print ("Deleted value :",lst.pop())

lst=[]

while True:

print("""

1)PUSH

2POP

3)EXIT

""")

ch=int(input("Enter Option:"))

if ch==1:

PushS(lst)

elif ch==2:

PopS(lst)

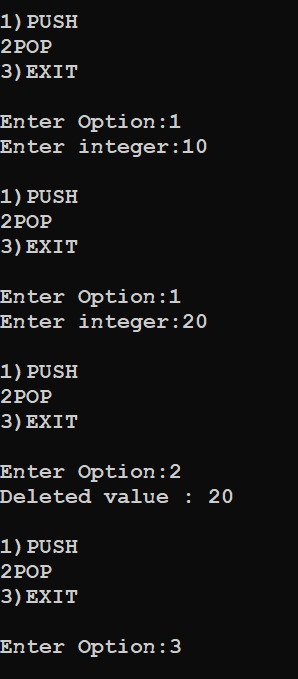
elif ch==3:

break

else:

print("Wrong Option")

**OUTPUT:-**



**Q. Write a menu driven program that has functions Make Push (package) and MakePop(package) to add a new Package and delete & Package from a List of Package Description, considering them to act as push and pop operations of the Stack**

**CODE:-**

def MakePush(package):

a = int(input("Enter package title: "))

package.append(a)

def MakePop(package):

if package==[]:

print("Stack empty")

else:

print ("Deleted element:",package.pop())

package=[]

while True:

print("""

1)PUSH

2)POP

3)EXIT

""")

ch=int(input("Enter Option:"))

if ch==1:

MakePush(package)

elif ch==2:

MakePop(package)

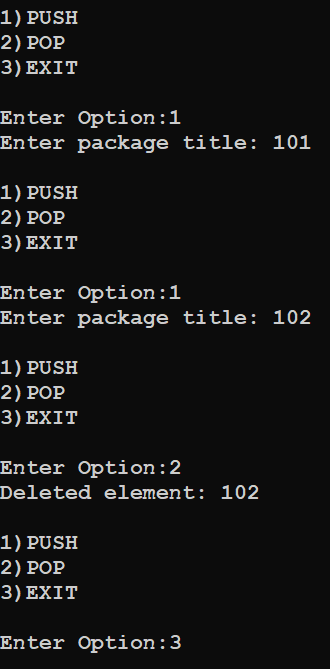
elif ch==3:

break

else:

print("Wrong Option")

**OUTPUT:-**



**Q. Write a program to implement a stack for these book details(Bookno, book name). That is now each item node of the stack contains two types of information-a bookno and its name.Just implement Push and display operations.**

def Push(stk,item):

stk.append(item)

top=len(stk)-1

def Display(stk):

if stk==[]:

print("Stack empty")

else:

top=len(stk)-1

print(stk[top],"<-top")

for a in range(-2,-top-2, -1):

print(stk[a])

stack=[]

top=None

while True:

print("STACK OPERATIONS")

print("1. Push")

print("2. Display stack")

print("3. Exit")

ch=int(input("Enter your choice (1-5) :"))

if ch==1:

bno=int(input("Enter Book no. to be inserted :"))

bname=input("Enter Book name to be inserted :")

item=[bno, bname]

Push(stack,item)

elif ch==2:

Display(stack)

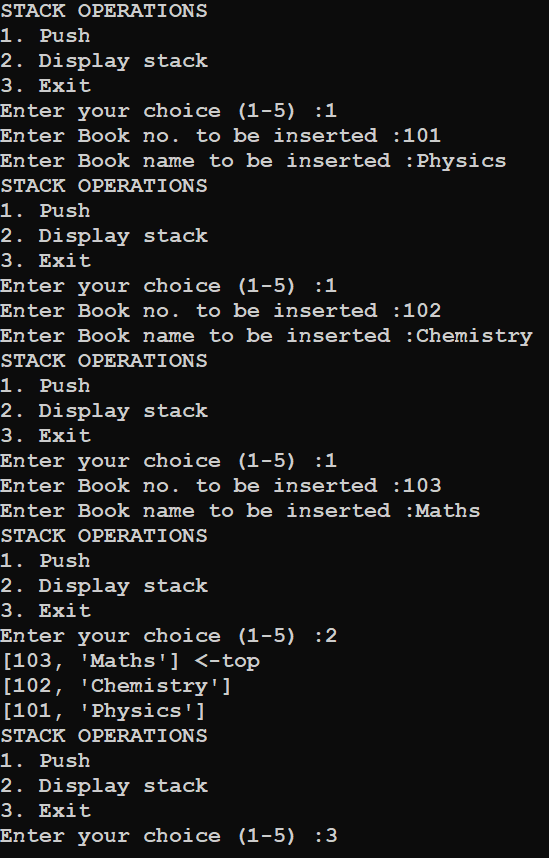
elif ch==3:

break

else:

print("Invalid choice!")

**OUTPUT:-**



**Q. Write a program to perform insert and delete operations on a Queue containing Members details as given in the following definition of itemnode:**

**MemberNo : integer**

**MemberName : String**

**Age : integer**

**CODE:-**

def isEmpty(Qu):

if Qu==[]:

return True

else :

return False

def Enqueue(Qu,item):

Qu.append(item)

if len(Qu)==1:

front=rear = 0

else:

rear=len(Qu) - 1

def Dequeue(Qu):

if isEmpty(Qu) :

return "Underflow"

else:

item= Qu.pop(0)

if len(Qu) == 8:

front=None

rear=None

return item

def Display(Qu):

if isEmpty(Qu):

print("Queue Empty!")

elif len(Qu)==1:

print(Qu[0], "<== front, rear")

else:

front=0

rear=len(Qu)-1

print(Qu[front],"<--front")

for a in range(1, rear):

print (Qu[a])

print (Qu[rear], "<-rear")

queue=[]

front=None

while True:

print("QUEUE OPERATIONS")

print("1. Enqueue")

print("2. Dequeue")

print("3. Display queue")

print("4. Exit")

ch= int(input("Enter your choice (1-5): "))

if ch==1:

print("For the new member, enter details below:")

memberNo= int(input("Enter member no :"))

memberName=input("Enter member name :")

age = int(input("Enter member's age :"))

item=[memberNo,memberName,age]

Enqueue(queue,item)

input("Press Enter to continue...")

elif ch==2:

item=Dequeue(queue)

if item=="Underflow":

print("Underflow! Queue is empty!")

else:

print("Dequeue-ed item is", item)

input("Press Enter to continue...")

elif ch==3:

Display (queue)

input("Press Enter to continue...")

elif ch == 4:

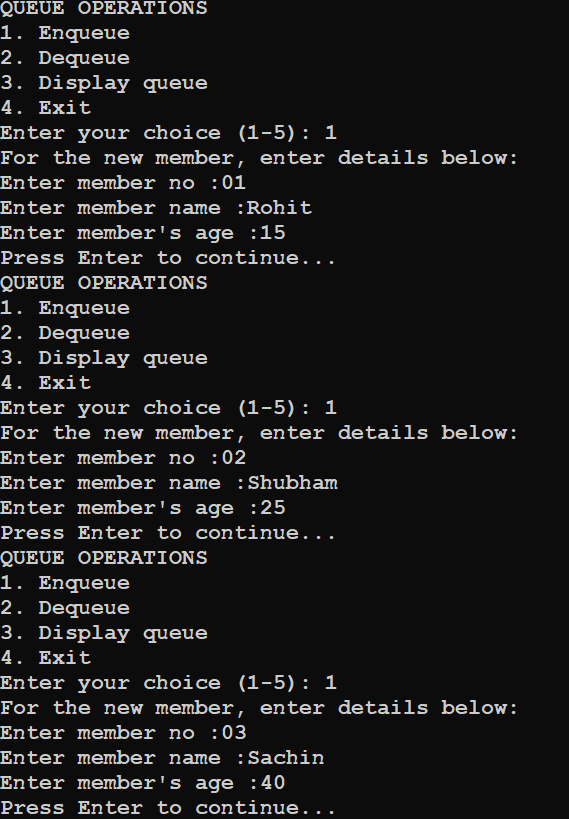
break

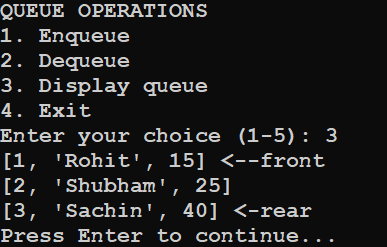
else :

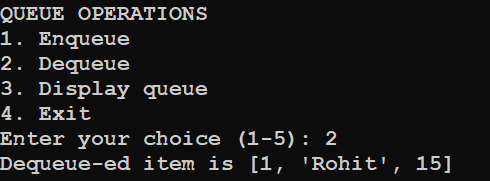
print("Invalid choice!")

input("Press Enter to continue...")

**OUTPUT:-**







**Q.Write a function in Python,INSERTQ(arr,data) and DELETEQ(Arr) for performing insertion and deletion operations in a Queue. arr is the list used for implementing queue and data is the value to be inserted.**

**CODE:-**

def INSERTQ(arr):

data =int(input("Enter data to be inserted: "))

arr.append(data)

def DELETEQ(arr):

if arr== []:

print("Queue empty")

else:

print ("Deleted element is:", arr[0])

arr.pop(0)

arr=[]

while True:

print("""

1)INSERTQ

2)DELETEQ

3)EXIT

""")

ch=int(input("Enter Option:"))

if ch==1:

INSERTQ(arr)

elif ch==2:

DELETEQ(arr)

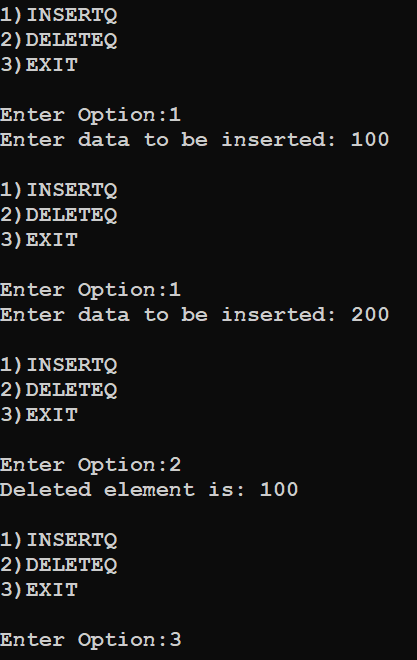
elif ch==3:

break

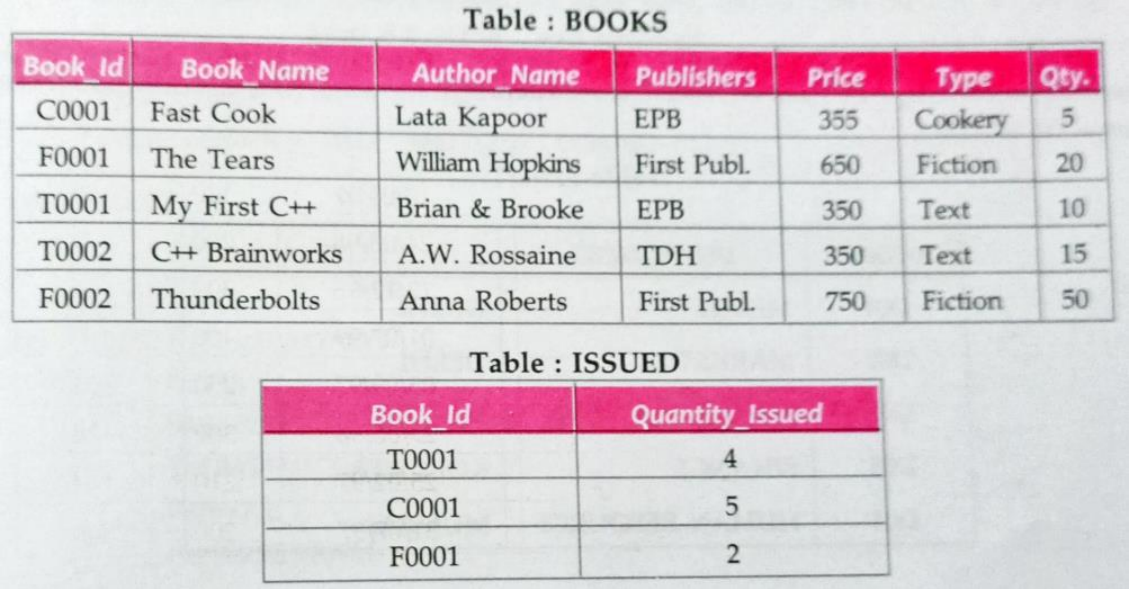
else:

print("Wrong Option")

**OUTPUT:-**

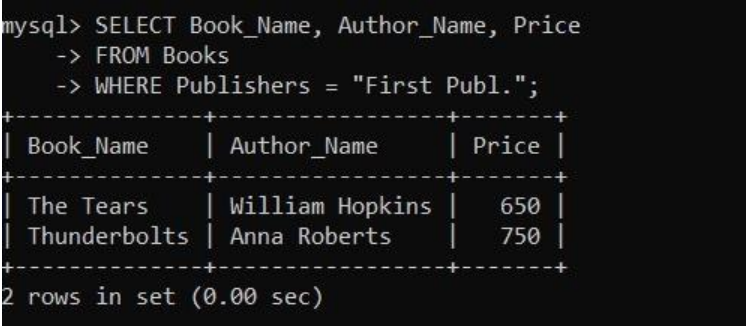


**Q.Given the following tables for a database LIBRARY** :

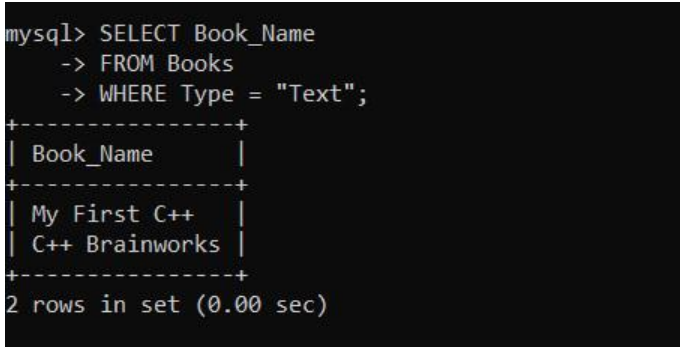


**Write SQL queries for (a) to (f):**

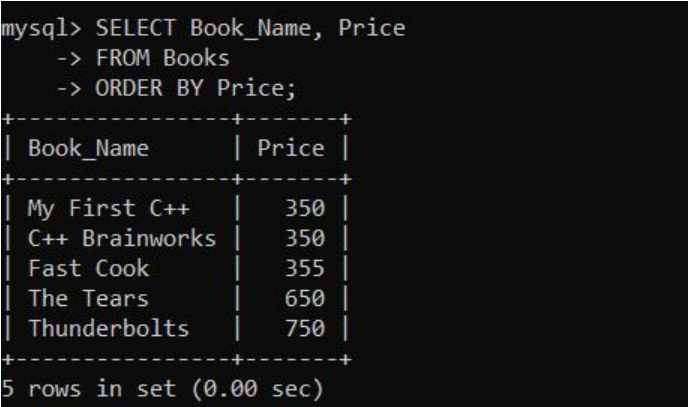
1. **To show Book name , Author name and Price of books of First Publc publishers.**



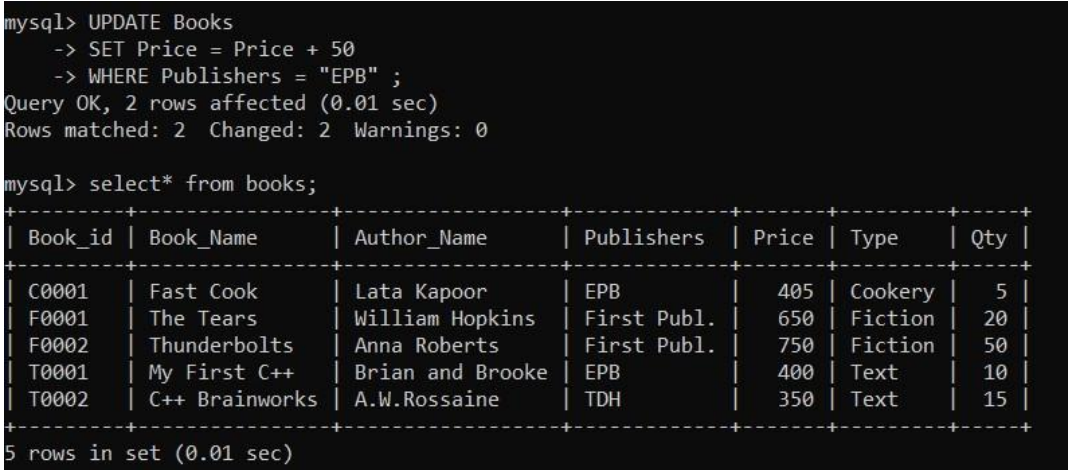
1. **To list the names from books of text type.**



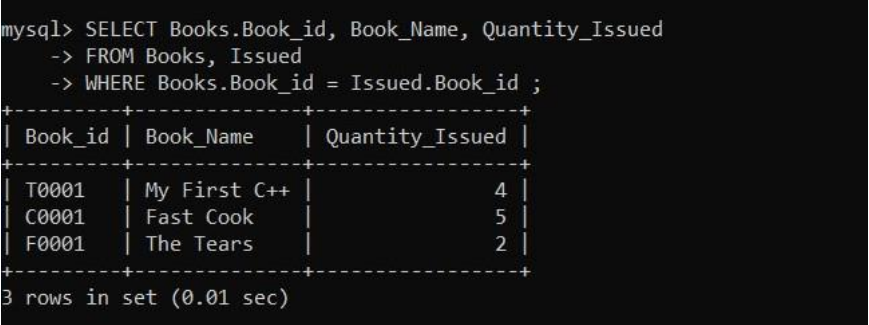
1. **To display the names and prices from books in ascending order of their price.**



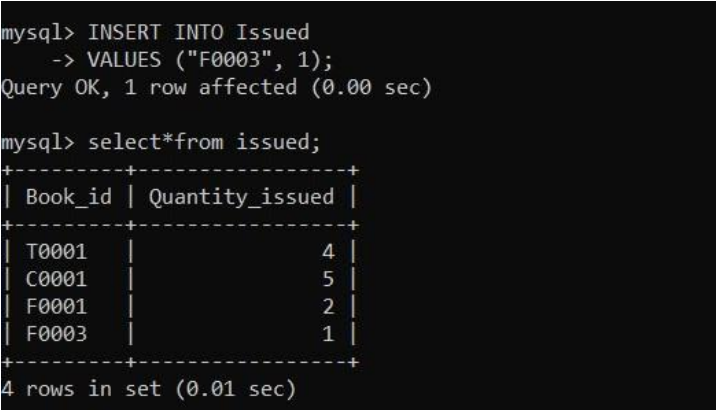
1. **To increase the price of all books of EPB Publishers by 50 .**



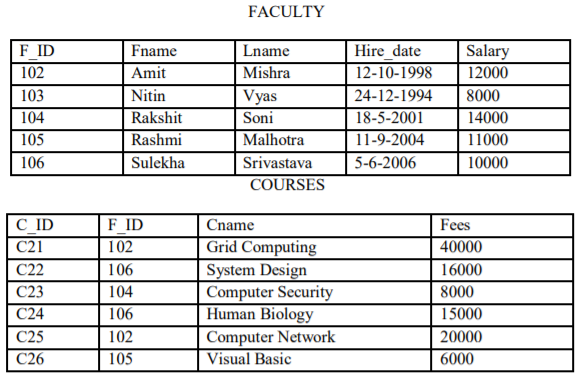
1. **To display the Book\_id,Book\_Name and Quantity\_Issued for all books which have been issued.(The query will require contents from both the tables.)**



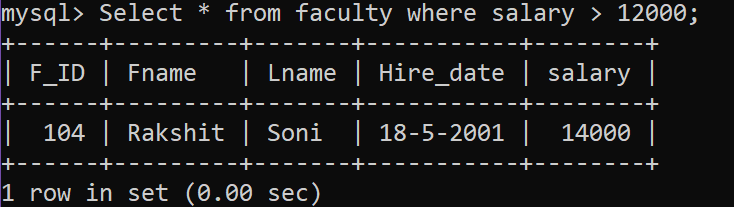
1. **To insert a new row in the table Issued having the following data : "F0003",1**



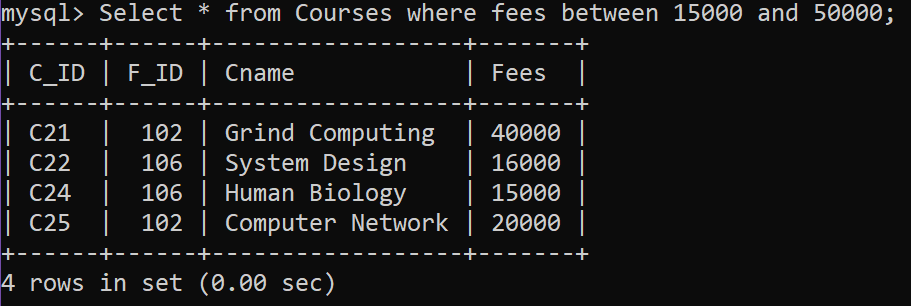
**Q. Consider the following tables FACULTY and COURSES. Write SQL commands for the statements (i) to (vii).**

****

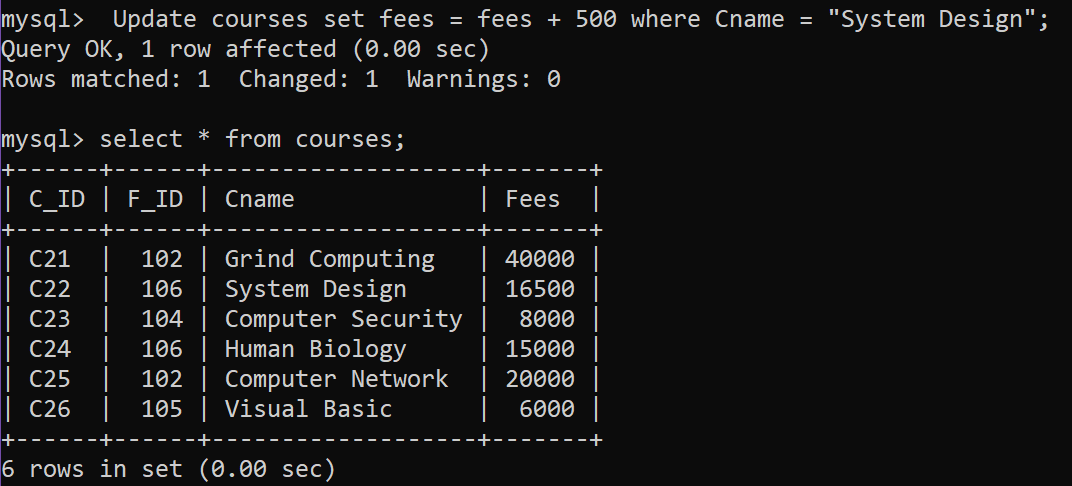
**i) To display details of those Faculties whose salary is greater than 12000.**

****

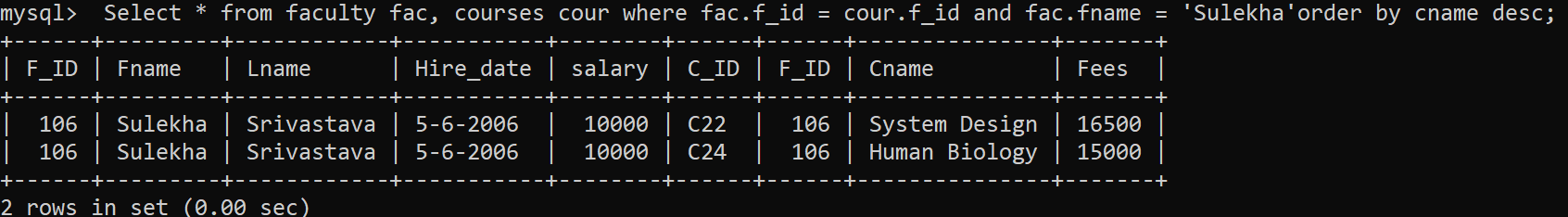
**ii) To display the details of courses whose fees is in the range of 15000 to 50000 (both values included).**

****

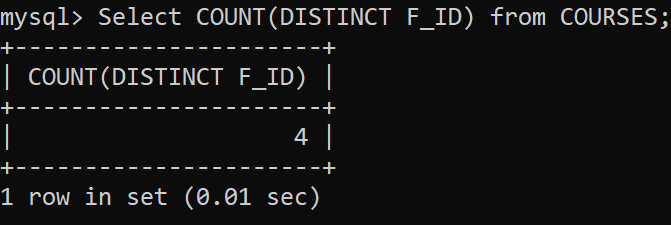
**iii) To increase the fees of all courses by 500 of “System Design” Course.**

****

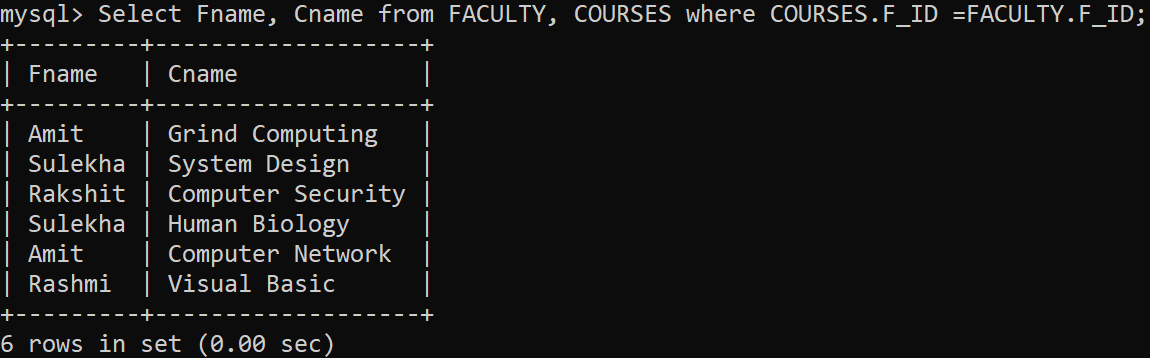
**iv) To display details of those courses which are taught by ‘Sulekha’ in descending order of courses.**

****

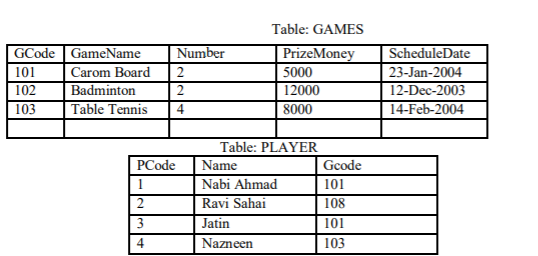
**v) To count all record with F\_ID**

****

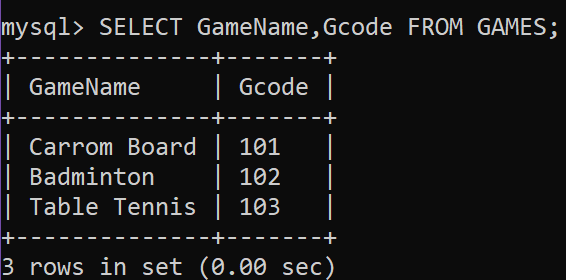
**vi) To display Fname, Cname from FACULTY, COURSES where F\_ID is same.**

****

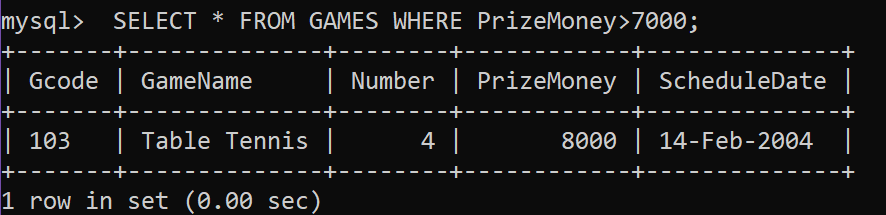
**Q. Consider the following tables GAMES and PLAYER. Write SQL commands for the statements (i) to (viii) .**

****

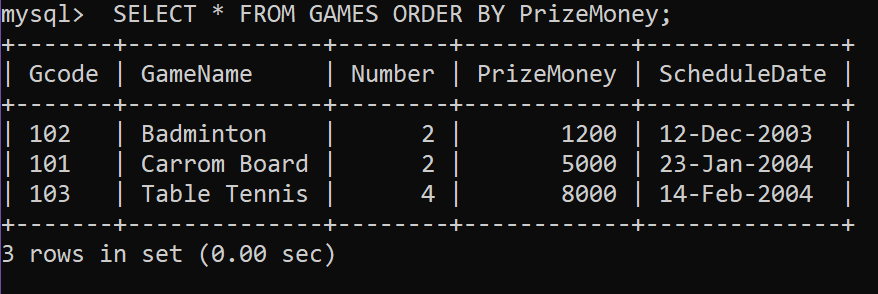
**(i) To display the name of all Games with their Gcodes.**

****

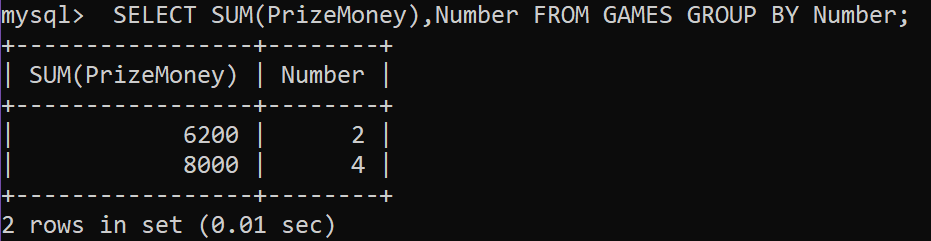
**(ii) To display details of those games which are having PrizeMoney more than 7000.**

****

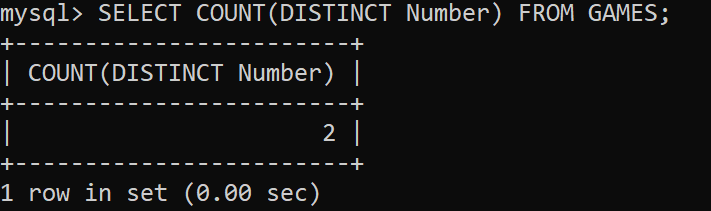
**(iii) To display the content of the GAMES table in ascending order of Prize.**

****

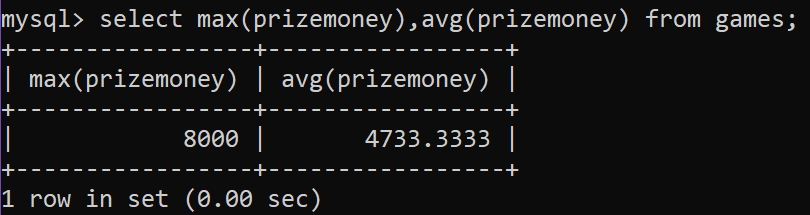
**(iv) To display sum of PrizeMoney for each of the Number of participation groupings (as shown in column Number 2 or 4).**

****

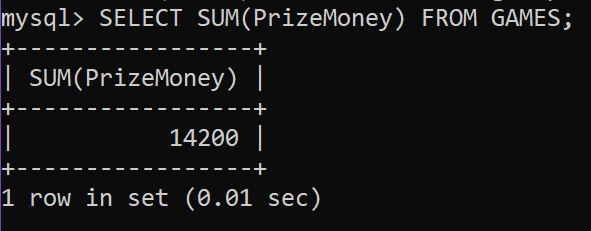
**(v) To display the Count of different games from games;**

****

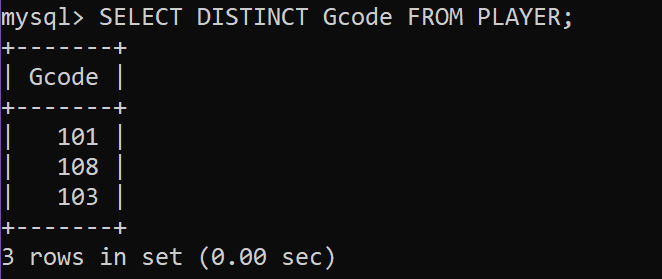
**(vi) To display the Maximum PrizeMoney and Average of PrizeMoney from Games**

****

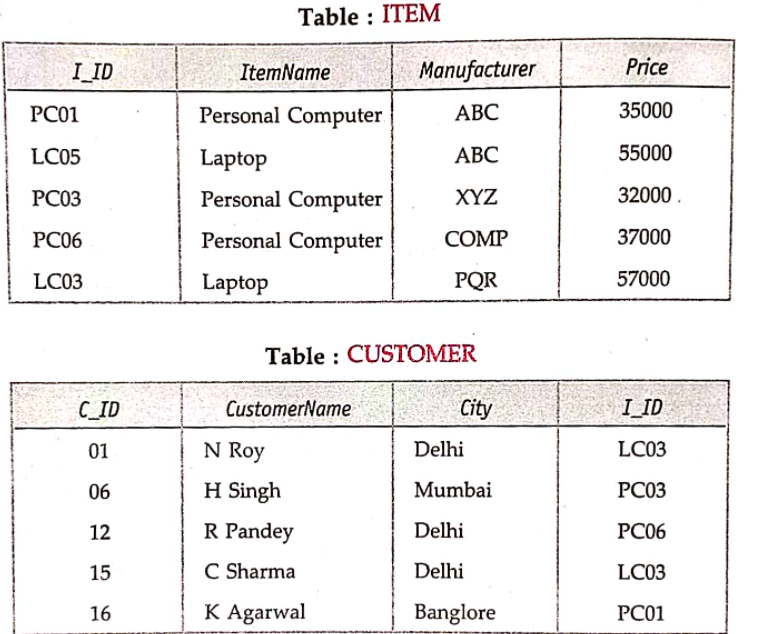
**(vii) SELECT SUM(PrizeMoney) FROM GAMES;**

****

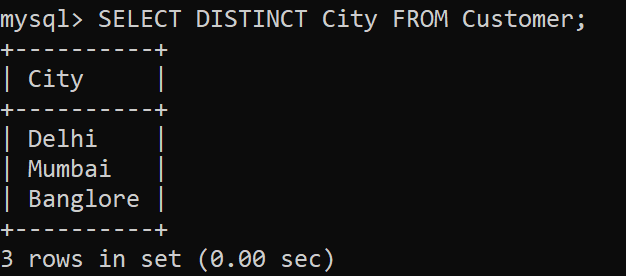
**(viii) To display different GCode From Player**

****

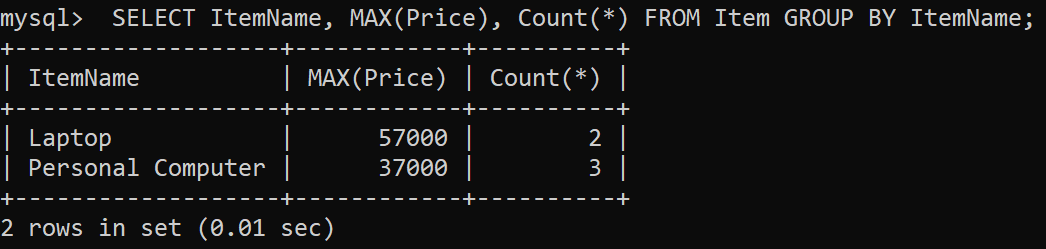
**Q. Consider the following tables ITEM and CUSTOMER. Write SQL commands for the statements (i) to (v).**

****

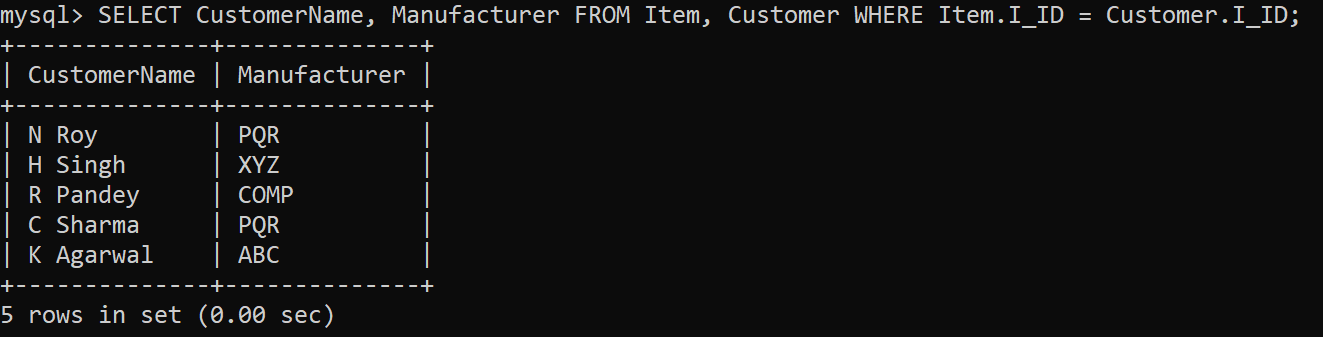
**(i)To display distinct cities of customers**

****

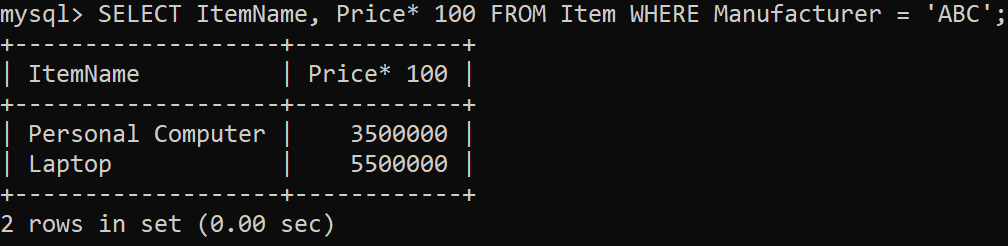
**(ii)To display Name of the items,maximum price and no. of records with same Item Name in table Item**

****

**(iii)** **To display customer name brought the item and their manufacturers**

****

**(iv)** **To display item name and price\*100 by ABC manufacturer from Item Table**

****

**(v)** **To display average price of items from item table**

