**Program 1(a)**

**Aim: Write a Python program to find factorial of a given number using recursion.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 19 02:25:48 2024

@author: it297

"""

def fact(n):

if(n<0):

return 1

elif n==0:

return 1

elif n==1:

return 1

else:

return n\*fact(n-1)

while True:

n=int(input("enter n value"))

if n<0:

print("please enter positive number")

else:

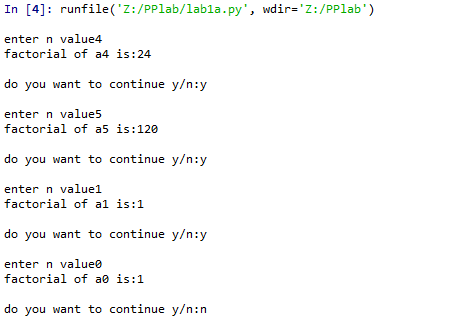
print("factorial of a{} is:{}".format(n,fact(n)))

c=input("do you want to continue y/n:")

if (c=='N' or c=='n'):

break

**Output:-**

****

**Program 1(b)**

**Aim: Write a Python program to print fibbanocci series upto n terms using recursion**.

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 19 02:46:52 2024

@author: it297

"""

def fib(n):

if(n<=1):

return n

else:

return(fib(n-1)+fib(n-2))

while True:

n=int(input("enter n value:"))

if(n<=0):

print("please enter a postive integer")

else:

print("fibonacci series is:")

for i in range (n):

while(fib(i)<=n):

print(fib(i))

break

c=input("do you want to continue Y/N:")

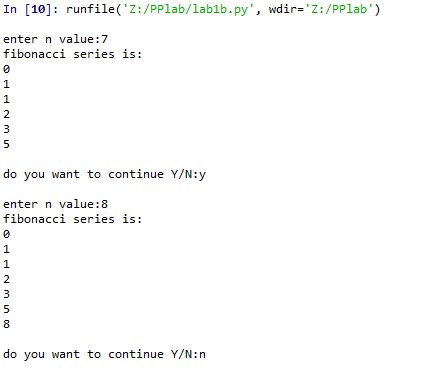
if (c=='Y' or c=='y'):

continue

else:

break

**Output:-**

****

**Program 1(c)**

**Aim: Write a Python program to print sum of digits of given number using recursion.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 19 03:10:54 2024

@author: it297

"""

while True:

def digit(n):

if n<=9:

return n

else:

rem=n%10

return rem+digit(n//10)

n=int(input("enter n value"))

print("sum of digits of given {} is {}".format(n,digit(n)))

ch=input("Do you want to continue Y|N")

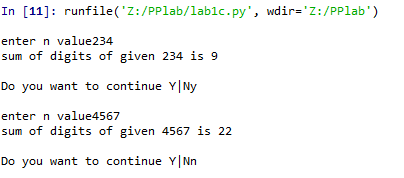
if ch=='y' or ch=='Y':

continue

else:

break

**Output:-**

****

**Program 2(a)**

**Aim: Write a Python program to print the dates 30 days before and after from the given date.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Mar 21 15:01:30 2024

@author: it297

"""

from datetime import datetime,timedelta

while True:

s=input("enter any date(dd:mm:yyyy): ")

d=datetime.strptime(s,"%d:%m:%Y")

d1=d.date()

print(" The given date is: ",d)

print("After 30 days:")

d2=d1

for i in range(30):

d1+=timedelta(days=1)

print(d1)

print("Before 30 days:")

for i in range(30):

d2-=timedelta(days=1)

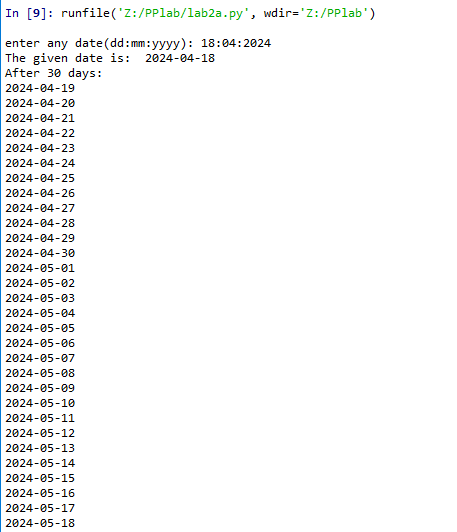
print(d2)

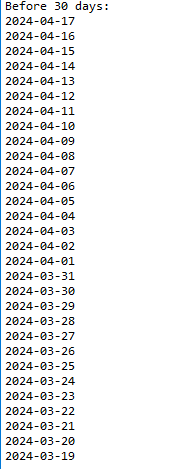
c=input("do you want to continue y/n: ")

if c=='Y' or c=='y':

break

**Output:-**





**Program 2(b)**

**Aim: Write a Python program to print 12 fixed dates from a specified date where the difference between two days will be 20.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Mar 21 14:56:13 2024

@author: it297

"""

from datetime import datetime,timedelta

s=input("enter any date(dd:mm:yyyy): ")

d=datetime.strptime(s,"%d:%m:%Y")

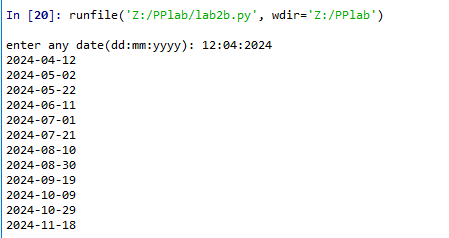
for i in range(12):

r=d+timedelta(days=20\*i)

result=r.strftime("%Y-%m-%d")

print(result)

**Output:-**

****

**Program 2(c)**

**Aim: Write a Python program to calculate your age.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Mar 22 07:35:50 2024

@author: it297

"""

from datetime import datetime

while True:

s=input("enter your date of birth(yyyy:mm:dd): ")

dob=datetime.strptime(s,"%Y:%m:%d")

t=datetime.now()

age=t-dob

years=age.days//365

months=(age.days%365)//30

days=(age.days%365)%30

print("your age is:{} years {} months {} days".format(years,months,days))

ch=input("do you want to continue (y/n): ")

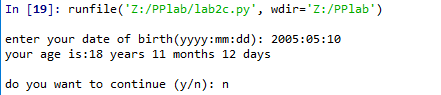
if ch=='y' or ch=='Y':

continue

else:

break

**Output:-**

****

**Program 3(a)**

**3. Python program to read a text file named “bec.txt” and display the following.**

**a) Aim: The word start with t and a or T and A.**

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 19 02:30:40 2024

@author: it297

"""

while True:

f=open("bec.txt","w")

n=input("enter any data separated by space:")

f.write(n)

f.close()

a=open("bec.txt","r")

s=a.read()

n=n.split(" ")

for i in n:

if i[0]=='t' or i[0]=='a' or i[0]=='A' or i[0]=='T':

print(i)

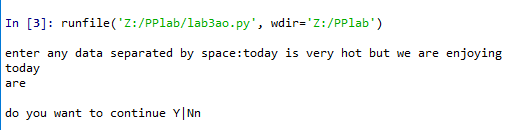
a.close()

ch=input("do you want to continue Y|N")

if ch=='N' or ch=='n':

break

**Output:-**



**Program 3(b)**

**Aim: Write a Python program to find the even length of words in a given file.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 19 02:31:29 2024

@author: it297

"""

while True:

f=open("bec.txt","w")

n=input("enter any data separated by space:")

f.write(n)

f.close()

a=open("bec.txt","r")

s=a.read()

n=s.split(" ")

for i in n:

if len(i)%2==0:

print(i)

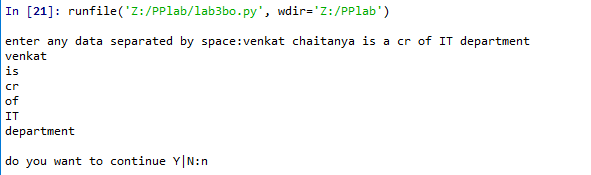
a.close()

ch=input("do you want to continue Y|N:")

if ch=='N' or ch**==**'n':

break

**Output:-**

****

**Program 3(c)**

**Aim: Write a Python program to read a binary file named ‘book.dat” has structure [book name, book no, author, price].**

**1) Write a user defined function createFile() to input data for a record and add to “book.dat”.**

**2) Write a user defined function countRec (author) which accepts author name as an argument and display a no.of books of the given author.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Mar 15 03:09:30 2024

@author: it297

"""

import pickle

while True:

def createFile():

file=open("book.dat","ab")

BookNo=int(input("enter book number:"))

Book\_Name=input("enter book name:")

Author=input("enter author:")

price=int(input("enter price:"))

record=[BookNo,Book\_Name,Author,price]

pickle.dump(record,file)

file.close()

def countRec(Author):

file=open("book.dat","rb")

cnt=0

try:

while True:

record=pickle.load(file)

if record[2]==Author:

cnt+=1

except EOFError:

pass

return cnt

file.close()

def testprogram():

while True:

createFile()

ch=input("add more records(Y|N)?")

if ch=='N' or ch=='n':

break

Author=input("enter author name to search:")

n=countRec(Author)

print("No.of books are",n)

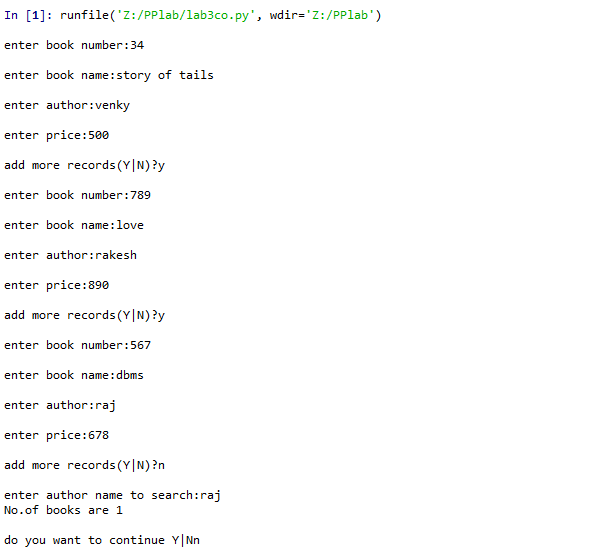
testprogram()

c=input("do you want to continue Y|N")

if c=='n' or c=='N':

break

**Output:**

**-**

**Program 4(a)**

**Aim: Write a Python program to implement the following data structures using Oops concepts.**

**a) Stack using list:**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 20 03:55:41 2024

@author: it297

"""

class Stack:

l=[]

def \_\_intit\_\_(self):

self.ele=0

def push(self):

self.ele=int(input("enter the element you want to push: "))

l.append(self.ele)

print("element is inserted successfully")

def pop(self):

if len(l)==0:

print("stack is empty")

else:

print("popped element is: ",l[-1])

l.pop()

def display(self):

if len(l)==0:

print("stack is empty")

else:

print("stack elements are: ")

i=len(l)

while i>0:

print(l[i-1])

i-=1

l=[]

ob=Stack()

while True:

print("1.push\n2.pop\n3.display\n4.exit\n")

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

if ch==1:

ob.push()

elif ch==2:

ob.pop()

elif ch==3:

ob.display()

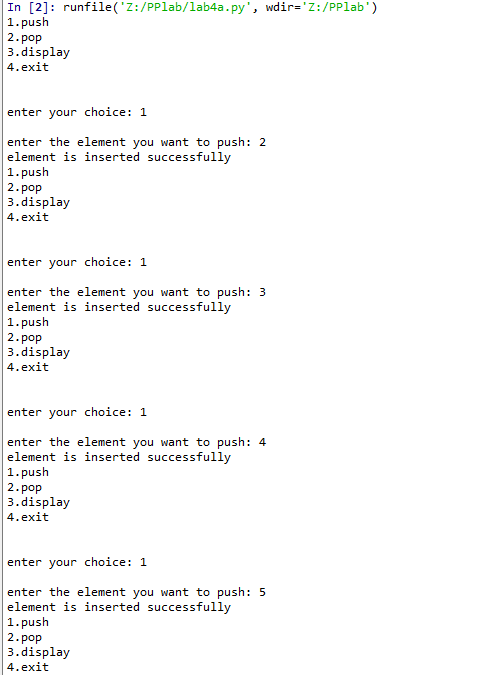
elif ch==4:

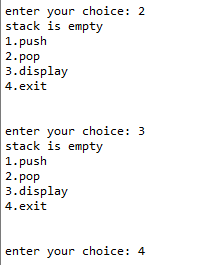
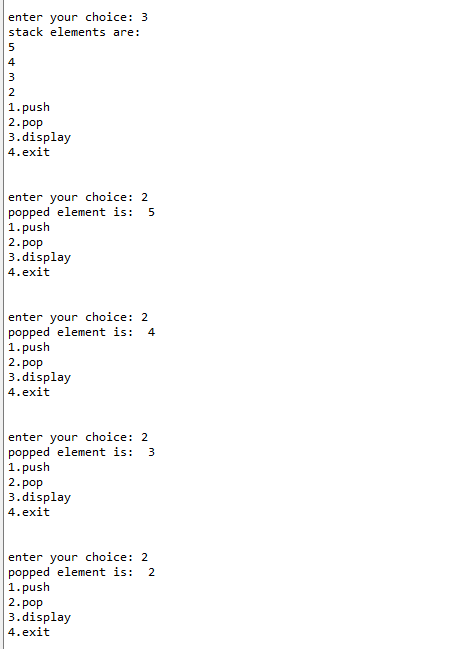
break

else:

print("invalid choice")

**Output:-**

****

****

**Program-4(b)**

**Aim: Write a Python program to implement Queue using list.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Mar 21 02:55:41 2024

@author: it297

"""

class Queue:

l=[]

def \_\_intit\_\_(self):

self.ele=0

def enqueue(self):

self.ele=int(input("enter the element you want to enqueue: "))

l.append(self.ele)

print("element is enqueued successfully")

def dequeue(self):

if len(l)==0:

print("queue is empty")

else:

print("dequeued element is: ",l[0])

l.pop(0)

def display(self):

if len(l)==0:

print("queue is empty")

else:

print("queue elements are: ")

for i in range(len(l)):

print(l[i],end=' ')

l=[]

ob=Queue()

while True:

print("\n1.enqueue\n2.dequeue\n3.display\n4.exit\n")

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

if ch==1:

ob.enqueue()

elif ch==2:

ob.dequeue()

elif ch==3:

ob.display()

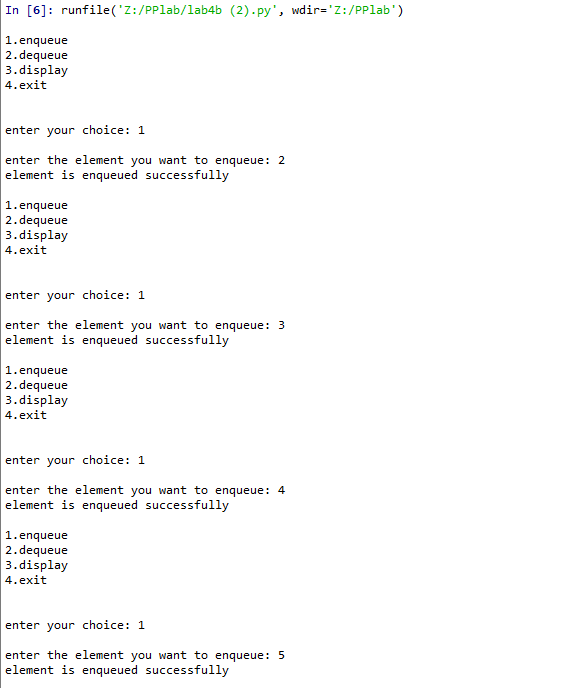
elif ch==4:

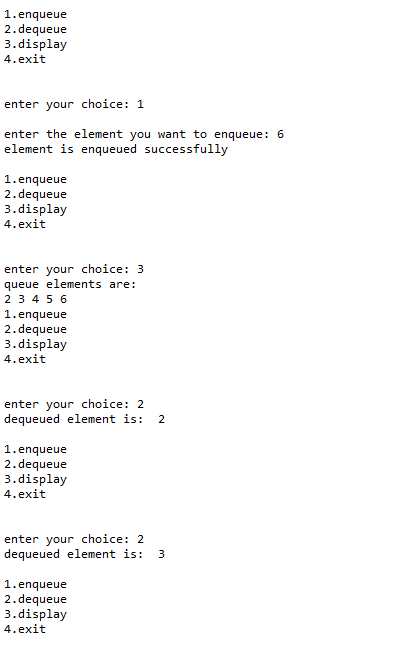
break

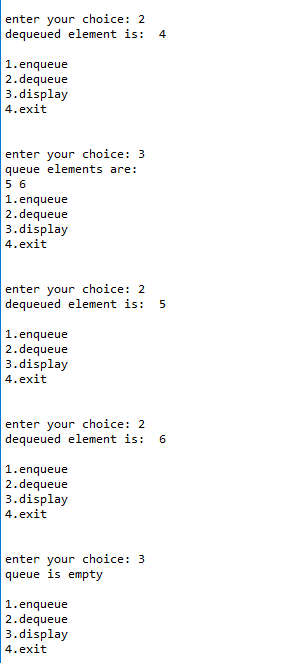
else:

print("invalid choice")

**Output:-**

****

****

****

****

**Program 4(c)**

**Aim: Write a Python program to implement Single Linked List.**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Apr 14 04:19:29 2024

@author: it297

"""

class Node:

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

self.data=data

self.next=None

class Linked\_list:

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

self.head=None

def insert(self,data):

if self.head:

temp=self.head

while(temp.next):

temp=temp.next

temp.next=data

else:

self.head=data

def traverse(self):

if self.head==None:

print("list is empty")

else:

temp=self.head

while(temp!=None):

print(temp.data,end="->")

temp=temp.next

def insert\_at\_begin(self,data):

if self.head==None:

self.head=data

else:

temp=self.head

self.head=data

self.head.next=temp

print("inserted successfully")

def delete\_at\_begin(self):

if self.head==0:

print("list is empty you cant delete")

else:

temp=self.head

print("deleted node is {}".format(temp.data))

self.head=temp.next

print("deleted successfully")

def insert\_at\_specific\_location(self,d,p):

if p==0:

self.insert\_at\_begin(d)

return

new\_node=Node(d)

current\_node=self.head

count=0

while current\_node:

if count==p-1:

new\_node.next=current\_node.next

current\_node.next=new\_node

break

current\_node=current\_node.next

count+=1

else:

print("invalid location")

l1=Linked\_list()

while True:

print("linked list operations")

print("1.create\n2.traverse\n3.insert at begin\n4.delete at begin\n5.insert at specific location")

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

if ch==1:

n=int(input("enter node data:"))

n1=Node(n)

l1.insert(n1)

elif ch==2:

l1.traverse()

elif ch==3:

ne=int(input("enter node data:"))

n1=Node(ne)

l1.insert\_at\_begin(n1)

elif ch==4:

l1.delete\_at\_begin()

elif ch==5:

p=int(input("enter the position:"))

if p<0:

print("invalid position")

elif p==0:

n=int(input("enter node data:"))

l1.insert\_at\_begin(n)

else:

n=int(input("enter node data:"))

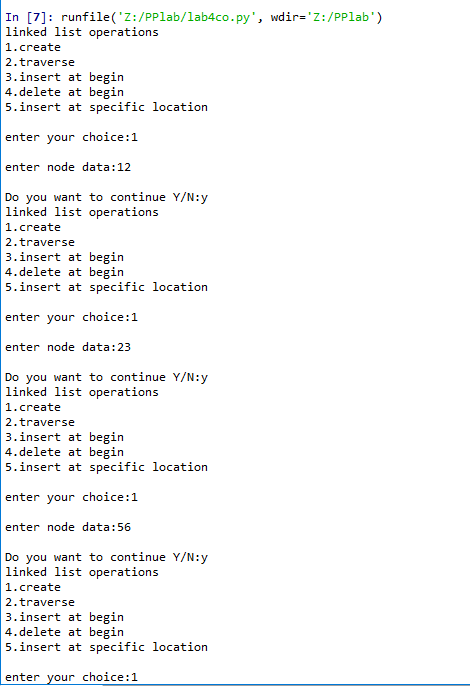
l1.insert\_at\_specific\_location(n,p)

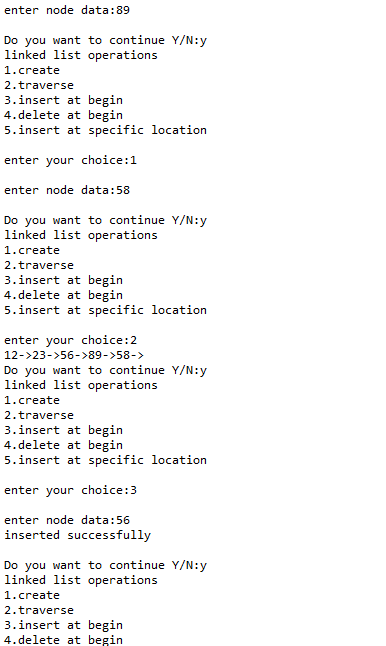
c=input("Do you want to continue Y/N:")

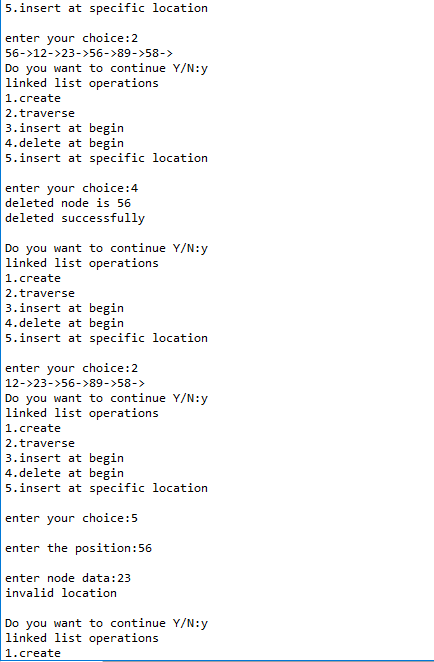
if c=='n' or c=='N':

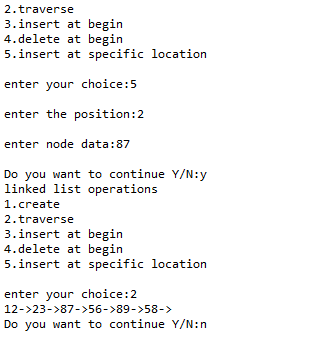
break

**Output:-**

****

****

****

****

**Program 5**

**Aim: Write a Python program to implement Bank Management System for the following:**

**a)Create b)Deposit c) Withdrawl d)Check Balance e)Mini Statement**

**Source Code:-**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Apr 12 04:28:35 2024

@author: it297

"""

import re

import datetime

import random

class Bank:

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

self.acc=0

self.name=" "

self.bal=0

self.pwd=" "

def create(self):

self.acc=random.randint(100000,100099)

self.name=input("Enter account holder name:")

self.bal=int(input("Enter opening balance:"))

if self.bal<500:

print("Opening balance should be grater then 500")

return

self.pwd=input("create a strong password containing atleast one capital,onesmall and a special character of lenght minimum 5 :")

if len(self.pwd)<5 or not re.search('[a-z]',self.pwd) or not re.search('[A-Z]',self.pwd) or not re.search('\W',self.pwd) or not re.search('\d',self.pwd):

print("enter proper password")

return

self.t=datetime.datetime.now()

print("Account is created succesfully at {}".format(self.t))

print("Your account number is{}".format(self.acc))

print("Your password is {}".format(self.pwd))

f=open("transs.txt","a")

f.write("Acc:")

f.write(str(self.acc))

f.write(" ")

f.write("Time:")

f.write(str(self.t))

f.write(" ")

f.write("Bal:")

f.write(str(self.bal))

f.write(" ")

f.write("pwd:")

f.write(str(self.pwd))

#print("\n")

f.write("\n")

f.close()

def deposit(self):

ch=int(input("Enter the account number you want to deposit:"))

p=input("Enter the password:")

for account in accounts:

if account.acc==ch and account.pwd==p:

self.t=datetime.datetime.now()

amount=int(input("Enter the amount you want to deposit:"))

account.bal+=amount

print("deposited succesfully")

f=open("transs.txt","a")

f.write("Acc:")

f.write(str(ch))

f.write(" ")

f.write("Time:")

f.write(str(self.t))

f.write(" ")

f.write("Dep:")

f.write(str(amount))

f.write(" ")

f.write("Bal:")

f.write(str(account.bal))

f.write(" ")

f.write("Pwd:")

f.write(p)

f.write("\n")

#print("\n")

f.close()

else:

print("Check password and Accno")

continue

def withdraw(self):

c=int(input("Enter the account number you want to wthdraw:"))

p=input("Enter the password:")

for account in accounts:

if account.acc!=c and account.pwd!=p:

print("Check password and Accno")

return

else:

self.t=datetime.datetime.now()

n=int(input("Enter the amount you want to withdraw:"))

if account.bal>=n:

account.bal-=n

print("withdraw succesful")

f=open("transs.txt","a")

f.write("Acc:")

f.write(str(c))

f.write(" ")

f.write("Time:")

f.write(str(self.t))

f.write(" ")

f.write("Wd:")

f.write(str(n))

f.write(" ")

f.write("Bal:")

f.write(str(account.bal))

f.write(" ")

f.write("Pwd:")

f.write(p)

f.write("\n")

f.close()

else:

print("Insufficient balance")

def check(self):

ch=int(input("Enter the account number you want to check the balance:"))

p=input("Enter the password:")

for account in accounts:

if account.acc==ch and account.pwd==p:

print("Your account has {} balance left".format(account.bal))

return

else:

print("Check the account number and password")

def mini(self):

f=open("transs.txt","r")

l=f.readlines()

f.close()

for i in l:

print(i.strip())

#print(l)

accounts=[]

while True:

ob=Bank()

print("\n1.create\n2.deposit\n3.withdraw\n4.check balance\n5.mini statement")

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

if ch==1:

ob.create()

accounts.append(ob)

elif ch==2:

ob.deposit()

elif ch==3:

ob.withdraw()

elif ch==4:

ob.check()

elif ch==5:

ob.mini()

elif ch==6:

break

else:

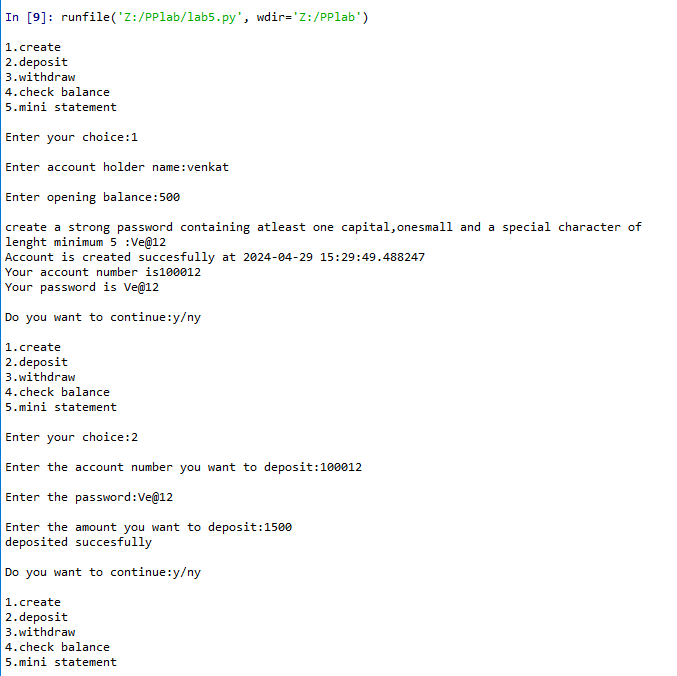
print("Invalid statement")

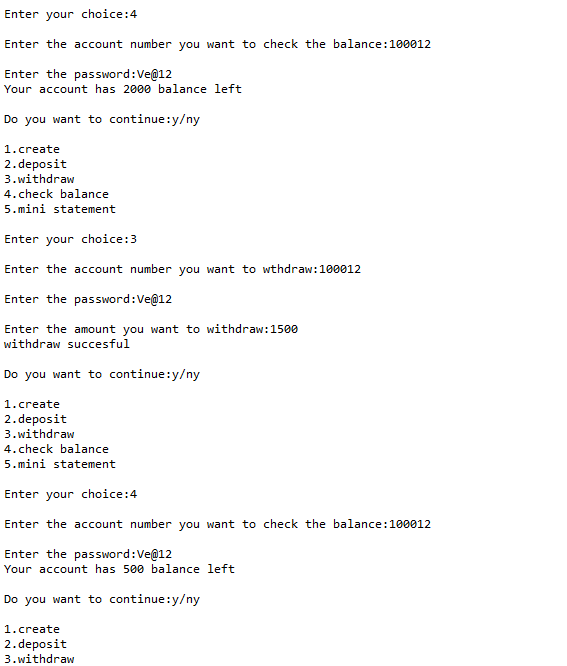
c=input("Do you want to continue:y/n")

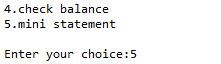
if c=='n' or c=='N':

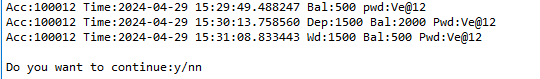
break

**Output:**

****

****

****

****

**Program 6**

**Aim: Write a Python program to implement Library Management System for the following modules by using Oops concept.**

**a) Insert Book Details b) Display Book Details**

**c) Search Book Details d) Update Book details**

**e) Delete Book Details**

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Apr 19 13:09:35 2024

@author: it297

"""

class Library:

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

self.bookno=0

self.price=0

self.author=" "

self.title=" "

self.publish=" "

def insert(self):

self.bookno=int(input("Enter bookNo:"))

self.title=input("Enter title name:")

self.price=int(input("Enter the price of the book:"))

self.author=input("Enter author name:")

self.publish=input("Enter publisher name:")

def display(self,l):

if not l:

print("Library is empty")

return

print("List of books are:")

for i in l: print("Bookno:{},title:{},price:{},author:{},publish:{}".format(i.bookno,i.title,i.price,i.author,i.publish))

if not l:

print("Library is empty")

return

def delete(self,l):

c=int(input("Enter bookno you want to delete:"))

if not l:

print("Library is empty")

return

for i in l:

if i.bookno==c:

l.remove(i)

print("Deleted succesfully")

def search(self,l):

c=int(input("Enter bookno you want to search:"))

for i in l:

if i.bookno==c:

print("Bookno:",i.bookno)

print("title:",i.title)

print("price:",i.price)

print("author:",i.author)

print("publish:",i.publish)

print("record is found:")

else:

print("record is not found:")

def update(self,l):

c=int(input("Enter bookno you want to update:"))

if not l:

print("Library is empty")

return

for i in l:

if i.bookno==c:

#i.bookno=int(input("Enter bookNo:"))

i.title=input("Enter new title name:")

i.price=int(input("Enter the new price of the book:"))

i.author=input("Enter new author name:")

i.publish=input("Enter new publisher name:")

List=[]

while True:

print("1.Insert\n2.display\n3.delete\n4.search\n5.update")

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

if ch==1:

ob=Library()

ob.insert()

List.append(ob)

print("Inserted successfully")

#print(List)

elif ch==2:

ob=Library()

ob.display(List)

elif ch==3:

ob=Library()

ob.delete(List)

elif ch==4:

ob=Library()

ob.search(List)

elif ch==5:

ob=Library()

ob.update(List)

else:

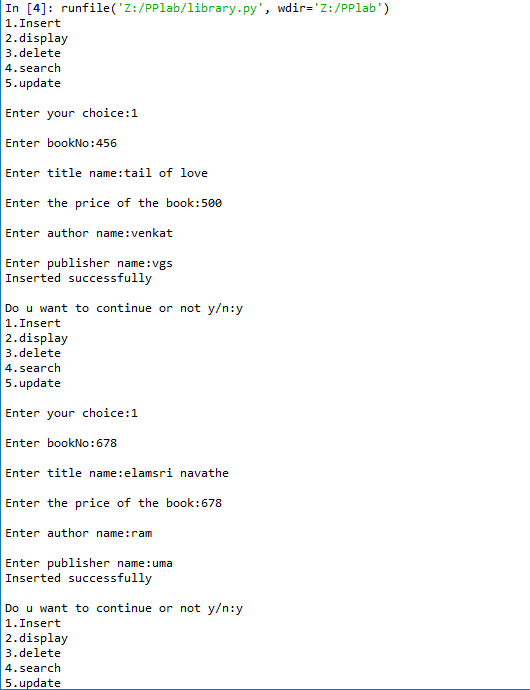
print("Invalid choice")

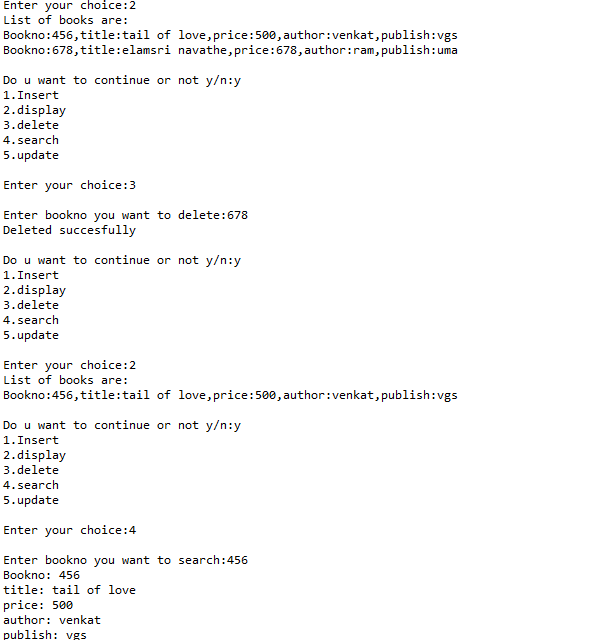
c=input("Do u want to continue or not y/n:")

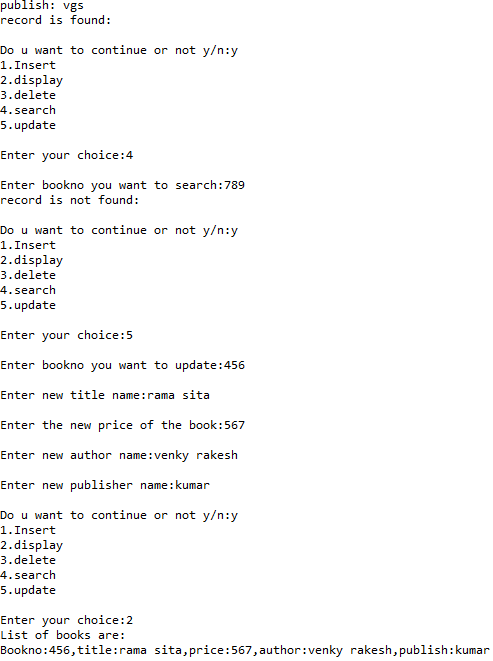
if c=='n' or c=='N':

break

**Output:**

****

****

****

****

**Program 7**

**Aim: Write a Python program to select the type of the user (General user, Prime user, Premium user) and find discount of selected item based on the following:**

|  |  |
| --- | --- |
| **Type of user** | **Discount** |
| 1. **General user** | **No discount** |
| 1. **Prime user** | **10% of the base price** |
| 1. **Premium user** | **15% of the base price** |

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Apr 14 05:55:27 2024

@author: it297

"""

class General:

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

self.laptop=20000

self.mobile=15000

self.WashingMashine=40000

def cal\_lap(self,n):

return n\*self.laptop

def cal\_mob(self,n):

return n\*self.mobile

def cal\_wash(self,n):

return n\*self.WashingMashine

class Prime(General):

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

super().\_\_init\_\_()

self.cooler=35000

self.Fridge=30000

self.Ac=50000

def cal\_lap(self,n):

return n\*(self.laptop-self.laptop\*10/100)

def cal\_mob(self,n):

return n\*(self.mobile-self.mobile\*10/100)

def cal\_wash(self,n):

return n\*(self.WashingMashine-self.WashingMashine\*10/100)

def cal\_cool(self,n):

return n\*(self.cooler-self.cooler\*10/100)

def cal\_fri(self,n):

return n\*(self.Fridge-self.Fridge\*10/100)

def cal\_ac(self,n):

return n\*(self.Ac-self.Ac\*10/100)

class Advance(Prime):

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

super().\_\_init\_\_()

self.oven=35000

self.tv=30000

#self.Ac=50000

def cal\_lap(self,n):

return n\*(self.laptop-self.laptop\*15/100)

def cal\_mob(self,n):

return n\*(self.mobile-self.mobile\*15/100)

def cal\_wash(self,n):

return n\*(self.WashingMashine-self.WashingMashine\*15/100)

def cal\_cool(self,n):

return n\*(self.cooler-self.cooler\*15/100)

def cal\_fri(self,n):

return n\*(self.Fridge-self.Fridge\*15/100)

def cal\_ac(self,n):

return n\*(self.Ac\*15/100)

def cal\_ov(self,n):

return n\*(self.oven-self.oven\*15/100)

def cal\_tv(self,n):

return n\*(self.tv-self.tv\*15/100)

while True:

print("1.General\n2.Prime\n3.Advance")

u=int(input("Enter the type of user:"))

if u==1:

ob=General()

t=0

print("Sorry You have no discount")

name=input("Enter the user name:")

while True:

print("The list of items you can purchase:")

print("1.laptop\n2.mobile\n3.WashingMashine")

se=int(input("Select any items from the above list:"))

if se==1:

n=int(input("How many laptops you want to buy:"))

t+=ob.cal\_lap(n)

print(t)

elif se==2:

n=int(input("How many mobiles you want to buy:"))

t+=ob.cal\_mob(n)

print(t)

elif se==3:

n=int(input("How many washing Mashines you want to buy:"))

t+=ob.cal\_wash(n)

else:

print("Invalid option")

c=input("Do you want to continue as general user(y/n):")

if c=="n" or c=="N":

# break

print("User name is:",name)

print("The bill of general user is:",t)

break

elif u==2:

ob1=Prime()

t1=0

print("You have 10% discount")

name=input("Enter the user name:")

while True:

print("The list of items you can purchase:")

print("1.laptop\n2.mobile\n3.WashingMashine\n4.cooler\n5.fridge\n6.Ac")

se=int(input("Select any items from the above list:"))

if se==1:

n=int(input("How many laptops you want to buy:"))

t1+=ob1.cal\_lap(n)

print(t1)

elif se==2:

n=int(input("How many mobiles you want to buy:"))

t1+=ob1.cal\_mob(n)

print(t1)

elif se==3:

n=int(input("How many washing Mashines you want to buy:"))

t1+=ob1.cal\_wash(n)

elif se==4:

n=int(input("How many coolers you want to buy:"))

t1+=ob1.cal\_cool(n)

print(t1)

elif se==5:

n=int(input("How many fridges you want to buy:"))

t1+=ob1.cal\_fri(n)

print(t1)

elif se==6:

n=int(input("How many acs you want to buy:"))

t1+=ob1.cal\_ac(n)

else:

print("Invalid option")

c=input("Do you want to continue as primel user(y/n):")

if c=="n" or c=="N":

# break

print("User name is:",name)

print("The bill of prime user is:",t1)

break

elif u==3:

ob2=Advance()

t2=0

print("You have 15% discount")

name=input("Enter the user name:")

while True:

print("The list of items you can purchase:")

print("1.laptop\n2.mobile\n3.WashingMashine\n4.cooler\n5.fridge\n6.Ac\n7.oven\n8.tv")

se=int(input("Select any items from the above list:"))

if se==1:

n=int(input("How many laptops you want to buy:"))

t2+=ob2.cal\_lap(n)

print(t2)

elif se==2:

n=int(input("How many mobiles you want to buy:"))

t2+=ob2.cal\_mob(n)

print(t2)

elif se==3:

n=int(input("How many washing Mashines you want to buy:"))

t2+=ob2.cal\_wash(n)

elif se==4:

n=int(input("How many coolers you want to buy:"))

t2+=ob2.cal\_cool(n)

print(t2)

elif se==5:

n=int(input("How many fridges you want to buy:"))

t2+=ob2.cal\_fri(n)

print(t2)

elif se==6:

n=int(input("How many acs you want to buy:"))

t2+=ob2.cal\_ac(n)

elif se==7:

n=int(input("How many ovens do you want to buy:"))

t2+=ob2.cal\_ov(n)

elif se==8:

n=int(input("How many tvs do you want to buy:"))

t2+=ob2.cal\_tv(n)

else:

print("Invalid option")

c=input("Do you want to continue as advance user(y/n)")

if c=="n" or c=="N":

# break

print("User name is:",name)

print("The bill of advance user is:",t2)

break

else:

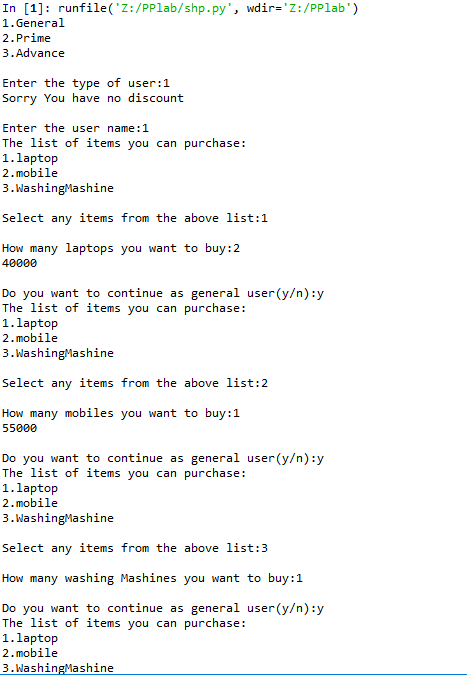
print("Invalid option")

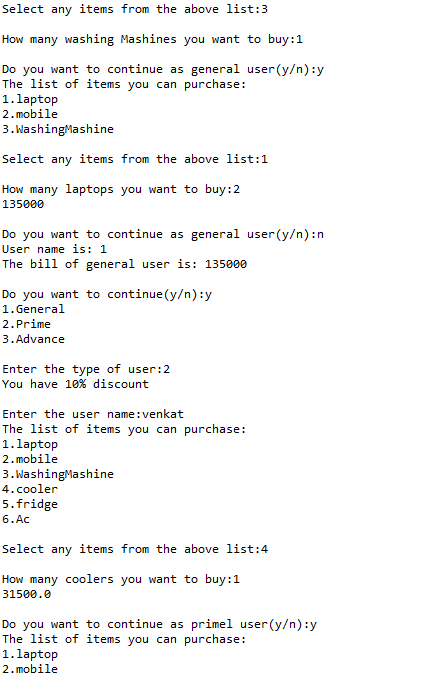
cc=input("Do you want to continue(y/n):")

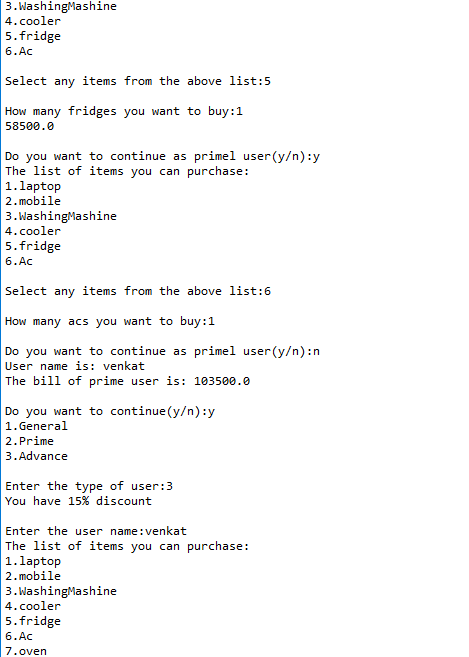
if cc=='n' or cc=='N':

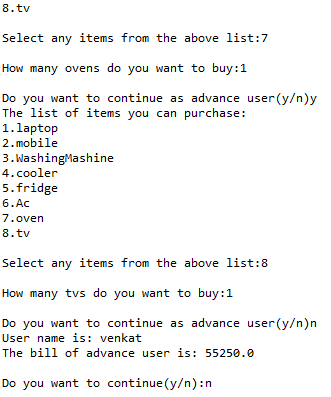
break

**Output:**

****

****

****

****

**Program 8**

**Aim: Write a Python program to implement the SQLite database for the following:**

**a) Create b) Insert c) Display d) Delete e) Search f) Update**

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Apr 14 05:52:52 2024

@author: it297

"""

import sqlite3,re

from datetime import datetime

def create():

try:

con=sqlite3.connect('employee.db')

query='''CREATE TABLE EMP(

ID INTEGER PRIMARYKEY,

NAME TEXT NOT NULL,

EMAIL TEXT NOT NULL UNIQUE,

JOINING\_DATE DATETIME,

SALARY REAL NOT NULL);'''

cursor=con.cursor()

print("successfully connected to SQLITE")

cursor.execute(query)

con.commit()

print("SQLite table created")

cursor.close()

except sqlite3.Error as error:

print("error while creating sqlite table ",error)

finally:

if con:

con.close()

print("SQLite con closed")

def insert():

try:

con=sqlite3.connect('employee.db')

cursor=con.cursor()

print("successfully connected to SQLITE")

id=int(input("Enter Id: "))

name=input("Enter Name: ")

while True:

email=input("Enter email id: ")

if re.search("^[a-z1-9]\*@[a-z.]\*$",email):

break

else:

print("enter valid email")

j=input("Joining Date dd:mm:yyyy: ")

Join=datetime.strptime(j,"%d:%m:%Y").date()

while True:

Sal=int(input("Enter Salary: "))

if Sal<0:

print("enter salary in positive")

else:

break

query="INSERT INTO EMP Values(?,?,?,?,?)"

cursor.execute(query,(id,name,email,Join,Sal))

con.commit()

print("record is inserted successfully into emp table")

cursor.close()

except sqlite3.Error as error:

print("error while creating sqlite table ",error)

finally:

if con:

con.close()

print("SQLite con closed")

def update():

try:

con=sqlite3.connect('employee.db')

cursor=con.cursor()

print("successfully connected to SQLITE")

id=input("enter id to update: ")

query="select \* from emp where id=?"

res=cursor.execute(query,(id,))

records=res.fetchall()

if len(records)==0:

print("record not found")

else:

print("\n1.name\n2.email\n3.joining date\n4.salary\n")

ch=int(input("choose what you want to update: "))

if ch==1:

name=input("enter value to update: ")

query="update emp set name=? where id=?"

cursor.execute(query,(name,id))

elif ch==2:

while True:

email=input("Enter email id: ")

if re.search("^[a-z1-9]\*@[a-z.]\*$",email):

break

else:

print("enter valid email")

query="update emp set email=? where id=?"

cursor.execute(query,(email,id))

elif ch==3:

j=input("enter date dd:mm:yyyy::: ")

joining\_date=datetime.strptime(j,"%d:%m:%Y").date()

query="update emp set joining\_date=? where id=?"

cursor.execute(query,(joining\_date,id))

elif ch==4:

salary=int(input("enter value to update: "))

if salary<0:

print("enter salary in positive number")

else:

query="update emp set salary=? where id=?"

cursor.execute(query,(salary,id))

else:

print("invalid option")

con.commit()

print("record updated successfully")

cursor.close()

except sqlite3.Error as error:

print("error while creating sqlite table ",error)

finally:

if con:

con.close()

print("SQLite con closed")

def search():

try:

l=[]

con=sqlite3.connect('employee.db')

cursor=con.cursor()

print("successfully connected to SQLITE")

i=int(input("enter id to search: "))

query='''select \* from emp;'''

res=cursor.execute(query)

l=res.fetchall()

l1=[]

count=0

for j in range(len(l)):

l1=(l[j])

if i in l1:

count+=1

if count>=1:

print("record found")

query="select \* from emp where id=?"

res=cursor.execute(query,(i,))

rec=res.fetchall()

print(rec)

else:

print("record not found")

cursor.close()

except sqlite3.Error as error:

print("error while creating sqlite table ",error)

finally:

if con:

con.close()

print("SQLite con closed")

return count

def delete():

try:

con=sqlite3.connect('employee.db')

cursor=con.cursor()

print("successfully connected to SQLITE")

id=input("enter id you want to delete: ")

query="select \* from emp where id=?"

res=cursor.execute(query,(id,))

records=res.fetchall()

if len(records)==0:

print("record not found")

else:

query="delete from emp where id=?"

cursor.execute(query,(id,))

con.commit()

print("record is deleted successfully")

cursor.close()

finally:

if con:

con.close()

print("SQLite con closed")

def display():

try:

con=sqlite3.connect('employee.db')

cursor=con.cursor()

print("successfully connected to SQLITE")

query='''select \* from emp'''

res=cursor.execute(query)

records=res.fetchall()

print("total rows are: ",len(records))

print("id\tname\t\t\temail\t\t\tjoining date\t\tsalary")

for row in records:

print(row[0],"\t",row[1],"\t\t",row[2],"\t",row[3],"\t\t",row[4])

cursor.close()

except sqlite3.Error as error:

print("error while creating sqlite table ",error)

finally:

if con:

con.close()

print("SQLite con closed")

while True:

print("1.create\n2.insert\n3.update\n4.display\n5.delete\n6.search\n7.exit")

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

if ch==1:

create()

elif ch==2:

insert()

elif ch==3:

update()

elif ch==4:

display()

elif ch==5:

delete()

elif ch==6:

search()

elif ch==7:

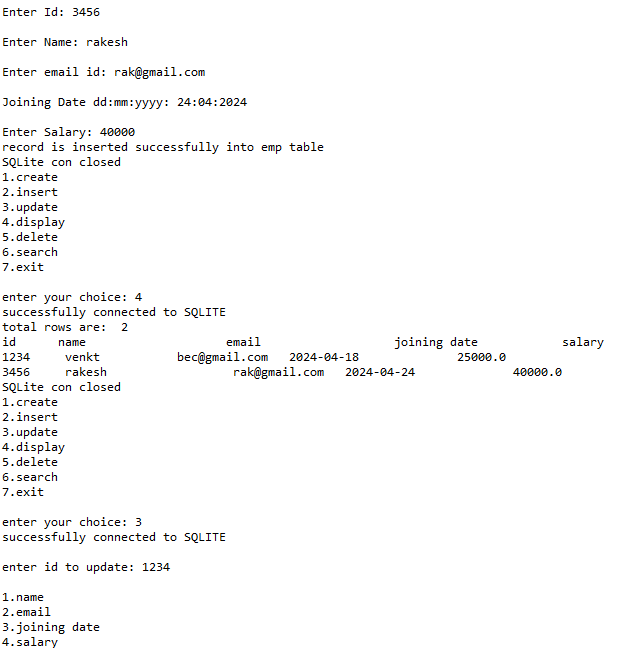
break

else:

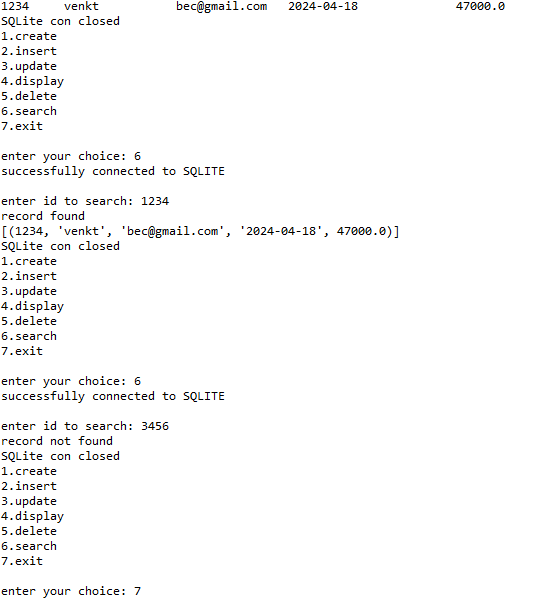
print("invalid option")

**Output:**

****

****

****

****

**Program 9**

**Aim: Write a Python program to perform all arithmetic operations on two numbers using GUI module.**

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Wed May 1 13:51:25 2024

@author: it297

"""

from tkinter import \*

class Mywindow:

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

self.lbl1=Label(win,text='First number')

self.lbl2=Label(win,text='Second number')

self.lbl3=Label(win,text='Result')

self.t1=Entry()

self.t2=Entry()

self.t3=Entry()

self.lbl1.place(x=100,y=50)

self.t1.place(x=200,y=50)

self.lbl2.place(x=100,y=100)

self.t2.place(x=200,y=100)

self.b1=Button(win,text='Add',command=self.add)

self.b2=Button(win,text='Subtract',command=self.sub)

self.b3=Button(win,text='Multiply',command=self.mul)

self.b4=Button(win,text='Divide',command=self.div)

self.b1.place(x=100,y=150)

self.b2.place(x=200,y=150)

self.b3.place(x=100,y=200)

self.b4.place(x=200,y=200)

self.lbl3.place(x=100,y=250)

self.t3.place(x=200,y=250)

def add(self):

num1=int(self.t1.get())

num2=int(self.t2.get())

result=num1+num2

self.t3.insert(END,str(result))

def sub(self):

num1=int(self.t1.get())

num2=int(self.t2.get())

result=num1-num2

self.t3.insert(END,str(result))

def mul(self):

num1=int(self.t1.get())

num2=int(self.t2.get())

result=num1\*num2

self.t3.insert(END,str(result))

def div(self):

num1=int(self.t1.get())

num2=int(self.t2.get())

result=num1/num2

self.t3.insert(END,str(result))

window=Tk()

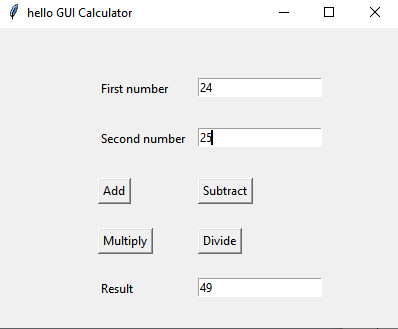
mywin=Mywindow(window)

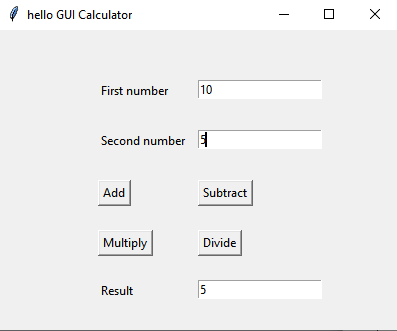
window.title('hello GUI Calculator')

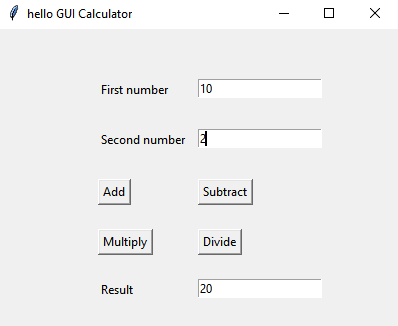
window.geometry("400x300+10+10")

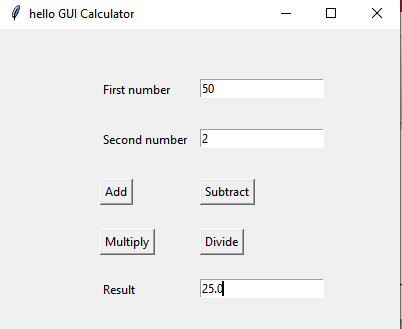
window.mainloop()

**Output:**

****

****

****

****

**Program 10**

**Aim: Write a Python program to read Excel file(student.xlsx) using pandas library and print the following:**

**a) Print all the student details of multiple sheets of a given Excel file.**

**b) Sort student details based on marks.**

**c) Print the student details whose name starts with ‘s’.**

**d) Print the student details specific range of marks.**

**e) Print the student details for based on registration number.**

**Source Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Apr 30 10:16:13 2024

@author: it297

"""

import pandas as pd

def read():

d=pd.read\_excel("student.xlsx",sheet\_name=0,index\_col=0)

d1=pd.read\_excel("student.xlsx",sheet\_name=1,index\_col=0)

read=pd.concat([d,d1])

print(read)

def sort():

d=pd.read\_excel("student.xlsx",sheet\_name=0,index\_col=0)

d1=pd.read\_excel("student.xlsx",sheet\_name=1,index\_col=0)

read=pd.concat([d,d1])

s=read.sort\_values(['Marks'],ascending=True)

print("Student details after sorting their marks: ")

print(s)

def startswith():

d=pd.read\_excel("student.xlsx",sheet\_name=0,index\_col=1)

d1=pd.read\_excel("student.xlsx",sheet\_name=1,index\_col=1)

read=pd.concat([d,d1])

filtered = read[read['Name'].str.startswith('s')]

print("List of student details whose name starts with s: ")

print(filtered)

def srange():

import pandas as pd

d=pd.read\_excel("student.xlsx",sheet\_name=0,index\_col=0)

d1=pd.read\_excel("student.xlsx",sheet\_name=1,index\_col=0)

read=pd.concat([d,d1])

def find(sr,er):

filtered=read[(read['Marks']>=sr)&(read['Marks']<=er)]

return filtered

sr=int(input("Enter start range of marks"))

er=int(input("Enter end range of marks"))

students=find(sr,er)

print("Students with marks between",sr,"and",er,":")

print(students)

def regno():

d=pd.read\_excel("student.xlsx",sheet\_name=0,index\_col=0)

d1=pd.read\_excel("student.xlsx",sheet\_name=1,index\_col=0)

read=pd.concat([d,d1])

def details(reg\_num):

sdetails=read[read['Regd.No']==reg\_num]

return sdetails

rg=input("Enter registration number you want to the details of by spaces :")

reg\_num=rg.split(" ")

for reg in reg\_num:

print("student details for registration number",reg,":")

print(details(reg))

print()

while True:

print("\n1.print all student details\n2.sorting of marks\n3.name starts with an alphabet s\n4.to print specific range of marks\n5.to print student details based on registration number\n6.exit\n")

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

if ch==1:

read()

elif ch==2:

sort()

elif ch==3:

startswith()

elif ch==4:

srange()

elif ch==5:

regno()

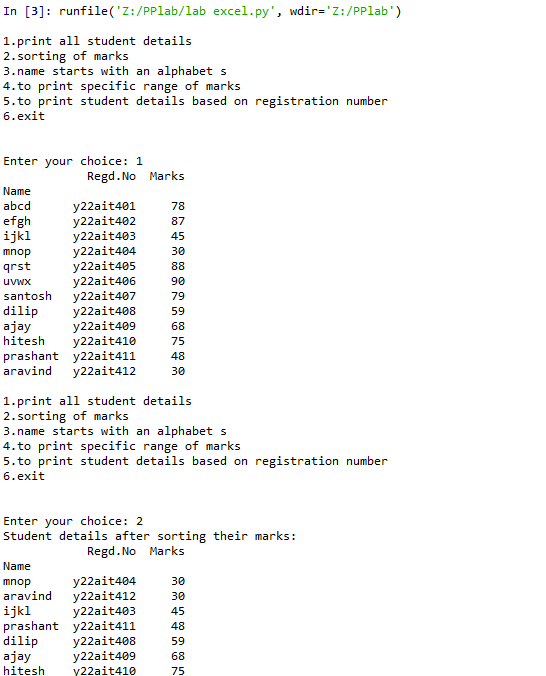
elif ch==6:

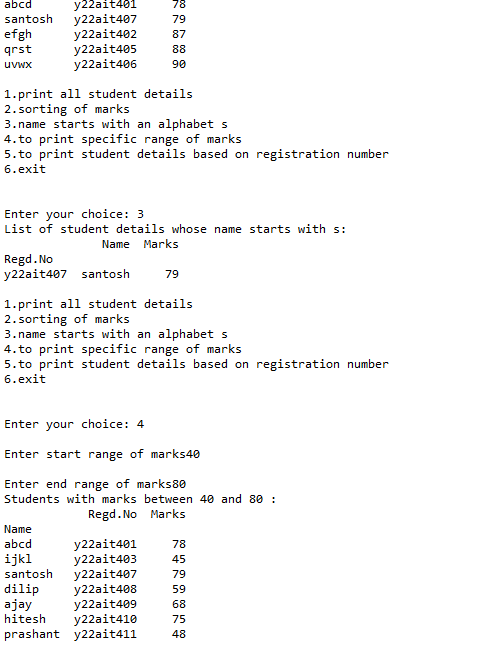
break

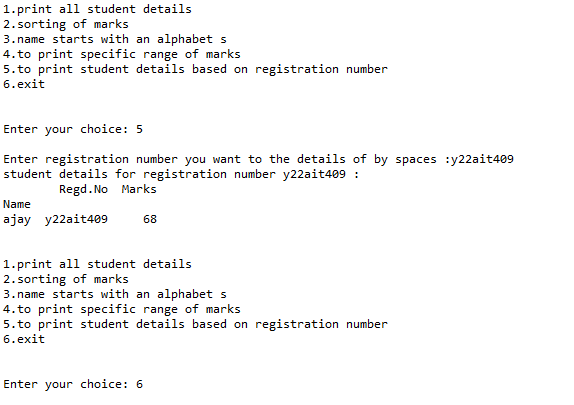
else:

print("enter a valid choice(1/2/3/4/5/6)")

**Output:**

****

****

****