**Python Basics**

1.Write a program using operator overloading to input two complex and to find:

(a)Sum of two complex numbers

(b)Multiplication of the two numbers

**Code:**

a=int(raw\_input("Enter the 1st real part"))

b=int(raw\_input("Enter the 1st img part"))

c=int(raw\_input("Enter the 2nd real part"))

d=int(raw\_input("Enter the 2nd img part"))

e=complex(a,b)

f=complex(c,d)

print e

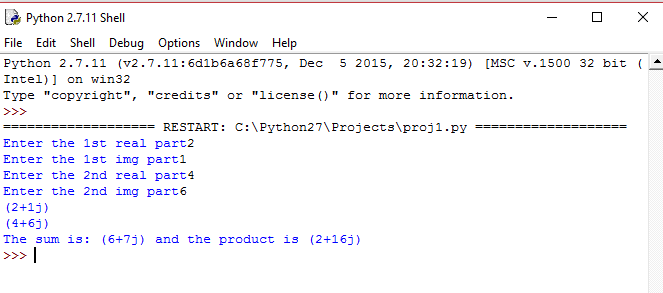
print f

g=e+f

h=e\*f

print "The sum is:",g,"and the product is",h

**Output:**

****

2.Write string method used to implement the following:

(a)To count the number of characters in the string.

(b)To change the first character in the string to capital letter.

(c)To check whether given character is letter or number.

(d)To change lower case to uppercase letter.

**Code:**

a=str(raw\_input("Enter a string."))

b=len(a)

c=a.capitalize()

d=a.isalnum()

e=a.upper()

print"The number of characters are:",b

print"The first letter capitalized:",c

if d==True:

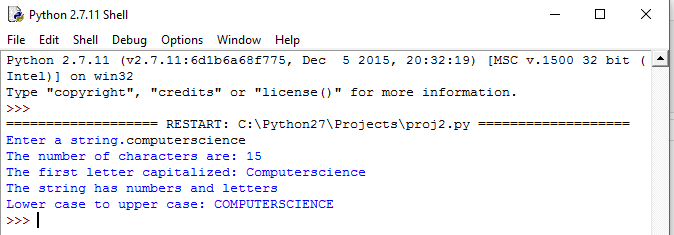
print"The string has numbers and letters"

else:

pass

print"Lower case to upper case:",e

**Output:**

****

3.Pay roll information system:

Declare the base class 'employee' with employee's number,name,designation,address,phone

number.Define and declare the function getdata() and putdata() to get the employee's details and

print employee's details.

Declare the derived class salary with basic pay,DA,HRA,Gross pay,PF,Income tax and net pay. Declare

define the function getdata1() to call getdata and get the basic pay. Define the function calculate()

to find the net pay. Define the function display() to call putdata() and display salary details.

Create the derived class object. Read the number of employees. Call the function getdata1() and

calculate() to each employee. Call the display() function.

**Code:**

class employee:

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

self.ENo=0

self.EName=""

self.Desig=""

self.Address=""

self.PNo=0

def getdata(self):

self.ENo=input("Enter the employee id: ")

self.EName=raw\_input("Enter the employee name: ")

self.Desig=raw\_input("Enter designation: ")

self.Address=raw\_input("Enter the address: ")

self.PNo=input("Enter the phone number: ")

def putdata(self):

print "Employee Number :",self.ENo

print "Employee Name: ",self.EName

print "Employee Designation: ",self.Desig

print "Employee Address: ",self.Address

print "Employee Phone Number: ",self.PNo

class salary(employee):

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

employee.\_\_init\_\_(self)

self.Basic=0

self.DA=0

self.HRA=0

self.Gross=0

self.PF=0

selfIncomeTax=0

self.NetPay=0

def getdata1(self):

employee.getdata(self)

self.Basic=input("Enter the Basic Pay: ")

print "----------------------------------------------------------"

def calculate(self):

self.NetPay=self.Basic+(self.Basic\*0.3)+(self.Basic\*0.8)+(self.Basic\*0.1)-(self.Basic\*0.5)-(self.Basic\*0.15)

return self.NetPay

def display(self):

employee.putdata(self)

print "Employee's Net Pay :",self.calculate()

print "----------------------------------------------------------"

n=int(raw\_input("Enter the number of Employees :"))

global l1

l1=list()

for x in range (n):

a=salary()

l1.append(a)

a.getdata1()

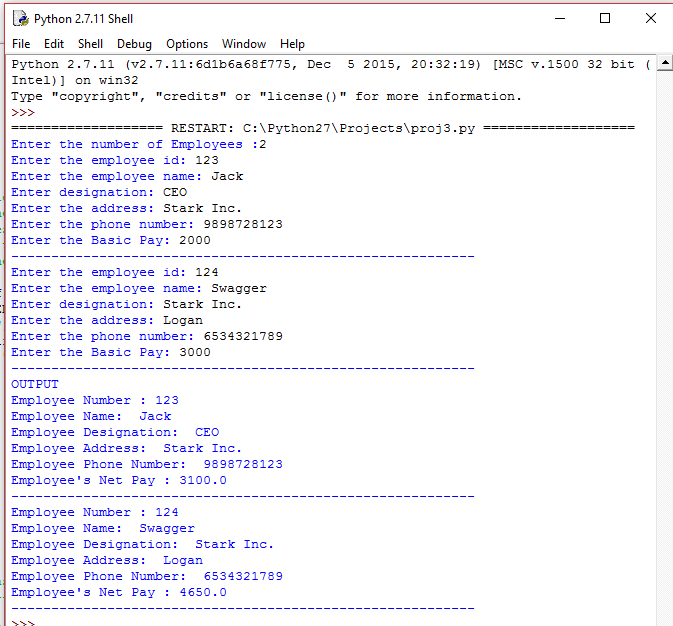
a.calculate()

print "OUTPUT"

for i in l1:

i.display()

**Output:**



4.Implement the following using multilevel information. Create a student class with student number

and name. Class graduate is created by using student. Graduate class is created using subject code

and subject name. Class Post Graduate is created by using Graduate. Post Graduate class is created

using master subject code and master subject name.

**Code:**

class Student:

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

self.StNo=0

self.StName=""

def readstud(self):

self.StNo=raw\_input("Enter Student code: ")

self.StName=raw\_input("Enter Student Name: ")

def display(self):

print "Student Code:",self.StNo

print "Student Name:",self.StName

class Graduate(Student):

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

Student.\_\_init\_\_(self)

self.Subcode=0

self.Subname=""

def readgrad(self):

Student.readstud(self)

self.Subcode=raw\_input("Enter Subject Code: ")

self.Subname=raw\_input("Enter Subject Name: ")

def display2(self):

Student.display(self)

print "Subject Code:",self.Subcode

print "Subject Name:",self.Subname

class PostGraduate(Graduate,Student):

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

Graduate.\_\_init\_\_(self)

Student.\_\_init\_\_(self)

self.MasterSubcode=0

self.MasterSubname=""

def readpost(self):

Graduate.readgrad(self)

self.MasterSubcode=raw\_input("Enter Master Subject Code: ")

self.MasterSubname=raw\_input("Enter Master Subject Name: ")

def display3(self):

Graduate.display2(self)

print "Master Subject Code:",self.MasterSubcode

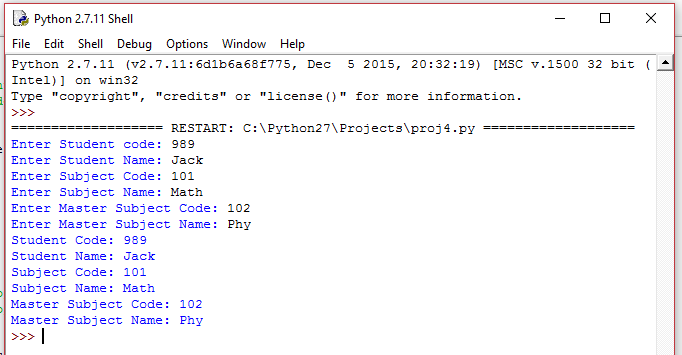
print "Master Subject Name:",self.MasterSubname

a=PostGraduate()

a.readpost()

a.display3()

**Output:**

****

5.Define a class ITEMINFO in Python with the following description:

Icode(Item Code), Item(Item Name), Price(Price of each item), Qty(Quantity in stock), Discount

(Discount percentage on the item), Netprice(Final Price) Methods A member function FindDisc() to

calculate discount as per the following rule: If Qty<=10 discount is zero. If Qty is from 11 to 20

discount is 15. If Qty>=20 discount is 20. A constructor(\_\_init\_\_method) to assign the value with 0

for Icode,Price,Qty,Netprice,discount and null for Item respectively. A function Buy() to allow user to

enter values for Icode,Item,Price,Qty and call function FindDisc() to calculate the discount and

Netprice(Price\*(Quantity-Discount/100)). A function ShowAll() to allow user to view the content of

all the data members.

**Code:**

class ITEMINFO:

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

self.ICode = 0

self.IName = None

self.Price = 0

self.Qty = 0

self.Discount = 0

self.NetPrice = 0

def Buy(self):

print ("Enter item details...")

self.ICode = int(raw\_input("Enter item code: "))

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

self.Price = float(raw\_input("Enter item price: "))

self.Qty = int(raw\_input("Enter item quantity: "))

self.Discount=self.FindDisc()

def FindDisc(self):

if self.Qty <= 10:

self.Discount = 0.0

elif self.Qty >= 11 and self.Qty <= 20:

self.Discount = 15.0

elif self.Qty > 20:

self.Discount = 20.0

self.NetPrice=((self.Price)\*(self.Qty))-(((self.Price)\*(self.Qty)\*(self.Discount))/100)

return self.Discount

def ShowAll(self):

print ("Item details...")

print("Item code:", self.ICode)

print("Item name:", self.IName)

print("Item price:", self.Price)

print("Item quantity:", self.Qty)

print("Discount percentage:", self.Discount)

print("Net price is:", self.NetPrice)

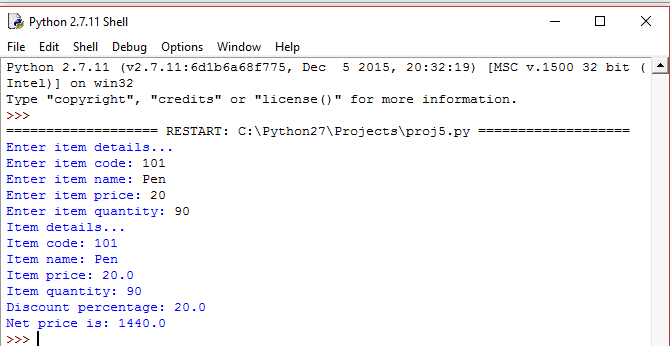
item=ITEMINFO()

item.Buy()

item.FindDisc()

item.ShowAll()

**Output:**

****

6.Railway Reservation System: Declare the base class Train with train number,name,Starting Station,

arrival time,etc. Define and declare the function getdata() and putdata() to get the train details and

print the details. Deine search() function to search train detail using train number. Declare the

derived class passenger with ticket number, PNR name of the passenger,gender, age, address,phone

number, etc. Declare and define the function getdata1() to call search() function to get the train

details and get the passenger's information. Define the function display() to call putdata() and

display passenger's details. Create base class object. Read the number of trains. Create the derived

class object. Read the number of passengers. Call the function getdata1() to each passenger. Call the

display() function.

**Code:**

class Train:

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

self.TNo=0

self.TName=""

self.SStat=""

self.DStat=""

self.DTime=[]

self.ATime=[]

self.list=[]

self.trainno=0

def getdata(self):

self.TNo=int(raw\_input("Enter the train number: "))

self.TName=raw\_input("Enter the train name: ")

self.SStat=raw\_input("Enter starting station: ")

self.DStat=raw\_input("Enter destination station: ")

self.DTime=eval(raw\_input("Enter the depature time (hh,mm,ss): "))

self.ATime=eval(raw\_input("Enter the arrival time (hh,mm,ss): "))

def putdata(self):

print "The train no: ",self.TNo

print "The train name: ",self.TName

print "The starting station: ",self.SStat

print "The destination station: ",self.DStat

print "The departure time: ",self.DTime

print "The arrival time: ",self.ATime

@staticmethod

def search(trainno):

for z in l1:

if z.TNo==trainno:

z.putdata()

break

else:

print "Train not found"

class Passenger(Train):

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

Train.\_\_init\_\_(self)

self.PNR=0

self.Tno=0

self.Name=""

self.gender=""

self.age=0

self.address=""

self.Pno=0

self.trainno

def getdata1(self):

self.PNR=input("Enter PNR: ")

self.Tno=input("Enter Train no: ")

Train.search(self.Tno)

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

self.gender=raw\_input("Enter gender: ")

self.age=input("Enter age:")

self.address=raw\_input("Enter address: ")

self.Pno=input("Phone number: ")

def display(self):

for z in l1:

if z.TNo==self.Tno:

z.putdata()

print "PNR of passenger: ",self.PNR

print "Name of passenger: ",self.Name

print "Gender: ",self.gender

print "Passenger Age: ",self.age

print "Passenger Address: ",self.address

l1=[]

l2=[]

tno=int(raw\_input("Enter the no of trains being entered: "))

for x in range (tno):

a=Train()

l1.append(a)

a.getdata()

pno=int(raw\_input("Enter the no. of passengers: "))

for i in range (pno):

b=Passenger()

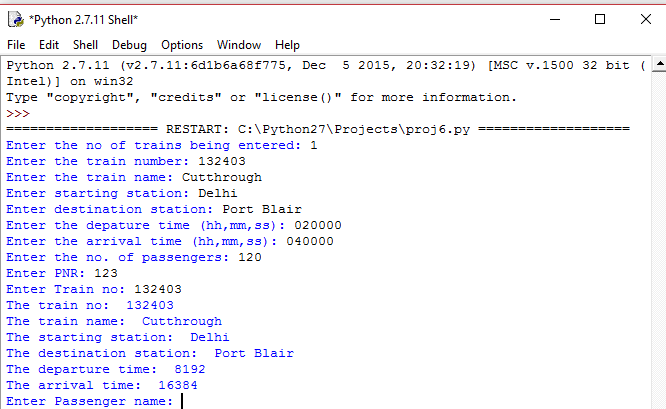
l2.append(b)

b.getdata1()

for y in l2:

y.display()

**Output:**

****

7.Write a menu driven program to perform all mathematical operations using the concept of operator

overloading.

**Code:**

class mathfunc:

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

self.sum=0.0

self.dif=0.0

self.pro=0.0

self.quo=0.0

self.rem=0.0

self.floor=0.0

self.num1=0.0

self.num2=0.0

def \_\_add\_\_(self):

self.num1=float(raw\_input("Enter first number: "))

self.num2=float(raw\_input("Enter the second number: "))

self.sum=self.num1+self.num2

print self.sum

def \_\_sub\_\_(self):

self.num1=float(raw\_input("Enter the number to be subtracted: "))

self.num2=float(raw\_input("Enter the number from which you want to subtract: "))

self.dif=self.num2-self.num1

print self.dif

def \_\_mult\_\_(self):

self.num1=float(raw\_input("Enter first number: "))

self.num2=float(raw\_input("Enter the second number: "))

self.pro=self.num1\*self.num2

print self.pro

def \_\_div\_\_(self):

self.num1=float(raw\_input("Enter the dividend : "))

self.num2=float(raw\_input("Enter the dividor: "))

self.quo=self.num1/self.num2

print self.quo

def \_\_mod\_\_(self):

self.num1=float(raw\_input("Enter the dividend for modulo : "))

self.num2=float(raw\_input("Enter the dividor for modulo: "))

self.rem=self.num1/self.num2

print self.rem

def \_\_floor\_\_(self):

self.num1=float(raw\_input("Enter first number: "))

self.num2=float(raw\_input("Enter the second number: "))

decide=int(raw\_input("Divide what by what? 1>num1/num2 <2>num2/num1: "))

if decide==1:

self.floor=self.num1/self.num2

elif decide==2:

self.floor=self.num2/self.num1

print self.floor

def \_\_fundecide\_\_(self):

print "1.Addition"

print "2.Subtraction"

print "3.Multipliction"

print "4.Division"

print "5.Modulo"

print "6.Floor Division"

choice=int(raw\_input("Enter the function to be performed:"))

if choice==1:

self.\_\_add\_\_()

elif choice==2:

self.\_\_sub\_\_()

elif choice ==3:

self.\_\_mult\_\_()

elif choice==4:

self.\_\_div\_\_()

elif choice==5 :

self.\_\_mod\_\_()

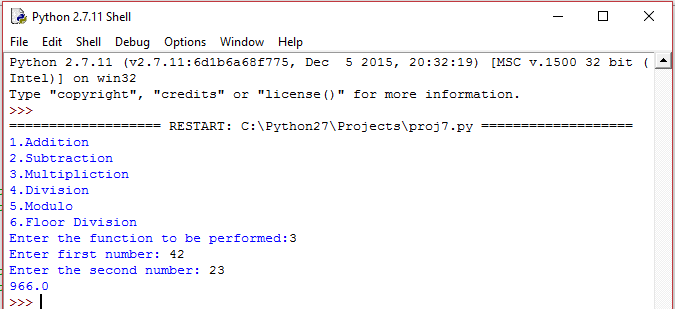
else:

self.\_\_floor\_\_()

a=mathfunc()

a.\_\_fundecide\_\_()

**Output:**

****

8.Write a menu driven program to perform basic list operations.

(a) Traversal in a list

(b) Insertion in a sorted list

(c) Deletion of an element from a list

**Code:**

import bisect

def traversal(List):

size=len(List)

for i in range(size):

print List[i],

def insertion(List):

Item=int(raw\_input("Enter element to be inserted in sorted list: "))

List.sort()

bisect.insort(List,Item)

print "List after adding ",Item," in sorted manner is "

print List

def deletion(List,Item):

beg=0

last=len(List)-1

while beg<=last:

mid=(beg+last)/2

if List[mid]==Item:

return mid

elif List[mid]>Item:

beg=mid+1

else:

last=mid-1

N=int(raw\_input("Enter the number of elements in a list: "))

List=[0]\*N

for x in range(N):

List[x]=int(raw\_input("Enter element "+str(x+1)+": "))

choice=int(raw\_input("Enter the function number to be performed:<1>Traversal <2>Insertion <3>Deletion: "))

if choice==1:

traversal(List)

elif choice==2:

insertion(List)

else:

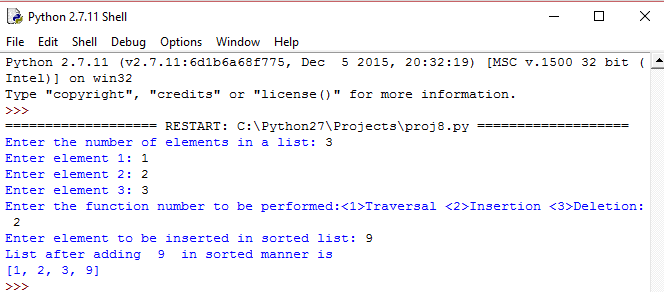
Item=int(raw\_input("Enter element to be deleted in list: "))

index=int(deletion(List,Item))

del List[index]

print "The list after deletion of element is: ",List

**Output:**

****

9. Write a menu-driven program to perform searching techniques in a list string:

(a) Linear search

(b) Binary Search

**Code:**

def LSearch(List,Item):

i=0

while i<len(List) and List[i]<>Item:

i+=1

if i<len(List):

if i==0:

return "0"

else:

return i

else:

return False

def BSearch(List,Item):

beg=0

last=len(List)-1

while beg<=last:

mid=(beg+last)/2

if List[0]==Item:

return "0"

else:

if List[mid]==Item:

return mid

elif List[mid]>Item:

beg=mid+1

else:

last=mid-1

else:

return False

N=int(raw\_input("Enter the number of elements in list: "))

List=[0]\*N

for x in range(N):

List[x]=int(raw\_input("Enter element "+str(x+1)+" : "))

Item=int(raw\_input("Enter the element to be searched for: "))

choice=int(raw\_input("Which search method to be used?: <1>Linear search <2>Binary Search: "))

if choice==1:

index1=LSearch(List,Item)

if index1:

print "Found element at index ",int(index1), "or position ",int(index1)+1

else:

print "The element doesn't exist!!"

else:

index2=BSearch(List,Item)

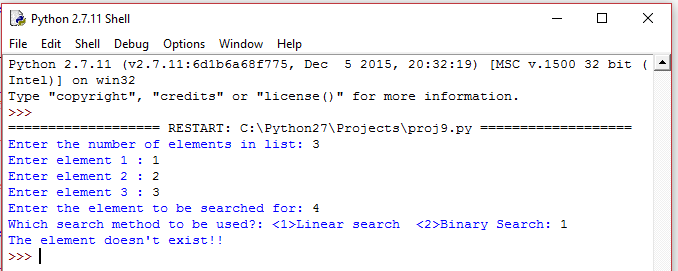
if index2:

print "Found element at index ",int(index2), "or position ",int(index2)+1

else:

print "Element doesn't exist"

**Output:**

****

10. Write a menu-driven program to perform sorting of a list:

(a) Selection Sort

(b) Bubble Sort

(c) Insertion Sort

**Code:**

L1=[]

n=int(raw\_input("Enter the length of your list"))

for i<n:

z=raw\_input("Enter the element")

L1.append(z)

i=i+1

print"1.BubbleSort"

print"2.SelectionSort"

print"3.InsertionSort"

ch=int(raw\_input("Enter your choice."))

if ch==1:

def bubblesort(mylist):

moreswaps=True

while moreswaps:

moreswaps=False

for element in range(len(mylist)-1):

if mylist[element]>mylist[element+1]:

moreswaps=True

temp=mylist[element]

mylist[element]=mylist[element+1]

mylist[element+1]=temp

return mylist

a=bubblesort(L1)

print a

elif ch==2:

def selectionsort(mylist):

curpos=0

for position in range(len(mylist)):

minpos=position

for scanpos in range(position+1,len(mylist)):

if mylist[scanpos]<mylist[minpos]:

minpos=scanpos

temp=mylist[minpos]

mylist[minpos]=mylist[curpos]

mylist[curpos]=temp

return mylist

b=selectionsort(L1)

print b

elif ch==3:

def insertsort(mylist):

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

v=L1[i]

j=i

while L1[j-i]>v and j>=1:

L1[j]=L1[j-1]

j=j-1

L1[j]=v

return mylist

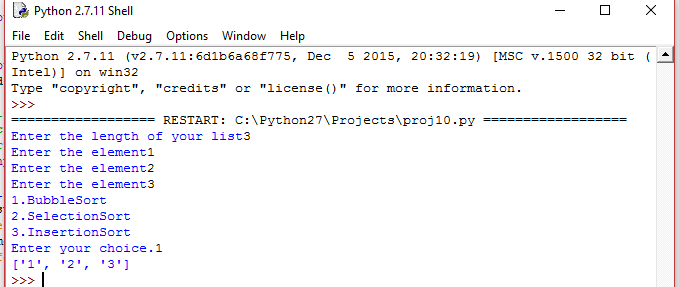
c=insertsort(L1)

print c

else:

print"Invalid Input"

**Output:**

****

11. Write a menu-driven program using list to perform the following operations

(a)Insert an element

(b)Delete an element

(c)Display the contents of the list

(d)Exit

**Code:**

**i**mport bisect

def insertion(List):

Item=int(raw\_input("Enter the Item to be inserted: "))

bisect.insort(List,Item)

print List

def deletion(List,Item):

beg=0

last=len(List)-1

while beg<=last:

if List[0]==Item:

return "0"

else:

mid=(beg+last)/2

if List[mid]==Item:

return mid

elif List[mid]>Item:

beg=mid+1

else:

last=mid-1

def display(List):

for x in range(len(List)):

print "Element ",x+1," : ",List[x]

N=int(raw\_input("Enter the number of elements in the list: "))

List=[0]\*N

for x in range(N):

List[x]=int(raw\_input("Enter element "+str(x+1)+" : "))

while True:

choice=int(raw\_input("perform what? <1>Insertion <2>Deletion <3>Display <4>Exit: "))

if choice==1:

insertion(List)

elif choice==2:

Item=int(raw\_input("Enter element to be deleted: "))

index=deletion(List,Item)

del List[index]

print List

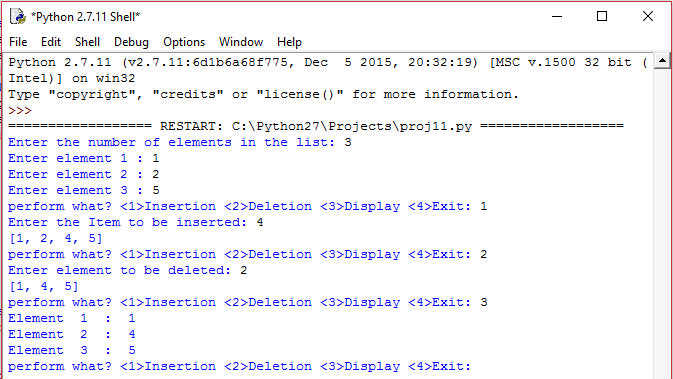
elif choice==3:

display(List)

else:

exit()

**Output:**

****

12.Write a program using lists in python to:

a) Find the sum of two matrices

b) Find the difference of two matrices

c) Product of two matrices

d) Exit

**Code:**

a=raw\_input("Enter 1,1")

b=raw\_input("Enter 1,2")

c=raw\_input("Enter 1,3")

d=raw\_input("Enter 2,1")

e=raw\_input("Enter 2,1")

f=raw\_input("Enter 2,1")

g=raw\_input("Enter 3,1")

h=raw\_input("Enter 3,1")

i=raw\_input("Enter 3,1")

print "second matrix"

j=raw\_input("Enter 1,1")

k=raw\_input("Enter 1,1")

l=raw\_input("Enter 1,1")

m=raw\_input("Enter 2,1")

n=raw\_input("Enter 2,1")

o=raw\_input("Enter 2,1")

p=raw\_input("Enter 3,1")

q=raw\_input("Enter 3,1")

r=raw\_input("Enter 3,1")

X = [[a,b,c],

[d,e,f],

[g,h,i]]

Y = [[j,k,l],

[m,n,o],

[p,q,r]]

result = [[0,0,0],

[0,0,0],

[0,0,0]]

# iterate through rows

for i in range(len(X)):

# iterate through columns

for j in range(len(X[0])):

result[i][j] = int(X[i][j]) + int(Y[i][j])

for r in result:

print(r)

# iterate through rows

for i in range(len(X)):

# iterate through columns

for j in range(len(X[0])):

result[i][j] = int(X[i][j]) - int(Y[i][j])

for r in result:

print(r)

# iterate through rows

for i in range(len(X)):

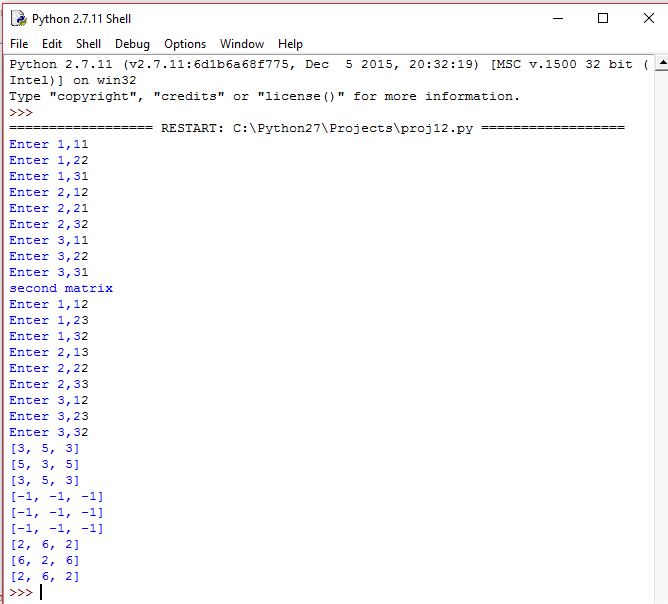
# iterate through columns

for j in range(len(X[0])):

result[i][j] = int(X[i][j]) \* int(Y[j][i])

for r in result:

print(r)

**Output:**

13.Write a program to implement Stack Operations as a list

a) push

b) pop

c)peek

d) Display Stack

e)Exit

**Code:**

L1=[]

n=int(raw\_input("Enter the length you want."))

for i in range(n):

a=raw\_input("Enter the element")

L1.append(a)

print L1

def is\_empty(stk):

if stk == []:

return True

else:

return False

def Push(stk,item):

stk.append(item)

top=len(stk)+ 1

def Pop(stk):

if is\_empty(stk):

print"Underflow"

else:

item=stk.pop()

if len(stk)==0:

top=None

else :

top=len(stk)-1

return item

def Peek(stk):

if is\_empty(stk):

print"Underflow"

else:

top=len(stk)-1

print stk[top]

def Display(stk):

for i in range(n):

print L1[i]

ans=1

while ans==1:

print"What do you want to do?"

print"1.Push"

print"2.Pop"

print"3.Peek"

print"4.Display"

ch=int(raw\_input("Enter your choice number."))

if ch==1:

x=raw\_input("Enter the item")

Push(L1,x)

print L1

elif ch==2:

Pop(L1)

print L1

elif ch==3:

Peek(L1)

print L1

elif ch==4:

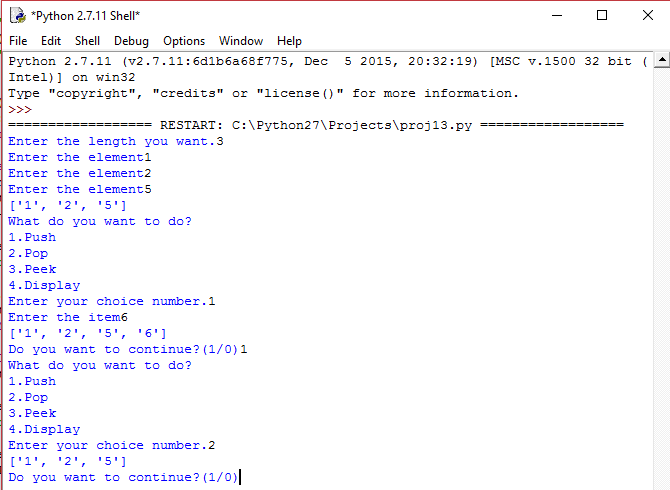
Display(L1)

else:

print"Invalid input"

ans=int(raw\_input("Do you want to continue?(1/0)"))

**Output:**

****

14.Write a program to implement Queue Operations as a list

a) Enqueue

b) Dequeue

c)peek

d) Display Queue

e)Exit

**Code:**

L1=[]

n=int(raw\_input("Enter the length you want."))

for i in range(n):

a=raw\_input("Enter the element")

L1.append(a)

print L1

def is\_empty(stk):

if stk == []:

return True

else:

return False

def Push(stk,item):

stk.append(item)

top=len(stk)+ 1

def Pop(stk):

if is\_empty(stk):

print"Underflow"

else:

item=stk.pop(0)

if len(stk)==0:

top=None

else :

top=len(stk)-1

return item

def Peek(stk):

if is\_empty(stk):

print"Underflow"

else:

top=len(stk)-1

print stk[top]

def Display(stk):

for i in range(n):

print L1[i]

ans=1

while ans==1:

print"What do you want to do?"

print"1.Push"

print"2.Pop"

print"3.Peek"

print"4.Display"

ch=int(raw\_input("Enter your choice number."))

if ch==1:

x=raw\_input("Enter the item")

Push(L1,x)

print L1

elif ch==2:

Pop(L1)

print L1

elif ch==3:

Peek(L1)

print L1

elif ch==4:

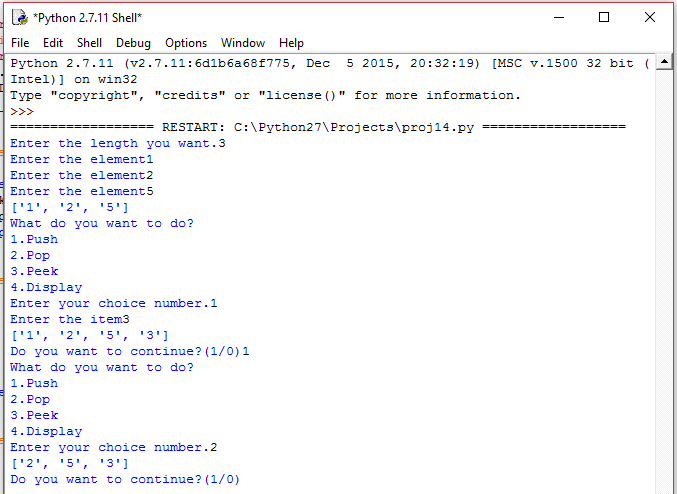
Display(L1)

else:

print"Invalid input"

ans=int(raw\_input("Do you want to continue?(1/0)"))

**Output:**

****

15. Define the class with following specification.

Class Donor

Private members:

Name, Date of birth, address, phone number, blood group

Public members:

Input()(create the file of donor)

Append()(append a record in the file)

Show()(display the record)

Search()(given the blood group displays name, address of the donor)

All the data must be stored in a file

**Code:**

class Donor:

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

self.\_\_name=''

self.\_\_DOB=''

self.\_\_address=''

self.\_\_pno=0

self.\_\_bgrp=''

def Input(self):

self.\_\_name=raw\_input("Enter the name of the donor: ")

self.\_\_DOB=raw\_input("Enter the date of birth: ")

self.\_\_address=raw\_input("Enter the address: ")

self.\_\_pno=int(raw\_input("Enter phone number: "))

self.\_\_bgrp=raw\_input("Enter the blood group: ")

a=open(self.\_\_name,'w')

a.close()

def Append(self):

name="Donor name: "+str(self.\_\_name)+'\n'

dob="Date of birth: "+str(self.\_\_DOB)+'\n'

add="Address :"+str(self.\_\_address)+'\n'

pn="Phone number: "+str(self.\_\_pno)+'\n'

bg="Blood group: "+str(self.\_\_bgrp)+'\n'

l=[name,dob,add,pn,bg]

a=open(self.\_\_name,'a+')

a.writelines(l)

a.close()

def Show(self):

a=open(self.\_\_name,'r')

string=a.read()

print

print string

a.close()

@staticmethod

def Search():

bgp=raw\_input("Enter the blood group for details: ")

for x in l:

if x.\_\_bgrp==bgp:

print "Name : ",x.\_\_name

print "Address: ",x.\_\_address

no=int(raw\_input("Enter the number of donors: "))

print

l=[]

for y in range (no):

obj=Donor()

obj.Input()

print

obj.Append()

l.append(obj)

for x in l:

print "DETAILS"

x.Show()

print "-------"

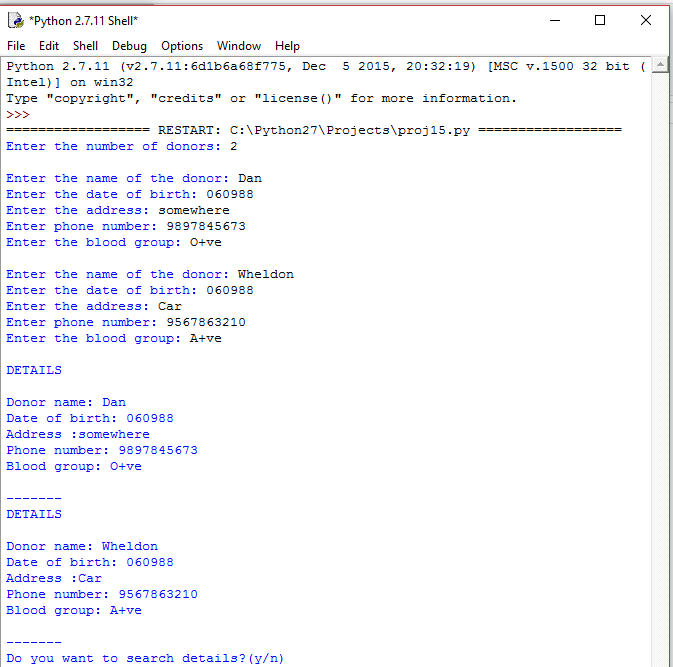
ans='y'

while ans=='y':

ans=raw\_input("Do you want to search details?(y/n) ")

if ans=='y':

Donor.Search()

****

**Output:**

16.Write a program in Python to count and display the number of lines not starting with alphabet 'A' present in a text file “STORY.TXT” Example: If the file “STORY.TXT” contains the following lines,

The rose is red.

A girl is playing there.

There is a playground.

An aero plane is in the sky.

Numbers are not allowed in the password

The function should display the output as 3.

**Code:**

#File will have to be created on desktop

f=open('story.txt','r')

count=0

for i in f.readlines():

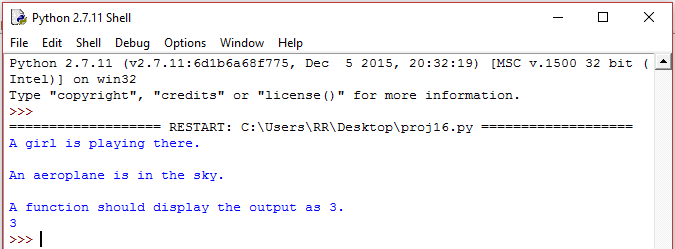
if i[0]=='A':

print i

count=count+1

print count

**Output:**

****

17.Write a menu driven program to:

a) Create a text file

b) Count number of vowels, digits and words

c)Create another file using the original which will contain the text after replacing all the blanks spaces with #.

**Code:**

while True:

ch=int(raw\_input("Please enter your option:"+'\n'+"1. Create a text file"+'\n'+'2.Count number of vowels, digits and words'+'\n'+'3.Replace spaces with #'+'\n'+'--- '))

if ch==1:

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

user=open(name,'w+')

l=[]

no=int(raw\_input("Enter the number of lines in the file: "))

for x in range (no):

string=raw\_input("Enter line number "+str(x+1) +'\n')

l.append(string)

l.append('\n')

user.writelines(l)

user.close()

vowels=['a','e','i','o','u']

if ch==2:

vcount=0

dcount=0

wcount=0

count=0

user=open(name,'r')

rd=user.read()

print rd

for x in rd:

if x in vowels:

vcount+=1

elif x.isdigit():

dcount+=1

else:

if x=='\n' or x==' ':

count+=1

wcount=count

user.close()

print "The number of vowels are: ",vcount

print "The number of digits are: ",dcount

print "The number of words are: ", wcount

if ch==3:

user=open(name,'a+')

rd=user.readlines()

li3=[]

for x in rd:

li=list(x)

print

for y in range(0,len(li)):

if li[y]==' ':

li[y]='#'

st=''

for a in li:

st+=str(a)

li3.append(st)

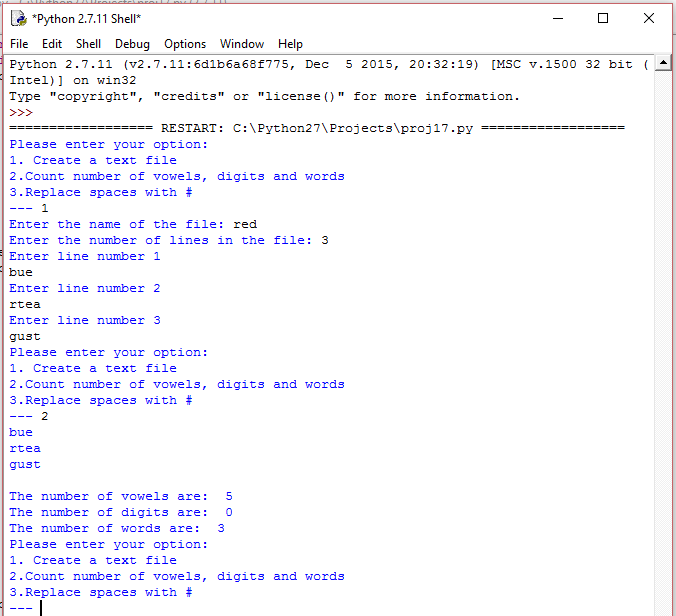
user.close()

user=open(name,'w+')

user.writelines(li3)

user.close()

**Output:**

****

18.Write a program that stores the decimal numbers and equivalent roman numbers in the form of a dictionary in a file. Next, read the file, load the dictionary and ask the user to enter a numeral and print its equivalent roman number.

**Code:**

import pickle

roman={1000:"M",900:"CM",500:"D",400:"CD",100:"C",90:"XC",50:"L",40:"XL",10:"X", 9:"IX",5:"V", 4:"IV",1:"I"}

file1=open("RomanNum.log","wb")

pickle.dump(roman,file1)

file1.close()

while True:

n=int(raw\_input("Enter a number (Enter -1 to exit) : "))

s=list(str(n))

x=list()

p=len(s)

if p>4:

print "Sorry, you can enter a maximum of 4 digits. Enter again"

continue

if n==-1:

break

if n>3999:

print "Sorry you cannot represent any number greater than 3999 with roman numberals. Enter another number"

continue

x.append(roman[1000]\*int(s[-4]))

if n/1000!=0:

x.append(roman[1000]\*int(s[0]))

if n/100!=0:

if s[-3]==9:

x.append(roman[900])

elif int(s[-3])>5 and int(s[-3])<9:

x.append(roman[500]+roman[100]\*(int(s[-3])-5))

elif int(s[-3])==5:

x.append(roman[500])

elif int(s[-3])==4:

x.append(roman[400])

elif int(s[-3])<4:

x.append(roman[100]\*(int(s[-3])))

if n/10!=0:

if s[-2]=="9":

x.append(roman[90])

elif int(s[-2])>5 and int(s[-2])<9:

x.append(roman[50]+roman[10]\*(int(s[-2])-5))

elif s[-2]=="5":

x.append(roman[50])

elif s[-2]=="4":

x.append(roman[40])

elif int(s[-2])<4:

x.append(roman[10]\*(int(s[-2])))

if s[-1]=="9":

x.append(roman[9])

elif int(s[-1])>5 and int(s[-1])<9:

x.append(roman[5]+roman[1]\*(int(s[-1])-5))

elif s[-1]=="5":

x.append(roman[5])

elif s[-1]=="4":

x.append(roman[4])

elif int(s[-1])<4:

x.append(roman[1]\*(int(s[-1])))

print "Roman numeral of",n,"is: ",

c=""

for i in x:

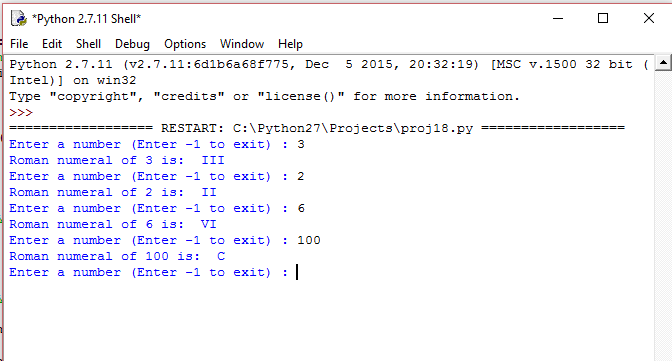
if i=="" or i==" ":

continue

c+=i

print c

**Output:**

****

19.Write a generator program in Python to find:

a) Cube of a number

b)Fibonacci Series

c)2^n Series

**Code:**

def fibonacci(max):

a,b=0,1

while a<max:

yield a

a,b=b,a+b

n=int(raw\_input("Enter till where"))

for i in fibonacci(n):

print i,

def cube(max):

a=0

while a<max:

yield a\*\*3

a=a+1

n=int(raw\_input("Enter till where"))

for i in cube(n):

print i,

def series(max):

a=0

while a<max:

yield 2\*\*a

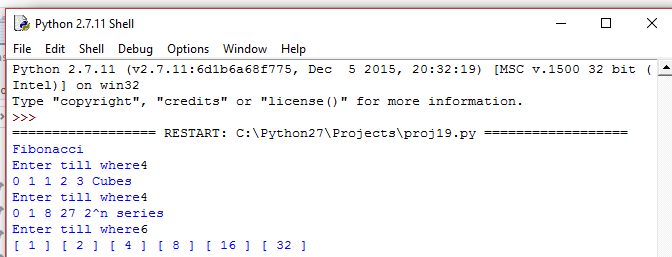
a=a+1

n=int(raw\_input("Enter till where"))

for i in series(n):

print "[",i,"]",

**Output:**

****

20.Write a function to read a Time class object storing hours and minutes. Raise a user-defined error if the values other than 0.23 is entered for hours and other than 0.59 is entered for minutes.

**Code:**

class Time\_error(Exception):

pass

hour=Time\_error("Hours can only take values from 0 to 23.")

minute=Time\_error("Minutes can only take values from 0 to 59.")

class Time:

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

self.hour=0

self.min=0

def get(self):

while True:

try:

self.hour=int(raw\_input("Enter hours: "))

if self.hour >23:

raise hour

self.min=int(raw\_input("Enter minutes: "))

if self.min>59:

raise minute

break

except Time\_error,e:

print e.message,"Enter again"

continue

def display(self):

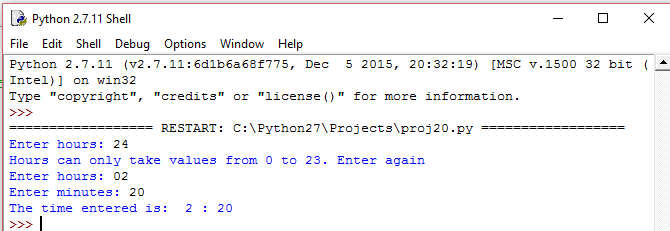
print "The time entered is: ",self.hour,":",self.min

x=Time()

x.get()

x.display()

**Output:**

****