**PYTHON APPLICATION PROGRAMMING**

**LAB PROGRAMS**

**NAME: CHINNI BHARGAVI S**

**USN:1PI11IS034**

**1:PROGRAM TO DISPLAY IDENTITY MATRIX OF GIVEN SIZE:**

n=input("enter matrix size\n")

a=[1,0,0]

for i in range(n):

for j in range(n):

a.append(0)

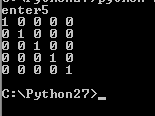
print(a[j]),

print

a[i]=0

a[i+1]=1

**OUTPUT:**



**2:PROGRAM TO FIND THE BIGGEST OF TWO:**

a=int(input("Enter a "))

b=int(input("Enter b "))

def dec(func):

def f(\*args):

print(".....................")

func(\*args)

print(".....................")

return f

@dec

def compare(a,b):

a=int(a)

b=int(b)

if a<b:

print(a,"is less than",b)

elif a>b:

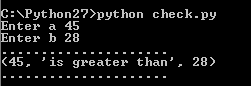
print(a,"is greater than",b)

else:

print(a,"equals",b)

compare(a,b)

**OUTPUT:**



**3:PROGRAM TO CHECK FOR VALID DATE:**

w="22-4-2008"

print(w)

a='11-22-2012'

print(a)

b=a.split('-')

print(b)

c=int(b[0])

d=int(b[1])

e=int(b[2])

if(c<=31 and e<=2013):

print("valid date")

if(e%4==0):

print("leap year")

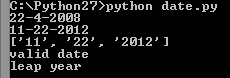
else:

print("not a leap year")

else:

print("invalid date")

**OUTPUT:**



**4:PROGRAM TO FIND LEAP YEAR:**

w="22-4-2008"

a=raw\_input("enter valid date of the format dd-mm-yyy \n")

b=a.split('-')

print(b)

c=int(b[0])

d=int(b[1])

e=int(b[2])

if (c<=31 and e<=2013) :

if (e%4!=0 and d==2 and c==29) :

print("not a valid date")

elif (e%4==0) :

print("leap year and valid date")

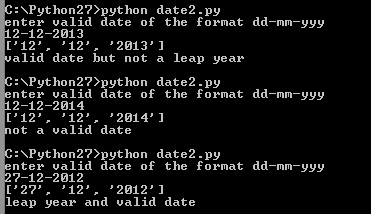
else:

print("valid date but not a leap year")

else:

print("not a valid date")

**OUTPUT:**



**5:PROGRAM TO DO STACK OPERATIONS:**

class mystack:

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

self.st = []

def push(self,p):

self.st.insert(0,p)

def pop(self):

try:

self.st.pop(0)

except:

print("Stack is empty")

def display(self):

print("Array elements")

print(self.st)

c1=mystack()

while(True):

ch=int(input("1.Push to stack\n2.Delete from stack\n3.Display elements\n4.exit\n"))

if (ch==1):

a=input("Enter element")

c1.push(a)

if(ch==2):

c1.pop()

if(ch==3):

c1.display()

if(ch==4):

print("Exiting")

exit()

**OUTPUT:**



**6:PROGRAM TO FIND AREA,VOLUME AND RADIUS:**

import MyMath

t=1

while(t==1):

choice=int(input("Enter 1 for computing area \nEnter 2 for computing volume \nEnter 3 for computing radius \nEnter any other value to terminate\n"))

if(choice==1):

r1=float(input("Enter radius to calculate area:\n"))

MyMath.area(r1)

elif(choice==2):

r2=float(input("Enter radius to calculate volume:\n"))

MyMath.volume(r2)

elif(choice==3):

v1=float(input("Enter volume to calculate radius:\n"))

MyMath.radius(v1)

else:

t=0

print("Thanks for trying\n")

module MyMath.py:

import math

def area(r):

a=4\*math.pi\*r\*r

print ("Area", a)

def volume(r):

v=4/3\*math.pi\*r\*r\*r;

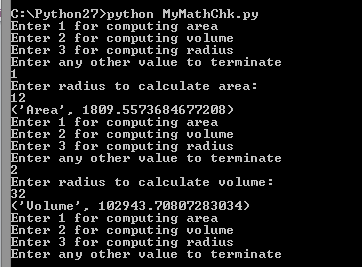
print ("Volume",v)

def radius(v):

r=math.pow((v\*3/(4\*math.pi)),1/3)

print ("Radius",r)

**OUTPUT:**



**7:PROGRAM TO FIND ODD PALINDROME NUMBERS IN A GIVEN RANGE:**

def reverse(n):

sum=0

while n:

sum=(sum\*10)+n%10

n=(int)(n/10)

return sum

def oddpal(n):

a=reverse(n)

if(a==n and n % 2==1):

return 1

else:

return 0

a=int(input("Enter the starting number"))

b=int(input("Enter the ending number"))

list1=[]

m=0

print("Displaying odd palindromes")

for i in range(a,(int)(b+1)):

m=oddpal(i)

if(m==1):

list1.append(i)

print(list1)

list2=[]

print("Displaying odd numbers")

for i in range(a,(int)(b+1)):

if(i%2==1):

list2.append(i)

print(list2)

print("Displaying only palindrome numbers")

list3=[]

for i in range(a,(int)(b+1)):

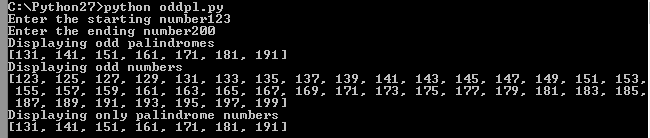
a=reverse(i)

if(i==a):

list3.append(i)

print(list3)

**OUTPUT:**



**8:PROGRAM TO FIND PRIME NUMBERS BETWEEN GIVEN RANGE:**

import math

def prime(a):

for i in range(2,(int)(math.sqrt(a)+1)):

if ((int)(a%i==0)):

return 0

return 1

a=int(input("Enter the starting number"))

b=int(input("Enter the ending number"))

m=0

print("The list is:")

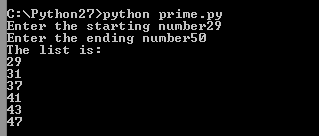
for i in range(a,(b+1)):

m=prime(i)

if m==1:

print(i)

**OUTPUT:**



**9:PROGRAM TO SEPARATE TAGS USING RE IN THE GIVEN PROGRAM:**

import re

p1='[A-Z 0-9]'

p='<.+?>'

s='<text> <sub>PAP</sub> <abc> </abc> <date> <dd>30</dd> <mm>9</mm> <yy>2013</yy> </date> </text>'

l=s.split(' ')

for i in l:

l1=re.split(p1,i)

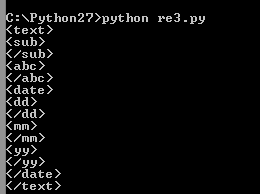
for j in l1:

a=re.search(p,j)

if a:

print(a.group())

**OUTPUT:**



**10:PROGRAM TO FIND THE SUM OF THE FOLLOWING FORM:**

m=input("enter number:\n")

h=range(1,m+1)

b='='

for i in range(m):

a=''

for j in range(i+1):

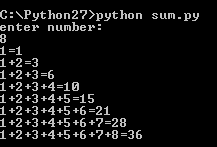
s=str(h[j])

a=a+s

sum=(str.join("+",a))

print(sum+b+str(eval(sum)))

**OUTPUT:**



**11:PROGRAMS ON REGULAR EXPRESSIONS:**

import re

sub='dear'

pat='[aeiou]'

m=re.search(pat,sub)

if m:

print("matched :",m.group())

def mymatch(subjects,pat):

print("pattern:",pat)

for sub in subjects:

m=re.search(pat,sub)

if m:

print(sub, ":",m.group())

print('-'\*40)

subjects=['0','9','5','.','-']

pat='[0-9]'

mymatch(subjects,pat)

subjects=['ab\*d','abcd','abbbbd']

pat='ab\*d'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='c.t'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='c\.t'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='c[:]t'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='cat'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='c[a]t'

mymatch(subjects,pat)

subjects=['cat','caat','at']

pat='[cat]'

mymatch(subjects,pat)

subjects=['ab','hjghaaabaafg']

pat='a.\*?b'

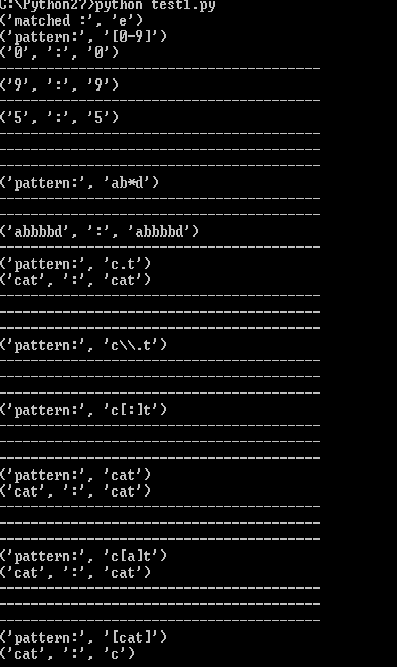
mymatch(subjects,pat)

subjects=['gfhghab','hjghaaabfdgdgbbb']

pat='a.\*b'

mymatch(subjects,pat)

**OUTPUT:**



**12:PROGRAM TO FIND UNION ,INTERSECTION OF GIVEN LIST:**

n1=input("enter list 1 range\n")

i=1

j=1

a=[]

b=[]

while(i<=n1):

num=input("numbr:")

a.append(num)

i+=1

print "the list 1 :"

print a

n2=input("enter list 2 range\n")

while(j<=n2):

num=input("numbr:")

b.append(num)

j+=1

print "the list 2 :"

print b

def intersect(a, b):

return list(set(a) & set(b))

print("the two lists are\n")

print(a)

print("\n")

print(b)

print("\n the intersected elements are")

print intersect(a, b)

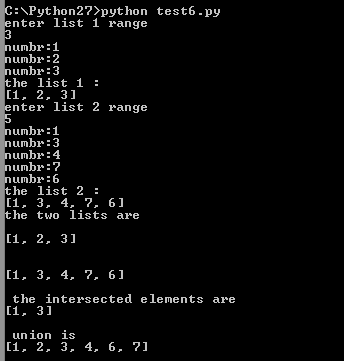
def union(a,b):

return list(set(a) | set(b))

print("\n union is ")

print union(a,b)

**OUTPUT:**



**13:PROGRAM TO FIND BIGGEST FILE IN THE DIRECTORY:**

import glob

import os

a=list(filter(os.path.isfile,glob.glob("\*")))

print("the files in the directory are:\n");

print(a)

i=0

j=1

b=[]

for i in range(0,5):

big=a[i]

for j in range(len(a[i])):

if(os.path.getsize(a[j])>big):

big=a[j]

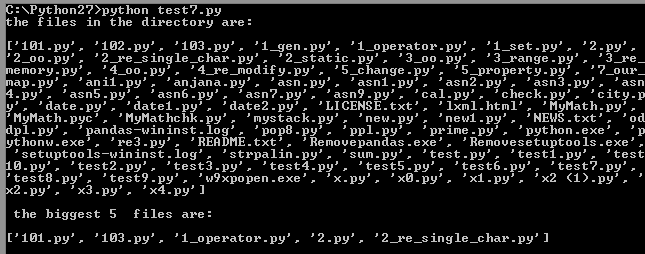
b.append(big)

a.remove(big)

print("\n the biggest 5 files are:\n")

print(b)

**OUTPUT:**



**14:PROGRAM TO GENERATE THE FOLLOWING PASCAL TRIANGLE:**

n=input("enter the value of n")

for i in range(n):

value=1

plist=[value]

for j in range(i):

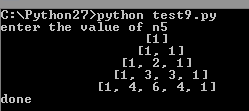
value=value\*(i-j)\*1/(j+1)

plist.append(int(value))

print("{:^40s}".format(plist))

print ("done")

**OUTPUT:**



**15:PROGRAM TO COUNT THE OCCURRENCE OF NUMBERS IN A LIST:**

from collections import Counter

a=[]

n=input("Enter the range of the list")

i=1

while (i<=n):

num=input("Number : ")

a.append(num)

i+=1

print "The list :"

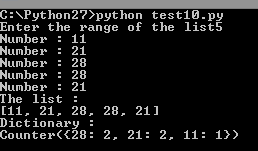
print a

print "Dictionary :"

c = Counter(a)

print c

**OUTPUT:**



**16:PROGRAM TO FIND THE BIGGEST FILE IN THE DIRECTORY:**

import glob

import os

l=list(filter(os.path.isfile,

glob.glob("\*")))

print(l)

n=[]

for i in range(0,5):

big=l[0]

for j in l:

if os.path.getsize(big)<os.path.getsize(j):

big=j

n.append(big)

l.remove(big)

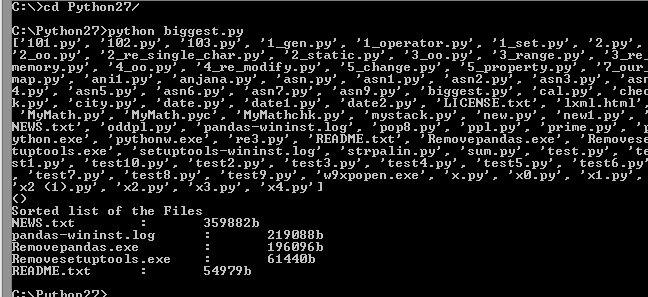
print()

print("Sorted list of the Files")

for i in n:

print(i+"\t:\t"+str(os.path.getsize(i))+"b")

**OUTPUT:**



**16: PROGRAM TO CHECK WHETHER GIVEN CHARACTER OCCURS AGAIN IN THE STRING.**

import re

sub=input("Enter the string")

pat=input("Enter character")

m=re.findall(pat,sub)

if len(m)>=2:

print("Character occurs more then once")

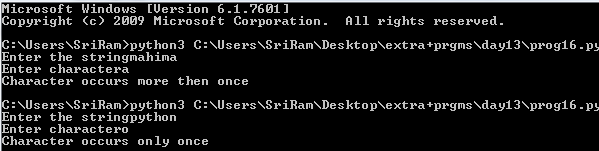
elif len(m)==1:

print("Character occurs only once")

else:

print("Character not found")

**OUTPUT:**



**17:PROGRAM TO FIND # OF OCCURENCES OF KEYWORDS IF DO THEN ENDDO ENDIF.**

import re

p1="IF|DO|ENDDO|THEN|ENDIF"

s="IF x<8 IF ENDDO x square ENDDO THEN x+8 ENDIF"

l=s.split(" ")

for i in l:

a=re.search(p1,i)

if(a):

print(a.group())

import re

regex=(r'IF|DO|ENDDO|THEN|ENDIF')

rc=re.compile(regex)

dict={}

dict['IF']=0

dict['DO']=0

dict['THEN']=0

dict['ENDDO']=0

dict['ENDIF']=0

string=''' IF x<8

IF

ENDDO

x square

ENDDO

THEN x+8

ENDIF'''

for i in rc.findall(string):

if i=='IF':

a=dict[i]

a+=1

dict[i]=a

elif i=='DO':

a=dict[i]

a+=1

dict[i]=a

elif i=='THEN':

a=dict[i]

a+=1

dict[i]=a

elif i=='ENDDO':

a=dict[i]

a+=1

dict[i]=a

elif i=='ENDIF':

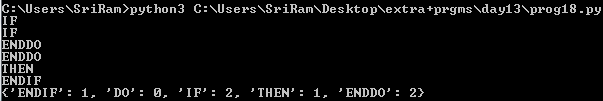
a=dict[i]

a+=1

dict[i]=a

print(dict)

**OUTPUT:**



**18:PROGRAM TO TOKENIZE THE FOLLOWING AS FOLLOWS.**

a---b->(a,ID) (--,op) (-,op) (b,ID)

a<<b->(a,ID) (<<,op) (b,ID)

a<b-c->(a, ID) (<,op) (b,ID) (-,op) (c,ID)

import re

r1=(r'[a-z]---[a-z]')

r2=(r'[a-z]<<[a-z]')

r3=(r'[a-z]<[a-z]-[a-z]')

string='''a---b\na<<<b\na<b-c'''

print(string)

rc1=re.compile(r1)

rc2=re.compile(r2)

rc3=re.compile(r3)

print("a--b")

for i in rc1.findall(string):

print(i[0],'ID')

print(i[1]+i[2],'OP')

print(i[3],'OP')

print(i[4],'ID')

print("a<<b")

for i in rc2.findall(string):

print(i[0],'ID')

print(i[1]+i[2],'OP')

print(i[3],'ID')

print("a<b-c")

for i in rc3.findall(string):

print(i[0],'ID')

print(i[1],'OP')

print(i[2],'ID')

print(i[3],'OP')

print(i[4],'ID')

**OUTPUT:**



**19:PROGRAM WHICH GETS NUMBER OF STRINGS USING VARIABLE NUMBER OF ARGUMENTS & RETURNS A LIST OF STRINGS WHICH ARE PALINDROMES.**

def reversenumber(n,partial=0):

if n==0:

return partial

return reversenumber(n/10,partial\*10+n%10)

trial=int(input("Enter the number"))

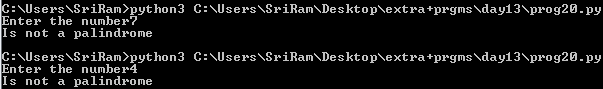
if reversenumber(trial)==trial:

print("Its a palindrome!!")

else:

print("Is not a palindrome")

**OUTPUT:**



**20: PROGRAM TO FIND THE FIVE BIGGEST FILES IN THE CURRENT DIRECTORY STORE THE RESULT IN A LIST AND SORT THE LIST USING:**

MAP, FILTER & GLOB MODULES

import os

import glob

os.chdir("/New folder")

filedict=dict()

for files in glob.glob("\*"):

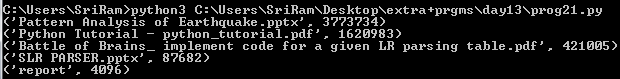
filedict[files]=os.path.getsize(files)

sf=(sorted(filedict.items(),key=lambda y:y[1],reverse=True))

for i in range(5):

print(sf[i])

**OUTPUT:**



**21:PROGRAM TO SUM THE ELEMENTS OF AN ARRAY USING THREADS.**

import threading

a=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

j=0

k=4

sum1=0

class mythread(threading.Thread):

def \_\_init\_\_(self,low,high):

threading.Thread.\_\_init\_\_(self)

self.low=low

self.high=high

self.total=0

def run(self):

for i in range(self.low,self.high):

self.total+=a[i]

for i in range(4):

thread=mythread(j,k)

thread.start()

thread.join()

sum1+=thread.total

print("Thread",i,":",thread.total)

j=k

k=k+4

print("sum:",sum1)

**OUTPUT:**



**22:PROGRAM TO ADD,DELETE CITIES :**

class city:

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

self.places = []

self.name = name

def add(self,p):

self.places.append(p)

def remove(self,r):

try:

self.places.remove(r)

except:

print("Place not present")

def display(self):

print("Places to visit")

print(self.places)

print ("City & places of visit")

c=input("Enter city name :")

c1=city(c)

while(True):

ch=int(input("1.Add a place to visit\n2.Delete a place of visit\n3.Display places to visit\n4.exit\n"))

if (ch==1):

a=input("Enter place ")

c1.add(a)

if(ch==2):

d=input("Enter place ")

c1.remove(d)

if(ch==3):

c1.display()

if(ch==4):

print("Exiting")

exit()

**23:PROGRAM TO FIND PALINDROMES:**

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

li=[]

for i in range(n):

li.append(input("Enter String "))

def pal(li):

l=[]co

for i in li:

if i==i[::-1]:

l.append(i)

return l

l=pal(li)

print(l)

**24:TCP SOCKET PROGRAMMING:**

client program

#!/usr/bin/env python  
import socket

TCP\_IP = "192.168.13.22"  
TCP\_PORT = 5005  
BUFFER\_SIZE = 1024  
MESSAGE = "Hello, World!"  
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
s.connect((TCP\_IP, TCP\_PORT))  
s.send(MESSAGE)  
data = s.recv(BUFFER\_SIZE)  
s.close()  
print "received data:", data

server program  
#!/usr/bin/env python  
import socket  
TCP\_IP = '192.168.13.22'  
TCP\_PORT = 5005  
BUFFER\_SIZE = 20  # Normally 1024, but we want fast response   
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
s.bind((TCP\_IP, TCP\_PORT))  
s.listen(1)  
conn, addr = s.accept()  
print 'Connection address:', addr  
while 1:  
        data = conn.recv(BUFFER\_SIZE)  
        if not data: break  
        print "received data:", data  
        conn.send(data)  # echo  
conn.close()



**25:DATABASE PROGRAM TO INSERT INTO GIVEN DATABASE:**

import pymysql

con=pymysql.connect()

cur=con.cursor()

name=input("Enter name : ")

dob=input("Enter dob(yyyy,mm,dd) : ")

cur.execute('use test')

cur.execute('''insert into details values (%s,%s)''',(name,dob))

cur.execute('select \* from details')

rows=cur.fetchall()

for row in rows:

print(row)

con.commit()

cur.close()

con.close()

**26:PROGRAM TO DISPLAY THE DATA FROM THE DATABASE:**

import pymysql

con=pymysql.connect()

cur=con.cursor()

cur.execute('use test')

cur.execute('select name,floor(datediff(curdate(),dob)/365) as age from details')

print('name\tage')

rows=cur.fetchall()

for row in rows:

for val in row:

print(val,end='\t')

print()

con.commit()

cur.close()

con.close()

**27:PROGRAM TO DISPLAY DOB WHIH FALLS IN THE GIVEN MONTH:**

import pymysql

con=pymysql.connect()

cur=con.cursor()

val=input("Enter a month to check : ")

cur.execute('use test')

cur.execute('select name from details where month(dob)= %s'%val)

print('name')

rows=cur.fetchall()

for row in rows:

for val in row:

print(val,end='\t')

print()

con.commit()

cur.close()

con.close()

**28:SIMPLE CALCULATOR PROGRAM USING LEX AND YACC:**

# -----------------------------------------------------------------------------

# calc.py

#

# A simple calculator with variables. This is from O'Reilly's

# "Lex and Yacc", p. 63.

# -----------------------------------------------------------------------------

import sys

sys.path.insert(0,"../..")

tokens = (

'NAME','NUMBER',

)

literals = ['=','+','-','\*','/', '(',')']

# Tokens

t\_NAME = r'[a-zA-Z\_][a-zA-Z0-9\_]\*'

def t\_NUMBER(t):

r'\d+'

try:

t.value = int(t.value)

except ValueError:

print ("Integer value too large", t.value)

t.value = 0

return t

t\_ignore = " \t"

def t\_newline(t):

r'\n+'

t.lexer.lineno += t.value.count("\n")

def t\_error(t):

print( "Illegal character '%s'" % t.value[0])

t.lexer.skip(1)

# Build the lexer

import ply.lex as lex

lex.lex()

# Parsing rules

precedence = (

('left','+','-'),

('left','\*','/'),

('right','UMINUS'),

)

# dictionary of names

names = { }

def p\_statement\_assign(p):

'statement : NAME "=" expression'

names[p[1]] = p[3]

def p\_statement\_expr(p):

'statement : expression'

print(p[1])

def p\_expression\_binop(p):

'''expression : expression '+' expression

| expression '-' expression

| expression '\*' expression

| expression '/' expression'''

if p[2] == '+' : p[0] = p[1] + p[3]

elif p[2] == '-': p[0] = p[1] - p[3]

elif p[2] == '\*': p[0] = p[1] \* p[3]

elif p[2] == '/': p[0] = p[1] / p[3]

def p\_expression\_uminus(p):

"expression : '-' expression %prec UMINUS"

p[0] = -p[2]

def p\_expression\_group(p):

"expression : '(' expression ')'"

p[0] = p[2]

def p\_expression\_number(p):

"expression : NUMBER"

p[0] = p[1]

def p\_expression\_name(p):

"expression : NAME"

try:

p[0] = names[p[1]]

except LookupError:

print( "Undefined name '%s'" % p[1])

p[0] = 0

def p\_error(p):

print( "Syntax error at '%s'" % p.value)

import ply.yacc as yacc

yacc.yacc()

while 1:

try:

s =input("calc > ")

except EOFError:

break

if not s: continue

yacc.parse(s)

**----------------------------------------------------END--------------------------------------------------**