# control structure of python

a=int(input("a : "))

b=int(input("b : "))

if a>b:

print('a is greater')

elif b>a:

print('b is greater')

else:

print('equals')

# list tuple and dictionary

l = [1,223] # mutable

t = (1,233) #immutable

d={1:'one'} # pair of key and values

for i in range(5):

x = int(input('enter : '))

l.append(x)

print(l)

t=tuple(l) # new tuple

print(t)

for i in range(5):

k=int(input('key : '))

v=input('value : ')

d.update({k:v})

print(d)

# function scoping and recursion

n=int(input('Enter Number : '))

def fact (n):

if n<=1 :

return 1

else:

return n\*fact(n-1)

print(fact(n))

# Object Oriented Programming

class sachin():

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

self.rno = rno

def call(self,name):

print('name : ',name)

return(self.rno)

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

name = input('name : ')

s=sachin(rno)

print(s.call(name))

# Exception handling

ls = list(input('enter : '))

for i in ls:

try:

div=10/int(i)

print(div)

except Exception as e:

print(e)

#Armstrong

n=input('enter no :') #153

sum=0

for i in range(len(n)): # 3

x= int(n[i])\*\*3

sum = sum + x

if int(n)==sum:

print('Armstrong')

else:

print('not armstrong')

# factorial

n = int(input('enter no : '))

fact=1

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

fact=fact\*i

print(fact)

# prime no

n=int(input('enter no : '))

def prime(n):

for i in range(2,n):

if (n%i)==0:

return('not prime')

if n<=1:

return('not prime')

return('prime')

print(prime(n))

# lambda function

sqr =lambda x:x\*\*2

n = int(input('enter no : '))

x = sqr(n)

print(x)

# Calculator

a=int(input('a : '))

b=int(input('b : '))

while(True):

print('enter 1/2/3/4 or 0')

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

if ch==1:

print(a+b)

elif ch==2:

print(a-b)

elif ch==3:

print(a\*b)

elif ch==4:

print(a//b)

elif ch==0:

break

else:

print('wrong')

# Binary

def binary(ls,s):

l=0

h=len(ls)-1

while l<=h:

m=(l+h)//2

if ls[m]==s:

return 1

else:

if ls[m]<s:

l=m+1

else:

h=m-1

return 0

ls = [5,3,2,5,56,11]

ls.sort() # sorted list

print(ls)

s = int(input('search : '))

if binary(ls,s):

print('found')

else:

print('not found')