class triangle:

  def \_\_init\_\_(self,row):

    self.row=row

  def myfunc(tri):

    for i in range (1,row):

      for j in range (1,i+1):

        print(j, end= ' ')

      print("\r")

row= int(input("rows : "))

p=triangle(row)

p.myfunc()

class pyramid:

  def \_\_init\_\_(self,row):

    self.row=row

  def myfunc(pyr):

      for i in range(0,row):

        for j in range(0,row-i-1):

               print(end=" ")

        for j in range(0, i+1):

              print("\*", end=" ")

        print("\r")

row= int(input("rows : "))

p=pyramid(row)

p.myfunc()

class inverseTriangle:

  def \_\_init\_\_(self,row):

    self.row=row

  def myfunc(invtri):

      for i in range(0,row):

        for j in range(i,row):

               print(" ",end=" ")

        for j in range(0, i+1):

              print("\*", end=" ")

        print("\r")

row= int(input("rows : "))

p=inverseTriangle(row)

p.myfunc()

class decreasingTriangle:

  def \_\_init\_\_(self,row):

    self.row=row

  def myfunc(dectri):

    for i in range (0,row):

      for j in range (i,row):

        print(j, end= ' ')

      print("\r")

row= int(input("rows : "))

p=decreasingTriangle(row)

p.myfunc()

import random

class Environment(object):

    def \_\_init\_\_(self):

        self.locationcondition={'A' : '1' , 'B' : '1'}

#RANDOM CONDITION

        self.locationcondition['A']=random.randint(0,1)

        self.locationcondition['B']=random.randint(0,1)

class Sreflexagent(Environment):

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

        print(Environment.locationcondition)

        #place vacum at random location

        vacuumlocation=random.randint(0,1)

        #if vacuum at A

        if vacuumlocation==0:

            print("vacuum is randomly placed at locationn A")

            #and if location A is dirty

            if Environment.locationcondition['A']==1:

                print("Location A is dirty")

                #suck the dirt and mark it clean

                Environment.locationcondition['A']==0;

                print("Location A has been cleaned")

                #move to B

                print("moving to location B")

                #if B is dirty

                if Environment.locationcondition['B']==1:

                    print("Location b is dirty")

                #suck the dirt and mark it clean

                    Environment.locationcondition['B']==0;

                    print("Location B has been cleaned")

            else:

              #move to B

              print("moving to location B")

              #if B is dirty

              if Environment.locationcondition['B']==1:

                 print("Location B is dirty")

                 #suck the dirt and mark it clean

                 Environment.locationcondition['B']==0;

                 print("Location B has been cleaned")

        elif vacuumlocation==1:

          print("vacuum is randomly placed at locationn B")

          #and if location B is dirty

          if Environment.locationcondition['B']==1:

                print("Location B is dirty")

                #suck the dirt and mark it clean

                Environment.locationcondition['B']==0;

                print("Location B has been cleaned")

                #Move to A

                print("moving to location A")

                if Environment.locationcondition['A']==1:

                    print("Location A is dirty")

                    #suck the dirt and mark it clean

                    Environment.locationcondition['A']==0;

                    print("Location A has been cleaned")

        else:

                #move to A

                print("moving to location A")

                #if A is dirty

                if Environment.locationcondition['A']==1:

                    print("Location A is dirty")

                #suck the dirt and mark it clean

                    Environment.locationcondition['A']==0;

                    print("location A has been clened")

      #DONE CLEANING

theEnvironment=Environment()

thevacuum=Sreflexagent(theEnvironment)

x=int(input('Enter Number: '))

if x>1:

  for i in range(2,int(x/2)+1):

    if (x%i==0):

     print (x,'is not a prime number')

     break

  else:

   print (x,'is a prime number')

else:

 print (x,'is not a prime number')

list1=[5,9,2,7,18]

list2=[1,10,11,3,19]

list3=[]

for i in range(len(list1)):

  if (list1[i]%2==0):

    list3.append(list1[i])

for i in range(len(list2)):

  if (list2[i]%2!=0):

    list3.append(list2[i])

print('Unsorted list : ',list3)

for i in range(0,len(list3)-1):

    for j in range(len(list3)-1):

       if(list3[j]>list3[j+1]):

        temp = list3[j]

        list3[j] = list3[j+1]

        list3[j+1] = temp

print('sorted list : ',list3)

graph = {2: [0, 3], 0: [1,2], 1: [2], 3: [3]}

discovered=[]

queue=[]

def bfs(discovered,graph,node):

  discovered.append(node)

  queue.append(node)

  while queue:

    g=queue.pop(0)

    print(g)

    for child in graph[g]:

      if child not in discovered:

        discovered.append(child)

        queue.append(child)

print('BFS : ')

bfs(discovered,graph,2)