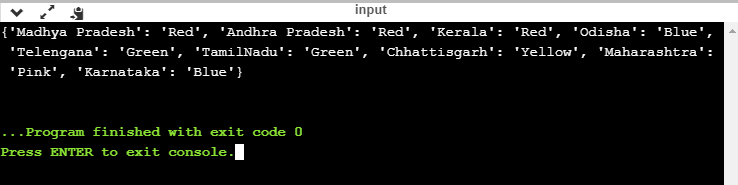
**INLAB**

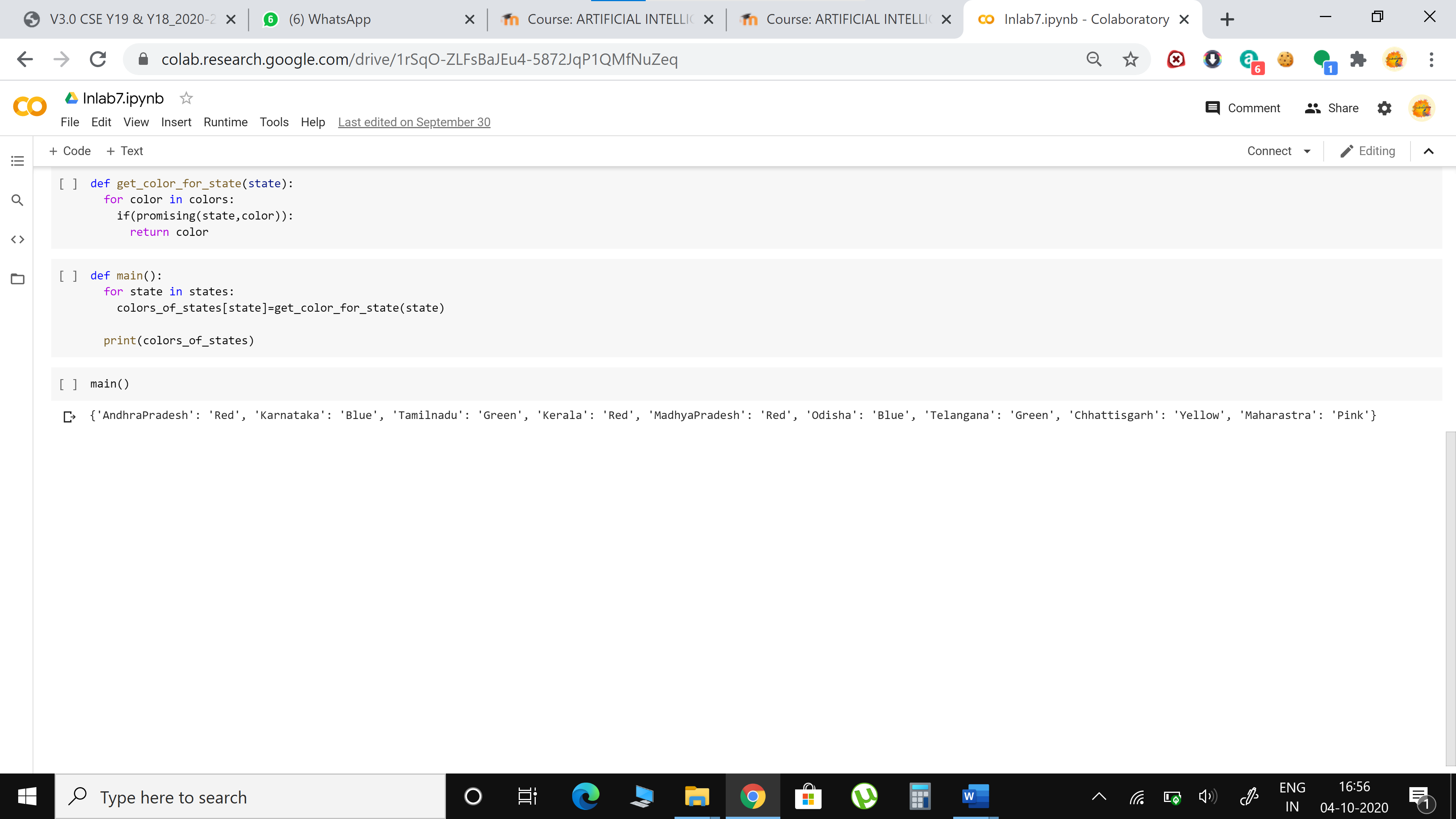
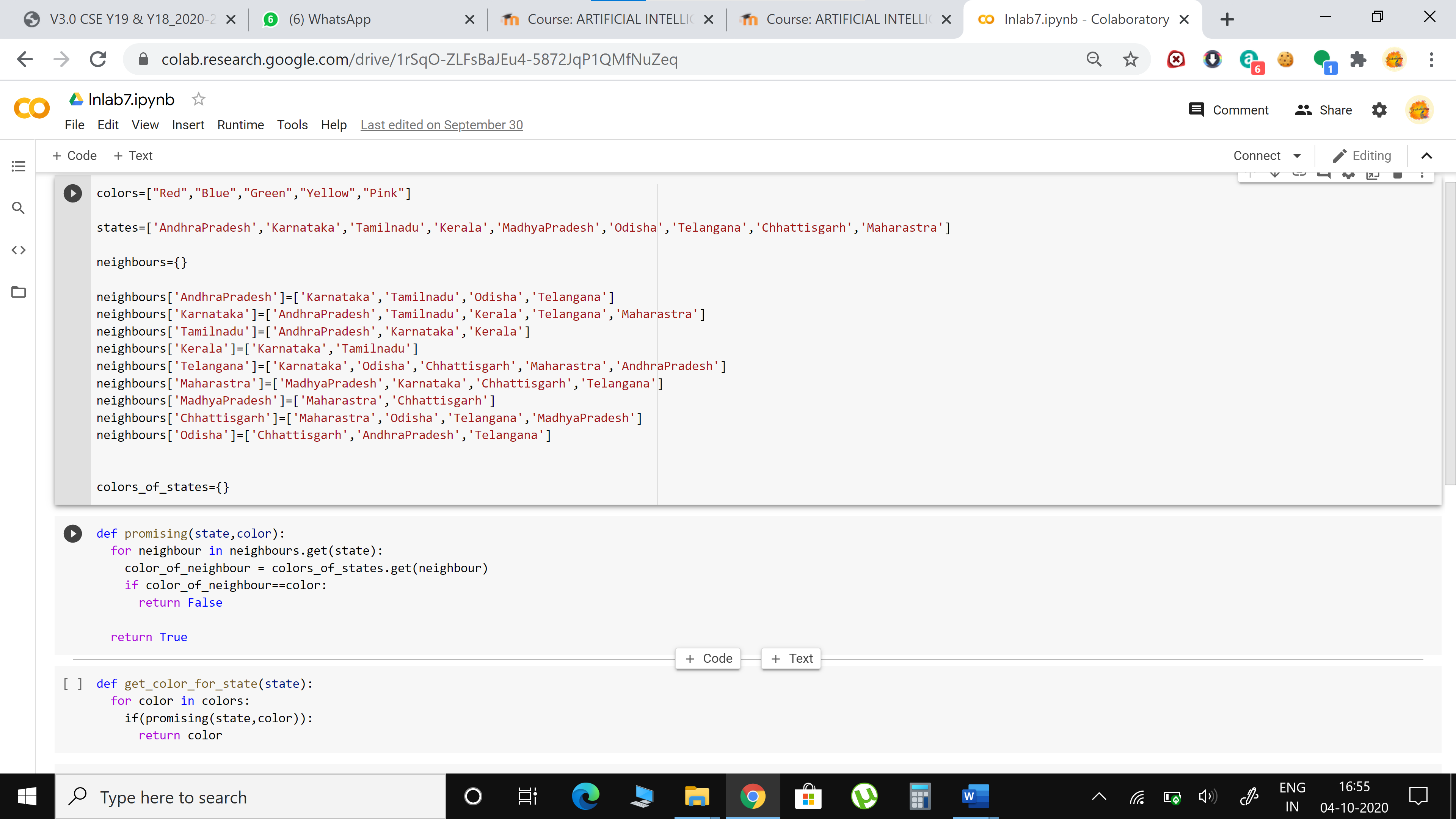
1. You have to color a map with different colors where no two neighboring regions can have the same color. Assume that you have 5 colors red, blue, green, yellow and pink.Write an efficient python code to color the regions in the following map.



Sample Output:



Code:



Code:

colors=["Red","Blue","Green","Yellow","Pink"]

states=['AndhraPradesh','Karnataka','Tamilnadu','Kerala','MadhyaPradesh','Odisha','Telangana','Chhattisgarh','Maharastra']

neighbours={}

neighbours['AndhraPradesh']=['Karnataka','Tamilnadu','Odisha','Telangana']

neighbours['Karnataka']=['AndhraPradesh','Tamilnadu','Kerala','Telangana','Maharastra']

neighbours['Tamilnadu']=['AndhraPradesh','Karnataka','Kerala']

neighbours['Kerala']=['Karnataka','Tamilnadu']

neighbours['Telangana']=['Karnataka','Odisha','Chhattisgarh','Maharastra','AndhraPradesh']

neighbours['Maharastra']=['MadhyaPradesh','Karnataka','Chhattisgarh','Telangana']

neighbours['MadhyaPradesh']=['Maharastra','Chhattisgarh']

neighbours['Chhattisgarh']=['Maharastra','Odisha','Telangana','MadhyaPradesh']

neighbours['Odisha']=['Chhattisgarh','AndhraPradesh','Telangana']

colors\_of\_states={}

def promising(state,color):

  for neighbour in neighbours.get(state):

    color\_of\_neighbour = colors\_of\_states.get(neighbour)

    if color\_of\_neighbour==color:

      return False

  return True

def get\_color\_for\_state(state):

  for color in colors:

    if(promising(state,color)):

      return color

def main():

  for state in states:

    colors\_of\_states[state]=get\_color\_for\_state(state)

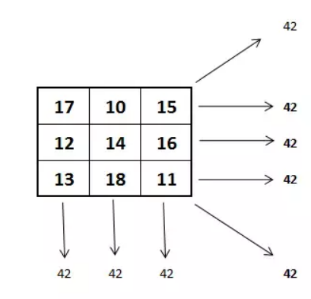
  print(colors\_of\_states)

main()

**POSTLAB**

1. Solve the following problem using Constraint Satisfaction Problems (CSP):

Test Case 1: Magic Square ([[10,11,12], [13, 14, 15], [16, 17, 18]])

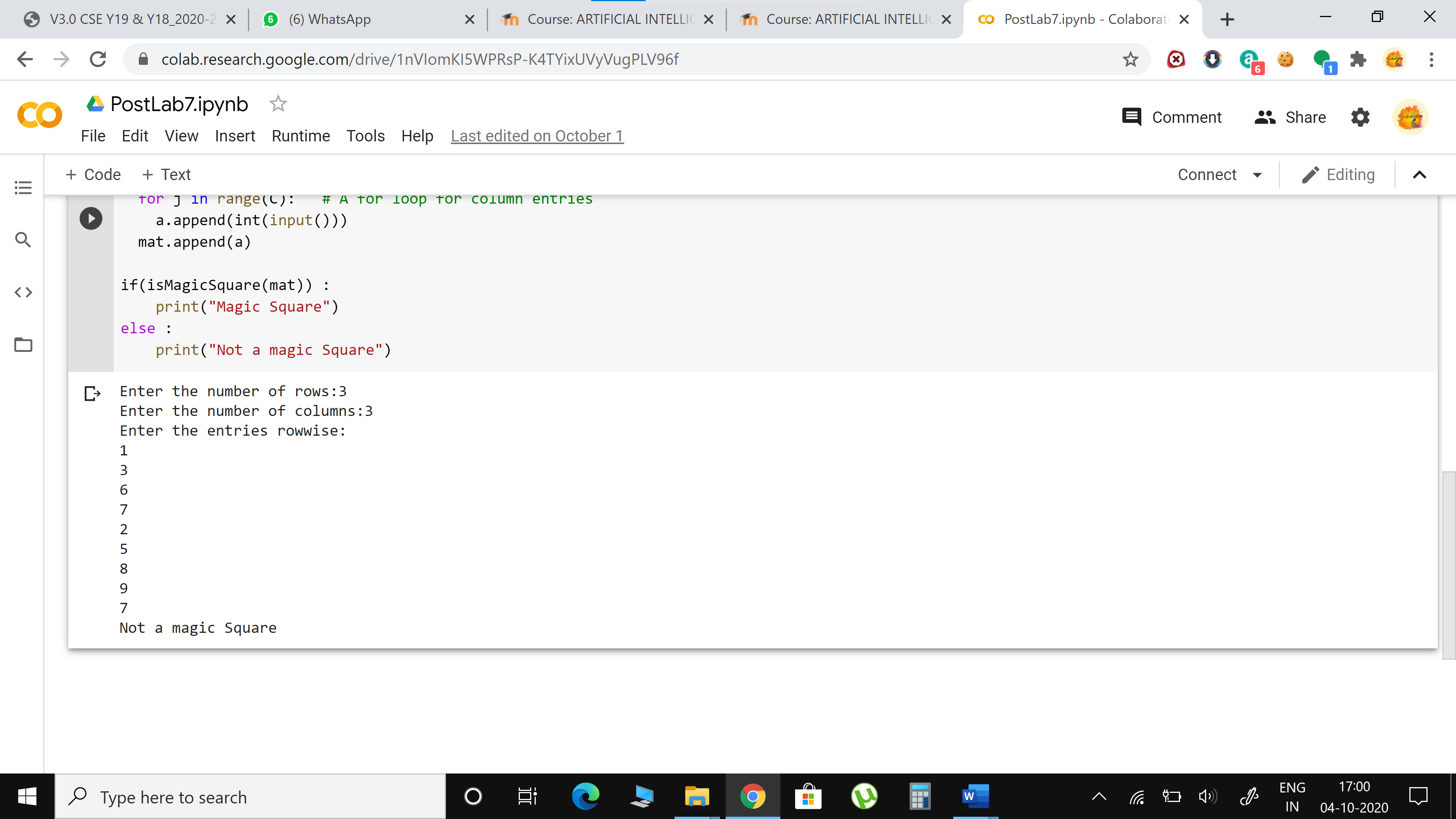
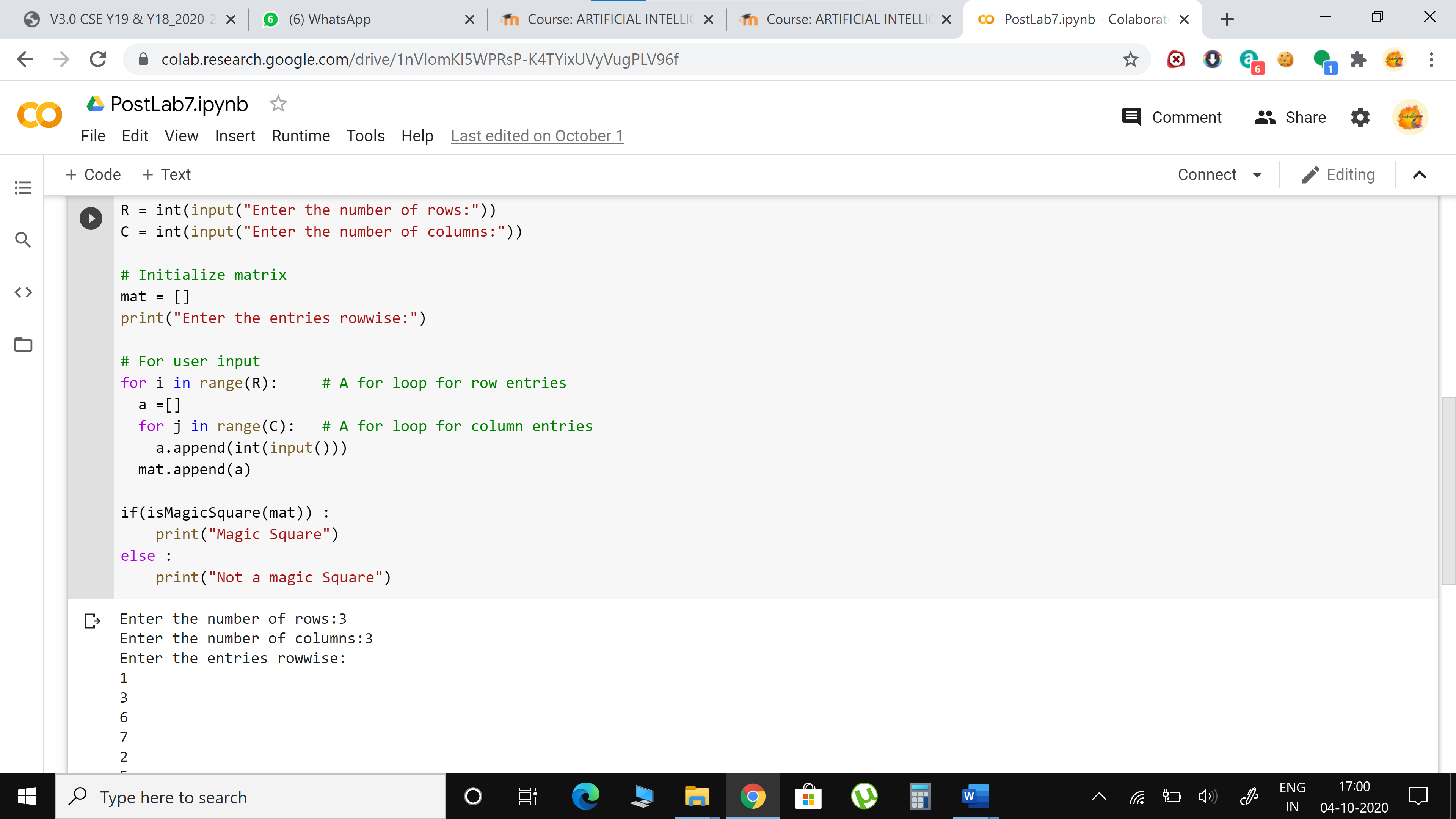
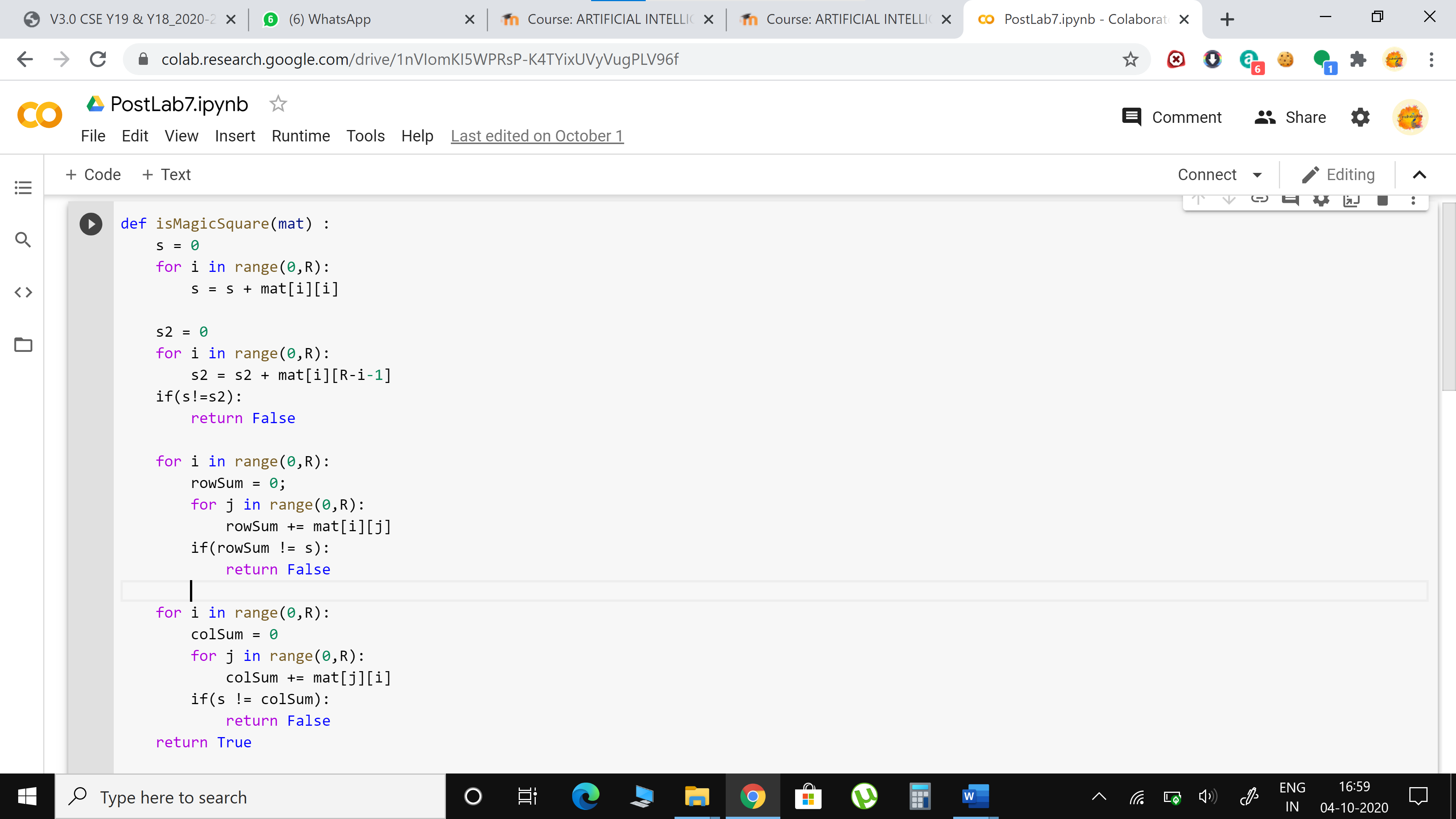
False, this is not a magic square. The numbers in the rows/columns/diagonals do not add up to the same value. Let’s try another list of lists.

Test Case 2: Magic Square ([[17,10,15],[12,14,16],[13,18,11]])

True

• Develop a python program that satisfies below operations.

Code:



Code:

def isMagicSquare(mat) :

    s = 0

    for i in range(0,R):

        s = s + mat[i][i]

    s2 = 0

    for i in range(0,R):

        s2 = s2 + mat[i][R-i-1]

    if(s!=s2):

        return False

    for i in range(0,R):

        rowSum = 0;

        for j in range(0,R):

            rowSum += mat[i][j]

        if(rowSum != s):

            return False

    for i in range(0,R):

        colSum = 0

        for j in range(0,R):

            colSum += mat[j][i]

        if(s != colSum):

            return False

    return True

R = int(input("Enter the number of rows:"))

C = int(input("Enter the number of columns:"))

# Initialize matrix

mat = []

print("Enter the entries rowwise:")

# For user input

for i in range(R):     # A for loop for row entries

  a =[]

  for j in range(C):   # A for loop for column entries

    a.append(int(input()))

  mat.append(a)

if(isMagicSquare(mat)) :

    print("Magic Square")

else :

    print("Not a magic Square")