# Python program to placing 8 chess queens on an 8x8 chessboard

import requests

url = ’<https://github.com/mypythoncode09/PYTHON_CHALLENGE/upload>’

x = requests.post(url,json={"qconfig":"<< global n

n = 8

def QSol(chessboard):

for i in range(n):

for j in range(n):

print chessboard[i][j],

print

def Safe(chessboard, row, col):

#Check this row on left side

for i in range(col):

if chessboard[row][i] == 1:

return False

#Check upper diagonal on left side

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if chessboard[i][j] == 1:

return False

#Check lower diagonal on left side

for i, j in zip(range(row, n, 1), range(col, -1, -1)):

if chessboard[i][j] == 1:

return False

return True

def NQSolve(chessboard, col):

# base case: If all queens are placed then return true

if col >= n:

return True

# Consider this column and try placing this queen in all rows one by one

for i in range(n):

if Safe(chessboard, i, col):

#Place this queen in board[i][col]

chessboard[i][col] = 1

#Recur to place rest of the queens

if NQSolve(chessboard, col + 1) == True:

return True

#If placing queen in board[i][col] doesn't lead to a solution, then queen from board[i][col]

chessboard[i][col] = 0

return False

#this function prints one of the feasible solutions

def SolveQ():

chessboard = [ [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0] ]

if NQSolve(chessboard, 0) == False:

print "Solution does not exist"

return False

QSol(chessboard)

return True

#A driver program to test above function

SolveQ()>>","userID":<<815018>>,"githubLink":"<<<https://github.com/mypythoncode09/PYTHON_CHALLENGE/upload>>>"})

print(x.text)