'''

Problem:

Given a cost matrix cost[][] and a position (m, n) in cost[][],

write a function that returns cost of minimum cost path to reach (m, n) from (0, 0).

Each cell of the matrix represents a cost to traverse through that cell.

Total cost of a path to reach (m, n) is sum of all the costs on that path

(including both source and destination).

You can only traverse down, right and diagonally lower cells from a given cell,

i.e., from a given cell (i, j), cells (i+1, j), (i, j+1) and (i+1, j+1) can be traversed.

Ways:

This problem is similar with edit distance,

considering there are 3 operations for a certain location.

A DP solution, using a matrix to solve

Questions:

what about at each location, the agent can traverse up?

Ref:

https://www.geeksforgeeks.org/min-cost-path-dp-6/

'''

import numpy as np

def min\_cost\_path(matr):

[rows, cols] = matr.shape

cost\_matr = np.zeros([rows, cols])

cost\_matr[0,0] = matr[0,0]

for irow in range(rows):

for jcol in range(cols):

if (irow == 0) and (jcol == 0):

continue

elif irow == 0:

cost\_matr[irow, jcol] = cost\_matr[irow][jcol-1] + matr[irow, jcol]

elif jcol == 0:

cost\_matr[irow, jcol] = cost\_matr[irow-1][jcol] + matr[irow, jcol]

else:

c\_down = cost\_matr [irow-1][jcol]

c\_right = cost\_matr [irow][jcol-1]

c\_diag = cost\_matr [irow-1][jcol-1]

cost\_matr[irow, jcol] = matr[irow, jcol] + min(c\_down, c\_right, c\_diag)

return cost\_matr, cost\_matr[rows-1, cols-1]

matr = np.asarray([[1,2,3],[4,8,2],[1,5,3]])

cost\_matr, min\_operation = min\_cost\_path(matr)

print(cost\_matr)

print(min\_operation)