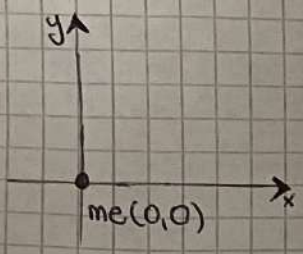
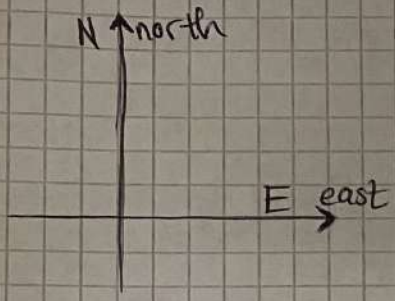
AnchorzUp Task



- goal is to reach point (X, Y)
- X steps east of current location
- Y steps north of current location
- move by taking steps which must lead east or north



→ moving east means to increase the X coord.

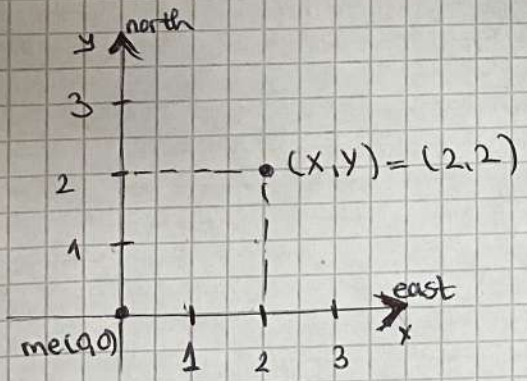
↑ moving north means to increase the Y coord.

- not allowed to take exactly 3 steps in a row, but less or more than 3 is totally good.

Example 1: Input: (2,2)

Output: number of valid paths: 6

routes for each path: EENN, ENEN
NEEN, ENNE
NENE, NNEE



Solution: Based on all the infos, we should create a 2D matrix, which stores the number of paths.

Warning: not allowed to take exactly 3 steps!!

each step must lead north or east.

⇒ Count all the valid ways of reaching the goal.

(X = 2, Y = 2)

We are at (x=0, y=0) and wanna reach (x=2, y=2)
How many ways to reach there??
Matrix to keep track of the number of ways to reach each point.
Size of matrix is determined by the maximum x and y.
x = 2, y = 2
matrix size is (x+1) by (y+1) = (2+1) by (2+1) = 3 by 3

so matrix is must be like

matrix = [[1] * (y+1) for _ in range (x+1)]

3 rows, 3 columns

	0	1	2
0	[1,	1,	1]
1	[1,	1,	1]
2	[1,	1,	1]

nested loop
```
for i in range (1, x+1)
    for j in range (1, y+1)
        matrix[i, j] = matrix[i-1][j] matrix[i][j-1]

for i in range (1, 2+1)
    for j in range (1, 2+1)
        matrix[i][j] = matrix[i-1][j] + matrix[i][j-1]

for i in range (1, 3)
    for j in range (1, 3)
```

Iteration 1 (i = 1, j = 1) = matrix[1, 1] = matrix[0][1] + matrix[1, 0]
= 1 + 1 = 2

updated matrix = $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Iteration 2 i = 1, j = 2 ... $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$

- After all iterations, the matrix is this:

	0	1	2
0	1	1	1
1	1	2	2
2	1	3	5

matrix [2][2] is 5, meaning there are 5 valid paths to reach the destination

= find_paths function, to find all the paths valid, which in our case are 5 valid paths.

```
if (x > 0)
    find_paths (x-1, y, path + "E")
```

checking if there are still steps that need to be taken east direction

find_paths (2, 2, "")

x=2, y=2

find paths (x-1, y, path + "E")

find paths (1, 2, path + E) = E , move east E 1x

x=1, y=2

find paths (0, 2, E) move east E ...

⇒ After all the steps, I find the steps

- And at the end, I tell the user to type x and y, which mustn't be negative numbers, so I give the number of routes and the routes themself

Example x=2, y=2 6 routes

Attached in repo, please find the printed results, in case you do not have the Python or stuff installed. //