Getting started: paired activity

# Reading .The board

Although there are lots of ways to set up a noughts and crosses board, you will be using a two-dimensional list. Initially, this will look like this:

| 1  2  3 | board = [[" ", " ", " "],  [" ", " ", " "],  [" ", " ", " "]] |
| --- | --- |

Each element in this two-dimensional list can be accessed using a variation of this code snippet:

| # r c  board[0][0] |
| --- |

The values in the square brackets are replaced with the row and column indexes. These are represented in the grid below:

| [0][0] | [0][1] | [0][2] |
| --- | --- | --- |
| [1][0] | [1][1] | [1][2] |
| [2][0] | [2][1] | [2][2] |

A procedure will then be used to display the board based on the data from the 2D list. The start code for a noughts and crosses board can be viewed below, or by accessing [this link](https://ncce.io/ks4-boardformat) (ncce.io/ks4-boardformat):

| def displayboard(board):  print(" ", board[0][0], "│", board[0][1], "│", board[0][2])  print(" ───┼───┼───") |
| --- |

# Task .The board

**Step 1**

Use the information on the first page to help you create a completed noughts and crosses board using the 2D list and the incomplete procedure.

**Step 2**

Call the procedure and see how the board is displayed on the screen. You might want to try different board layouts to personalise it further.

**Step 3**

Modify the board 2D list by adding in Os and Xs and check if your display board still looks like a workable board for your game.

An example output might be:



**Step 4**

Copy your code below and make sure that this is shared with each person in your pair.

|  |
| --- |

# Task .Checking for a win

**Step 1**

Work out all of the possible ways that you can win at noughts and crosses. Write the total number below:

| Answer: |  |
| --- | --- |

**Tip:** You might be able to work this out in your head but if not, get some scrap paper and keep drawing boards until you have drawn all of the possibilities and then count them.

**Step 2**

Using the grid below as a guide, write down all of the combinations for a noughts and crosses win. The **first one** has been done for you.

| [0][0] | [0][1] | [0][2] |
| --- | --- | --- |
| [1][0] | [1][1] | [1][2] |
| [2][0] | [2][1] | [2][2] |

| Answer: | 1. [0][0] [0][1] [0][2] |
| --- | --- |

**Step 3**

A function has been started that will check if the current 2D list contains a winning combination. Here is the start code:

| def check\_win(board, player):  won = False  if board[0][0] == player and board[0][1] == player and board[0][2] == player:  won = True |
| --- |

**Complete** this function using the combinations that you provided in step 2.

**Step 4**

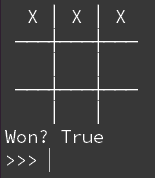
**Test** your function by editing your board so that X wins for each possible combination. For example, to test the first combination, set your board like this:

| 1  2  3 | board = [["X", "X", "X"],  [" ", " ", " "],  [" ", " ", " "]] |
| --- | --- |

**Tip:** You will need to create a variable for the player to ensure that this argument is passed to the check\_win function. Make sure that the player holds the value X or your program will not work.

**Tip:** To make sure that your code works, you will need to print won and see if it is True when a winning combination is used.

Here is an example of what the output for the first combination might look like:



**Step 5**

Copy your working code below and make sure that this is shared with each person in your pair.

|  |
| --- |