**Software Workshop I**

**Assignment 2**

**Marks available: 40**

**Date of assessment:**
**04/12/2023, Monday at 15h00**

**Set by: Jacqui Chetty**

**Background:**

This assignment is based on tic-tac-toe. However, for this assignment, there are different rules required to play the game. These are discussed below.

**Instructions:**

1. Create a new Python **project** called yourSurnameStudentNumber, for example, Smith123456
2. Create a **file** called board-game.py.
3. Copy the contents of the template given to you into this file and make use of the instructions below to complete the code for the game.
4. Do not put any code that gets user input.
5. Global variables are found at the top of the template and are as follows: rows, columns.
6. You may add more variables and / or functions, but you must not remove existing ones.
7. For this game, a valid tic-tac-toe board will always be displayed as an odd number of rows / columns, for example, 3 x 3; 5 x 5; 7 x 7; 9 x 9; etc. Assume that 1 X 1 is not allowed. The rows and columns must **always** have the same number. If rows = 3, then columns = 3.
8. To play tic-tac-toe, a player places either an 'X' or an 'O' on the board. Both are alphabetic.
9. Your solution must work for different board sizes. You should be able to change the global rows /columns

variables to contain other combinations and the game should still work.

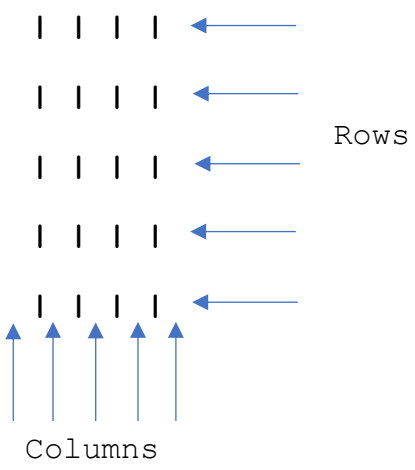
10. Please familiarise yourself with the template before starting.
11. The examples below show the output in the console. There is a full set of output examples that reflect the tests given as part of the template.
12. There is no need to develop any GUI.

**Submission instructions:**

Submit your work by zipping the **project** (for example, Smith123456 – see step 1 above) which includes your board-game.py file and upload the .zip file to Canvas.

**Now you are ready to start your assignment solution:**

1. **Go to the function generate_board() and create a tic-tac-toe board based on the following criteria. [4]**
    a. Make use of the global variables rows and columns located at the top of the template. The default values of rows = 5 and columns = 5 has been provided, i.e. a 5 x 5 board. Hint: your board is a list. The initial test cases are based on this board size.
    b. Make use of the board list given in the function and complete the necessary code required to generate a board of any valid row / column combination (see Rule 7 above). See the example below for what an empty board will look like. The separator below will only show once you have developed the print_board() function as part of question 2 below.




Note: this is an empty board, each cell holds a space, i.e. space = " ". This space will be replaced with either an X or an O as you play the game.

c. Return the board list.

2. **Go to the function print_board(board) and develop the code to print the board out in the following format:** **[2]**

   a. The *board* is passed in as a parameter.

   b. Each vertical line represents a separator. For example, this board is a 5 x 5 board where an element (i.e. an 'X') could be positioned anywhere, so X is at position row 2, column 2.

```
| | | |
| | | |
| |X| |
| | | |
| | | |
```

After 2 legal moves a board could look like this:

```
| | | |
|X| | |
| |O| |
| | | |
| | | |
```

3. **Go to the function place_row_col(board, option, a_row, a_col) where:** **[7]**
   a. The *board* is passed in as a parameter.
   b. The parameter *option* holds the value of either an 'X' or an 'O'.
   c. The *a_row* parameter and the *a_col* parameter is the position of where an 'X' or an 'O' is to be placed on the board.
   d. The game must always start with an 'X' being placed first. Your code must accommodate for this. If a move is successful, display a message = "Move successfully played" (see template). See page 5 & 6 for output examples.
   e. Only an 'X' or an 'O' is allowed to be placed. If the *option* parameter is not an 'X' or an 'O', a message = "Option isn't X or O" should display (see template).
   f. An 'X' and the 'O' must alternate. So, if on the previous turn an 'X' was placed, if another 'X' is placed immediately after this, then it cannot be placed. A message = "It's not your turn" should display (see template).
   g. An 'X' or an 'O' cannot be placed in a position if an 'X' or an 'O' already exists. A message = "Illegal move, this position already holds an X or an O" should display (see template).
   h. There will always be only **one** message returned. If no messages are sent, then the default is message = " " (see template).
   i. Return the board list and the message variable.
4. **Go to the function check_win(board) and determine if a win has occurred, based on the following rules (see page 7 for examples). The template contains all test cases required to test your project:**

a. **Corners win –** if there are either all 'X's or all 'O's at each corner of the board, this is a win. [7]
b. **Diagonal win -** if there are either all 'X's or all 'O's on the diagonal (top left cell to bottom right cell), this is a win. [4]
c. **Reversed diagonal win -** the diagonal can also be placed the other way around (reversed diagonal, i.e. top right cell to bottom left cell). [4]
d. **Middle cross win –** if there are either all 'X's or all 'O's in the middle of the board, this is a win. A middle cross win means that there will be either five 'X's or five 'O's regardless of the board size. [8]
e. In each of the above cases, the is_winner variable must be returned as a string containing an appropriate message (see template). [4]

**Examples of game-board output, these examples reflect the test cases in the template.**

```
| | | |
| | | |
| | |X|
| | | |
| | | |
```

```
Play 1 move
```

**Move successfully played**

```
| | | |
|X| | |
| |O| |
| | | |
| | | |
```

```
Play 2
moves
```

**Move successfully played**

```
 | | | |

 | | | |

 | | | |

 | | | |

 | | | |
```

Play using an illegal character

**Option isn't X or O**

```
 | | | |

 |X| | |

 | | | |

 | | | |

 | | | |
```

Play 2 X's

**It's not your turn**

```
X| | | |

 | | | |

 | |O| |

 | | | |

 | | | |
```

Play an illegal move

**Illegal move, this position already holds an X or an O**

**(See next page for winning combinations, test cases given in the template are the same)**

**The different ways of winning: (the winning combination is in bold only for you to spot what a win looks like, no need to replicate this in your code):**

**O**|X|X| |**O**

| | | |X

|X| | |

| | | |

**O**| | | |**O**

**O wins on corners**

**O**|X|X| |

|**O**| | |X

|X|**O**| |

| | |**O**|

|X| | |**O**

**O wins on diagonal**

|O|O| |**X**

| | |**X**|O

|O|**X**| |

|**X**| | |

**X**|O| | |

**X wins on reversed diagonal**

X| | |X|

X| |**O**|X|

|**O**|**O**|**O**|

|X|**O**| |

| | | |

**O wins on cross**