Congratulations! You just got a job as a game developer. The company you're working for doesn't develop the kind of games that have detailed graphics, but rather the types of games that you would typically play with a paper and pencil. It's your first day on the job and your boss tells you that 'Ultimate Tic-Tac-Toe 2' needs to be released by the end of the week, and it's your job to pick up where the previous developer left off (she is currently away mastering the art of beach towel origami).

'Ultimate Tic-Tac-Toe 2' is just a regular game of Tic-Tac-Toe with a fancy name so you sell more copies. It must be two-player so that people can play with a friend. As expected, one user will take a turn, the other user will take a turn, and so on until a winner is found or a tie game is reached when the board is filled.

Files

For this assignment, you are asked to submit one file named **tictactoe.py** containing all of the code required to complete this assignment.

Functional Requirements

Tic-Tac-Toe has a few rules we must be aware of:

1. There must be two players. The player that movies first will use the token 'X', and the player that moves second will use the token 'O'.
2. The first player to lay **three** of their tokens in a row wins the game. This can be a horizontal row, a vertical row, or a diagonal row.
3. Once a player wins, the game ends and the winner is announced (in this case, printed).

In 'Ultimate Tic-Tac-Toe 2', both players must enter their name before the game begins. Then, at each turn, the player whose turn it is will have their name announced. When a game is won, the program will print '_____ is the winner!' where the space is filled with the player's name. If the game is tied, the program will print 'Tie game!'.

Specification

As mentioned previously, you are picking up where the previous developer left off. This means that you already have some code to work with! This saves you time and energy. You open your company-provided laptop and look at the 'Ultimate Tic-Tac-Toe 2' repository to see the code provided below. You are permitted (and encouraged) to use all of the code in your solution.

```
board = [['_','_','_'], ['_', '_', '_'], ['_', '_', '_']]

def printBoard():
    '''
    function for printing the tic-tac-toe-board

    boss mentioned that print('some text', end='|') or something similar might be useful
    here's some documentation for it:

    The end parameter in the print function is used to add any string. At the end of the output of
    the print statement in python. By default, the print function ends with a newline. Passing the
    whitespace to the end parameter (end=' ') indicates that the end character has to be identified
    by whitespace and not a newline.

    Maybe this would be helpful in printing the tic-tac-toe board? Further investigation needed
    '''

def checkForWin():
    '''
    The checkForWin function takes no arguments and checks if the board has a win for 'X' or 'O'
    If there is a win on the board for 'X', function returns the string 'X'
    If there is a win on the board for 'O', function returns the string 'O'
    If there is a tie on the board (the board is full but there are no winners), function should return the string 'T'
    ** Note that the ability to check for ties has not yet been written yet, so that needs to be done
    otherwise, if there are no wins, losses, or ties, function returns the string '_'
    '''
    # check rows for a win
    for row in board:
        if row.count('X') == 3:
            return 'X'
        elif row.count('O') == 3:
            return 'O'

    # check columns for a win
    for column in range(3):
        if board[0][column] == board[1][column] == board[2][column] == 'X':
            return 'X'
        elif board[0][column] == board[1][column] == board[2][column] == 'O':
            return 'O'

    # check diagonals for a win
    if board[0][0] == board[1][1] == board[2][2] == 'X' or board[0][2] == board[1][1] == board[2][0] == 'X':
        return 'X'
```

```
    elif board[0][0] == board[1][1] == board[2][2] == 'O' or board[0][2] == board[1][1] == board[2][0] == 'O':
        return 'O'

    # check for a tie game
    # TODO: write this!

    # if no winner is found, an underscore
    return '_'

gameOver = False

playerOneName = input('Please enter the name of player one: ')
playerTwoName = input('Please enter the name of player two: ')


currentPlayerKey = 'X' # the player that moves first starts with 'X', so let's set this to 'X' to begin

printBoard()

while(gameOver != True):
    if (currentPlayerKey == 'X'):
        # print something to say it's player one's turn
    else:
        # print something to say it's player two's turn

    turnComplete = False
    while(turnComplete != True):
        # this should be filled with code that collects the current player's selection for row and column
        # then it should check to see if those are valid inputs
        # if they are, we should lay down that player's token and flip turnComplete to True
        # if either input is not valid, we should ask for a new row and column entry

    # we'll probably want to switch the currentPlayerKey here (to the opposite key for the next player's turn)

    printBoard() # so we can see the move that was just made!

    # we should check here to see if the game has concluded and whether or not there is a winner yet
    # let's try using the checkForWin function above (once it's finished by adding the ability to check for a tie)
```

In addition to the repository, your coworker left some notes scribbled onto a post-it on the monitor. The note reads:

- The function checkForWin needs to be finished up – everything is done except checking for a tie. That check needs to be added in!
- The variable called 'board' holds the 2D-list representing the tic-tac-toe board.

- Thus, board[0][0] represents row one column one (the top left tile) whereas board[2][2] represents row three column three (the bottom right tile).
- The '_' element in the inner lists of board mean that a token has not yet been placed in that tile.
- The function printBoard should print the board in its current state. So, if board is equal to [['_','_','_'], ['_', 'O', '_'], ['X', '_', '_']] then the board should be printed as:

_	_	_
_	O	_
X	_	_
- Remember: Player one will always use the token 'X', whereas player two will always use the token 'O'.

Sample Code Execution

Your coworker conveniently also left some expected output for 'Ultimate Tic-Tac-Toe 2'! How convenient. This can be used to test your final code against to see if it's functioning as expected.

Example One:

```
Please enter the name of player one: Mario
Please enter the name of player two: Luigi

| _ | _ | _ |
| _ | _ | _ |
| _ | _ | _ |
Mario (X), it is your turn!
Row: 1
Column: 1

| X | _ | _ |
| _ | _ | _ |
| _ | _ | _ |
Luigi (O), it is your turn!
Row: 2
Column: 4
Invalid entry, please try again!

Row: 2
Column: 1

| X | _ | _ |
| O | _ | _ |
| _ | _ | _ |
Mario (X), it is your turn!
```

```
Row: a
Column: a
Invalid entry, please try again!

Row: 3
Column: 1

| X | _ | _ |
| O | _ | _ |
| X | _ | _ |
Luigi (O), it is your turn!
Row: 2
Column: 2

| X | _ | _ |
| O | O | _ |
| X | _ | _ |
Mario (X), it is your turn!
Row: 1
Column: 3

| X | _ | X |
| O | O | _ |
| X | _ | _ |
Luigi (O), it is your turn!
Row: 1
Column: 2

| X | O | X |
| O | O | _ |
| X | _ | _ |
Mario (X), it is your turn!
Row: 3
Column: 3

| X | O | X |
| O | O | _ |
| X | _ | X |
Luigi (O), it is your turn!
Row: 3
Column: 2

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

```
Luigi is the winner!
```

Example Two:

```
Please enter the name of player one: Peach
Please enter the name of player two: Daisy

| _ | _ | _ |
| _ | _ | _ |
| _ | _ | _ |
Peach (X), it is your turn!
Row: 3
Column: 3

| _ | _ | _ |
| _ | _ | _ |
| _ | _ | X |
Daisy (O), it is your turn!
Row: 0
Column: 0
Invalid entry, please try again!

Row: 0
Column: 1
Invalid entry, please try again!

Row: 2
Column: 2

| _ | _ | _ |
| _ | O | _ |
| _ | _ | X |
Peach (X), it is your turn!
Row: 1
Column: 1

| X | _ | _ |
| _ | O | _ |
| _ | _ | X |
Daisy (O), it is your turn!
Row: 2
Column: 1

| X | _ | _ |
```

```
| O | O | _ |
| _ | _ | X |
Peach (X), it is your turn!
Row: 3
Column: 1

| X | _ | _ |
| O | O | _ |
| X | _ | X |
Daisy (O), it is your turn!
Row: 3
Column: 2

| X | _ | _ |
| O | O | _ |
| X | O | X |
Peach (X), it is your turn!
Row: 2
Column: 3

| X | _ | _ |
| O | O | X |
| X | O | X |
Daisy (O), it is your turn!
Row: 1
Column: 3

| X | _ | O |
| O | O | X |
| X | O | X |
Peach (X), it is your turn!
Row: 2
Column: 2
The selected space is already filled! Please try again.

Row: 1
Column: 2

| X | X | O |
| O | O | X |
| X | O | X |

Tie game!
```

Non-Functional Requirements

At the top of your code, you should include comments identifying:

- Your name
- Your student number
- Date
- Course name
- Instructor name
- A brief description of your code

Your code should include comments throughout describing chunks of logic.

The code should use informative variable names that follow the conventions discussed in class.

Remember that you should develop your code with Python 3.9 or higher. Failure to do so may result in your code not working when it is tested using Python 3.11 (and thus you will lose marks).

Rules

- Read and follow the instructions carefully.
- Only submit the Python file described in the 'Files' section of this document.
- Submit the assignment on time!
- Late submissions will receive a late penalty of 10% per day up to three days.
- Submissions must be made using OWL. They will not be accepted by email.
- You may re-submit your code to OWL as many times as you like before the deadline.
- Assignments must be done individually and must be your own work. Software will be used to detect academic dishonesty and those found cheating will receive a mark of 0 (and may face additional penalties).

Grading Guidelines

The assignment will be marked as a combination of the output of your tictactoe.py script in addition to your code logic, comments, formatting, style, etc. Below is a breakdown of the grading guideline for this assignment.

- [60 marks] Code execution (does it produce the expected output?)
- [20 marks] Code logic and completeness
- [10 marks] Sufficient comments
- [5 marks] Code formatting
- [5 marks] Meaningful variables names and properly formatted variables Total: 100 marks

Good luck! Gamers everywhere are looking forward to ‘Ultimate Tic-Tac-Toe 2’.