**1.3.1 CONNECT 4 (Players moving at random)**

Connect 4 game between 2 players moving at Random and determining the winning statistics:

Extending the logic implemented in tictactoe.py, the connect 4 game has been developed for players playing in random.

The algorithm used for the implementation of Connect 4 is given below:

1. Initialize the Game matrix for the board size of 6x7.
2. Initialize player number, move counter and set **noWinnerYet** to True.
3. While **move\_still\_possible(gameState)** and **noWinnerYet**:
4. Invoke the method **move\_at\_random** by passing the **gameState** matrix and the current player number.

The random move is marked in the **gameState**.

1. Print the **gameState** to see the move of the player
2. Invoke the method **find\_winner** (Returns True if sequence is formed and False when there is no sequence found) by passing the **gameState** and **player**

* Horizontal Check : Loop through the rows in the **gameState**. Invoke the method **is\_sequence\_formed** by passing the **gameState**, **row** and **sequence** which returns ‘True’ if there’s a sequence of 4 or ‘False’ if there is no sequence found.
* When the **is\_sequence\_formed** returns true:

Invoke **modify\_gameStats**.

Find the index of the sequence formed by using the **position** and the **substring**.

*i=findSubstring(S[position],substring)*

Counter the position in **gameStats**. (Game Statistics)

*gameStats[position,i:i+len(substring)]+=1*

* Vertical Check : Loop through the columns in the **gameState**. Invoke the method **is\_sequence\_formed** by passing the **gameState**, **column** and **sequence** which returns ‘True’ if there’s a sequence of 4 or ‘False’ if there is no sequence found.
* When the **is\_sequence\_formed** returns true:

Invoke **modify\_gameStats**.

Find the index of the sequence formed by using the **position** and the **substring**.

*i=findSubstring(S[:,position],substring)*

Counter the position in **gameStats**. (Game Statistics)

*gameStats[i:i+len(substring),position]+=1*

* Diagonal Check (Left\_to\_Right diagonal and Right\_to\_Left diagonal):

Looping offset in the range *GameState S.shape[1]\*-1 to S.shape[1]*:

Get the diagonals :

*left\_to\_right = S.diagonal(offset, 1, 0).tolist()*

*right\_to\_left = S[::-1].diagonal(offset, 1, 0).tolist()*

If both the sequences are less than 4 , **Continue**

Invoke the method **is\_sequence\_formed** and if it returns true:

* Invoke **modify\_gameStats**.

Find the index of the sequence formed by using the **position** and the **substring** and counter the position in the matrix.

If left\_to\_right:

*i=findSubstring(S.diagonal(position,1,0),substring)*

Counter:

*starts = [0,0+((-1)\*position)] if position<0 else [0+position,0]*

*starts = [starts[0]+i,starts[1]+i]*

*for X in range(0,4):*

*gameStats[starts[0]+X, starts[1]+X]+=1*

if right\_to\_left:

*i=findSubstring(S[::-1].diagonal(position, 1, 0),substring)*

Counter:

*starts = [S.shape[0]-1,0+((-1)\*position)] if position<0 else [S.shape[0]-1-position,0]*

*starts = [starts[0]-i,starts[1]+i]*

*for X in range(0,4):*

*gameStats[starts[0]-X, starts[1]+X]+=1*

1. If **find\_winner** returned true set the flag **noWinnerYet** as false, display the winner and increment the **player1wins** or **player2wins** counter by 1 according to the win.
2. If **noWinnerYet** is True, display the message game ended in draw and increment the **gameDraw** counter by 1.
3. Print the **gameStats** (Game statistics), **player1wins**, **player2wins** and **gameDraw** to get a complete picture of the statistics of 2 players playing in random.

**1.3.2 CONNECT 4 (with user input and GUI)**

pygame version :1.9.1

Program ran on ubuntu version Ubuntu 17.10 which has python 2.7.14 and python 3.6.3 installed

1.The program should be ran by running main\_original.py

2.Once the board appears the desired column in which the coin is to be dropped is to be clicked and hence the game will proceed.