**Computing BSc ALL project 1**

**Semester 1**

**Associated modules 120CT & 121COM**

Group Report – Team TWO

**Introduction:**

This report will outline the stages taken to complete the objective of the Activity Led Learning (A.L.L.) project, which was to build a basic game of Noughts and Crosses (or Tic-Tac-Toe) programmed in Python and to be performed on a Raspberri Pi, functioning as the server, to enable multiple users to play against one another from different PC end-systems. As this was a group project, the report will detail from each group member the contributions made, difficulties encountered and the solutions made during the course of this project.

**Project Management:**

To start we made a Product Catalogue, of all tasks that we needed to complete. This included; designing a game board, Ai element, win conditions, menu, player selection, server/ client, etc. This allowed us to have a clear knowledge of what we needed to complete, and could be used as a progress reference.

We used an AGILE project Plan to allocate time to each task, which was subject to change as we discovered what tasks would take a longer or shorter time than expected. We also used the course plan to see what weeks, certain programming techniques were being taught, so that we could use what we had learnt and implement it into the game. The project plan also had the project brief included for easy reference.

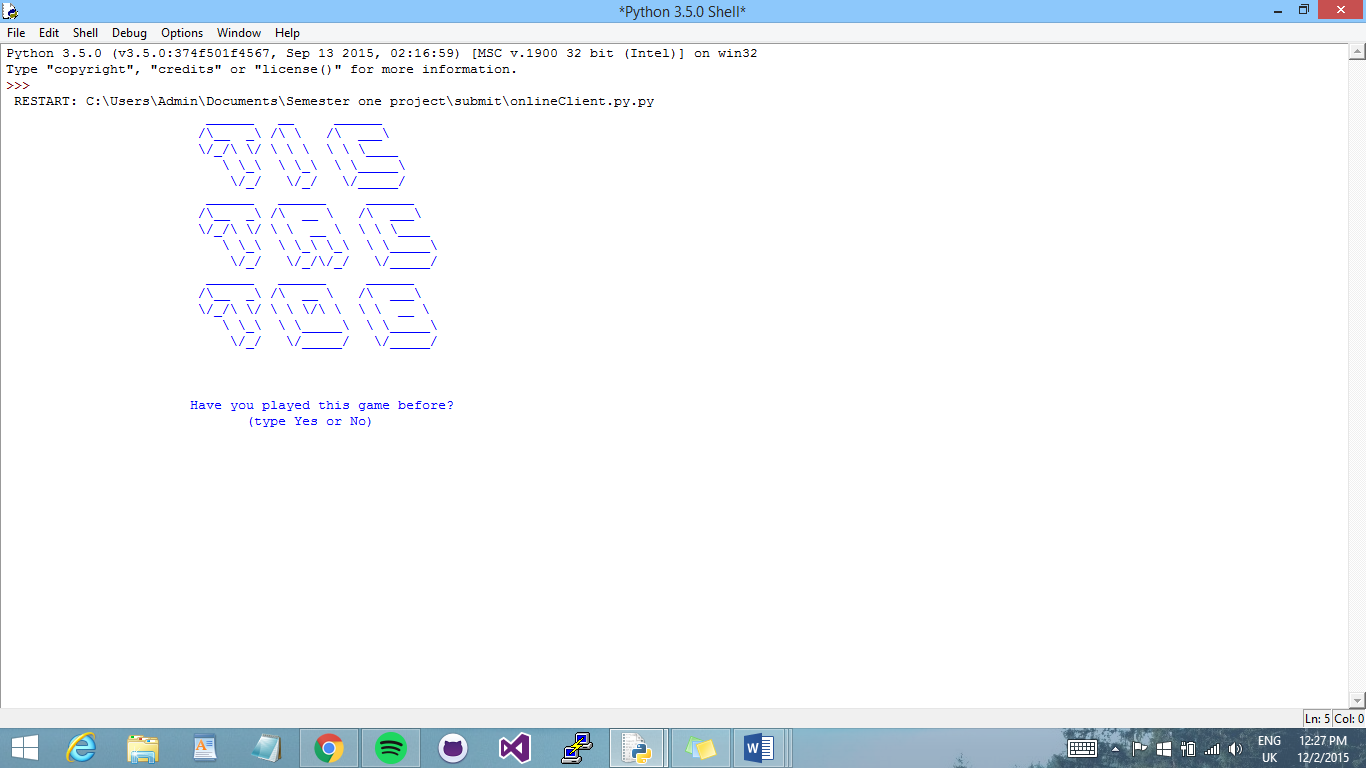
To keep track of tasks and to work on the same files, we used a repository on GitHub. Using this to upload segments of the game, ie; Game Board, win conditions, win results, etc. Which would then be uploaded to the main game file. This was to ensure that each segment would work within the game and avoid errors. It also allowed group members to be able to quickly see what another person has done without having to search through the main game file. We also used a ‘README’ file to use as a checklist for what has and hasn’t been completed, which could be seen within the repository.

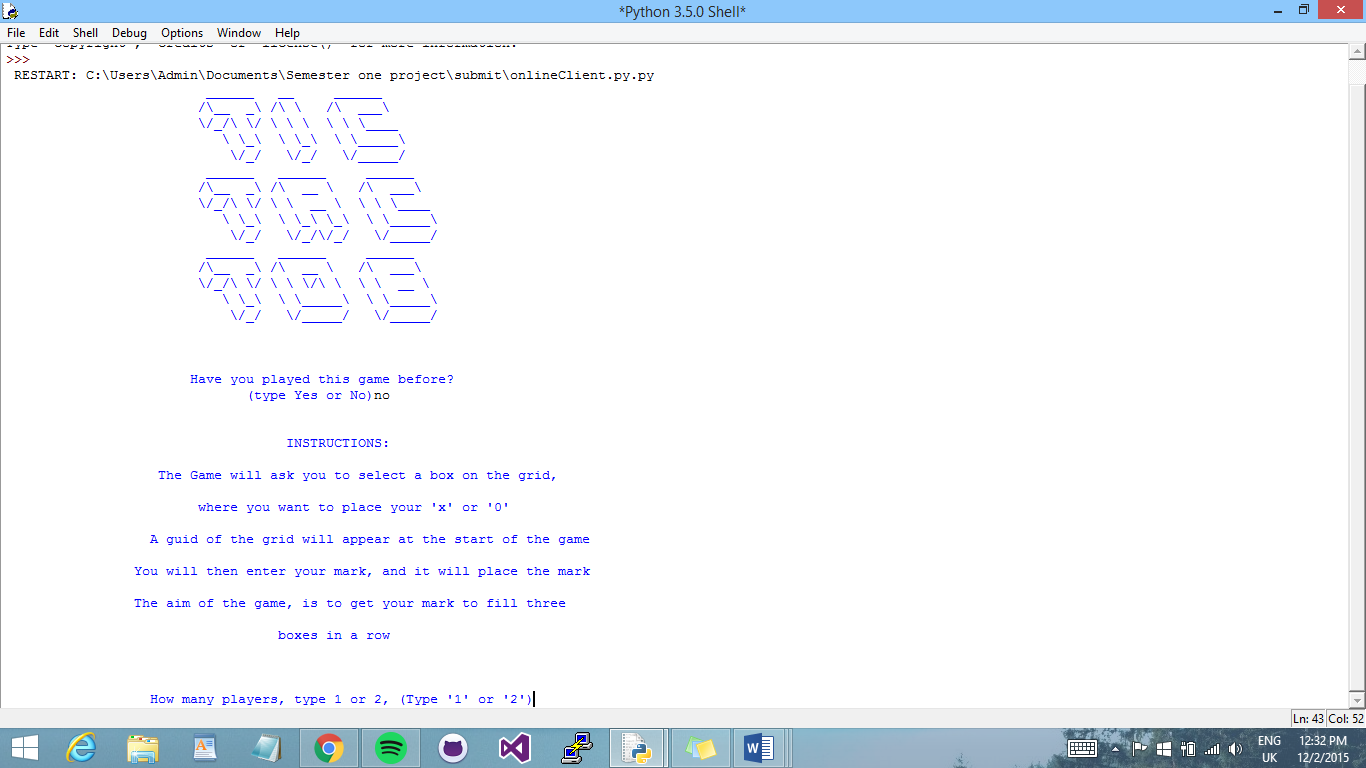
To decide who would work on a specific part of the game, was left open to all group members for them to come forward and state which part that they wished to do. Then they would declare that it was there segment when they uploaded it to GitHub. This allowed everyone to work on the specific part of the game that they felt most comfortable with, but didn’t restrict any person to just doing one certain task.

Despite each group member being able to work on a segment of the game each, often a group member would upload there segment and then another member would add to their code or adapt it slightly. This allowed us to develop segments of the game quickly and to improve each other’s techniques and understanding of the code.

**User Guide:**

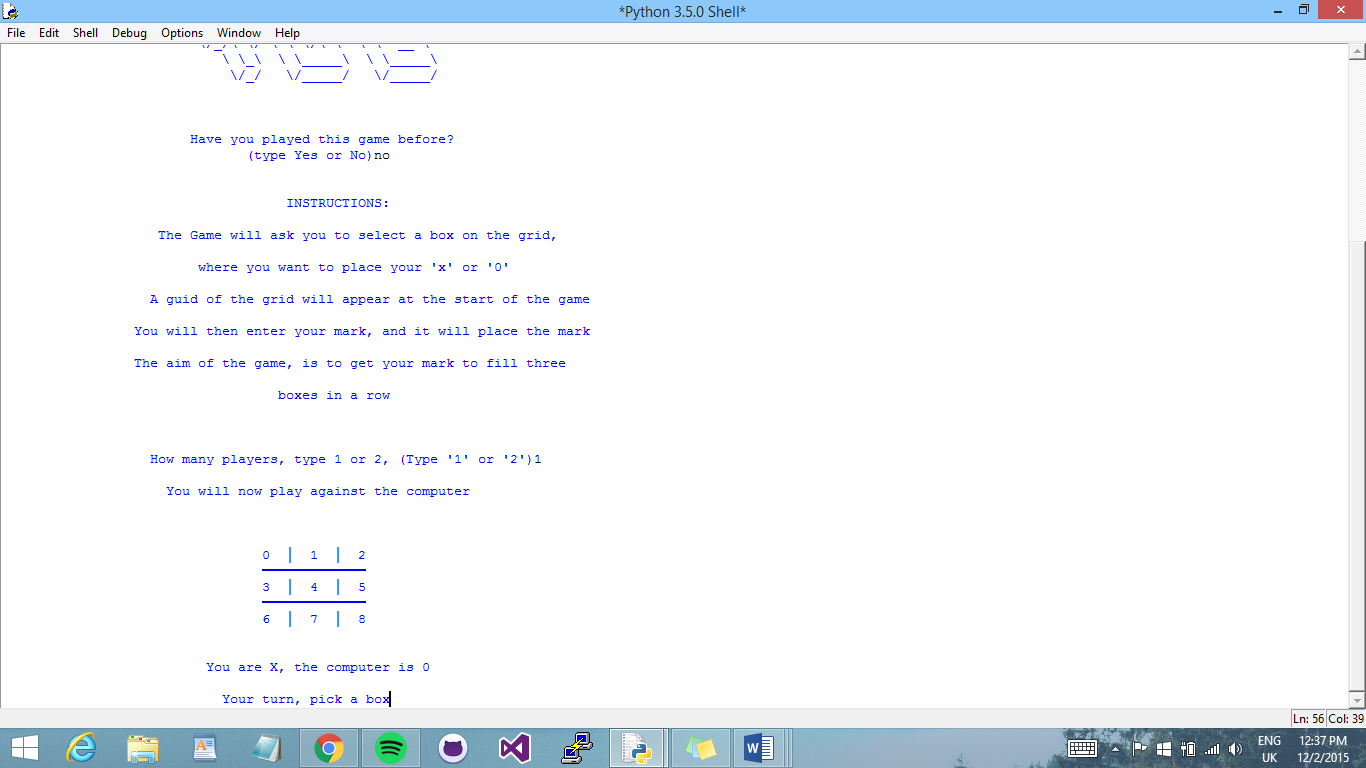
When the game starts, the Title page is show. It will ask if you have played the game before. To reply just type “Yes “ or “No”, on the screen.



If you type ‘Yes’ then you will skip the instructions and move on to player selection. If you type ‘No’, the instructions will be shown, to help you play the game.

After the instructions, you have the player selection options available. You can choose to play one player against the computer, or two player offline or online. If you wish to play two player offline, then the two players must both be on the same computer. However if you choose offline, you can play on two separate computers.

**The Game Board:**



This is the introductory game board. It will be displayed before the start of every game.

It gives reference for players to know which numbers are for which place on the board.

The boxes are in increasing order from right to left. (0 – 9).

During the Game, players will be asked to choose a box to place their ‘X’ or ‘0’, to place your mark just type the number box that you want to place your mark in.

**Playing Against the computer:**

To play against the computer you must select one player. Then the game will start, you the user is player ‘X’ and the computer is player ‘0’. You will go first. As stated before, the reference board is displayed, and the players are declared.

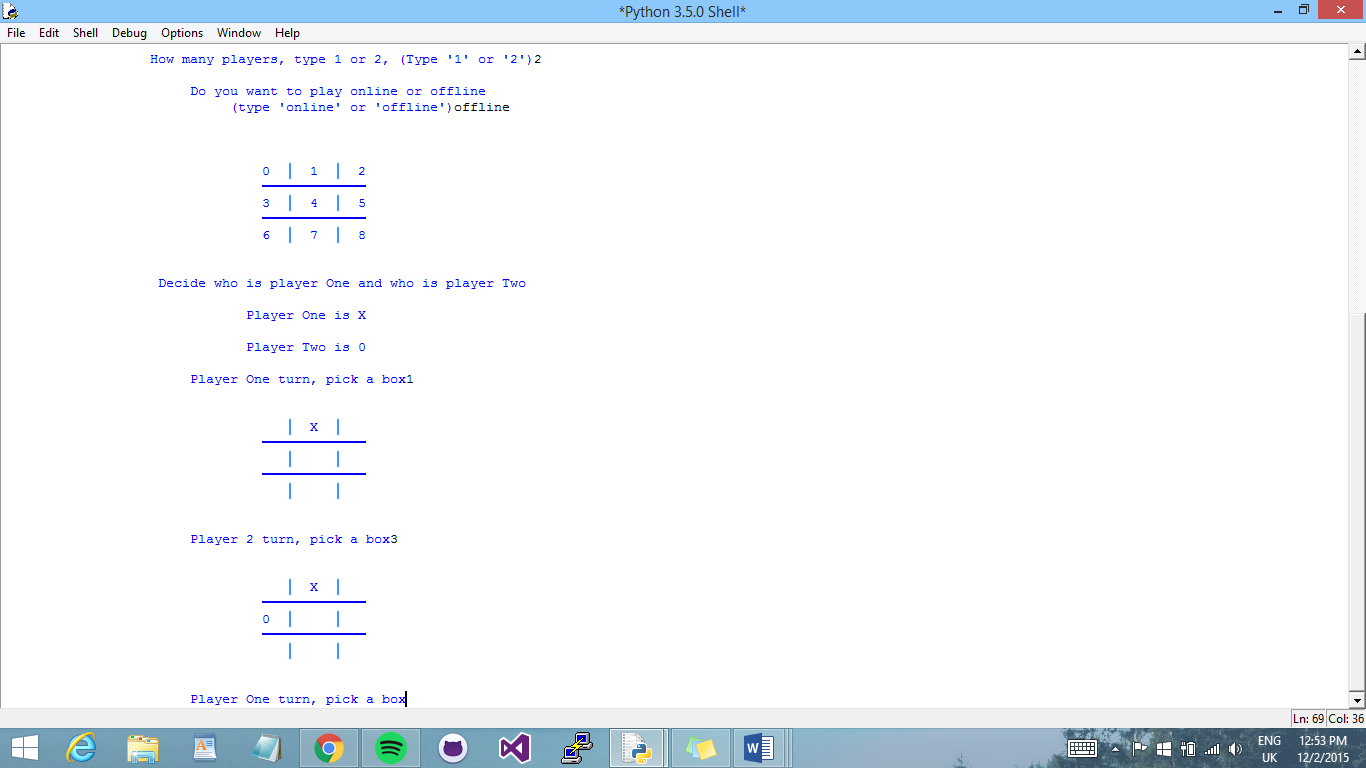
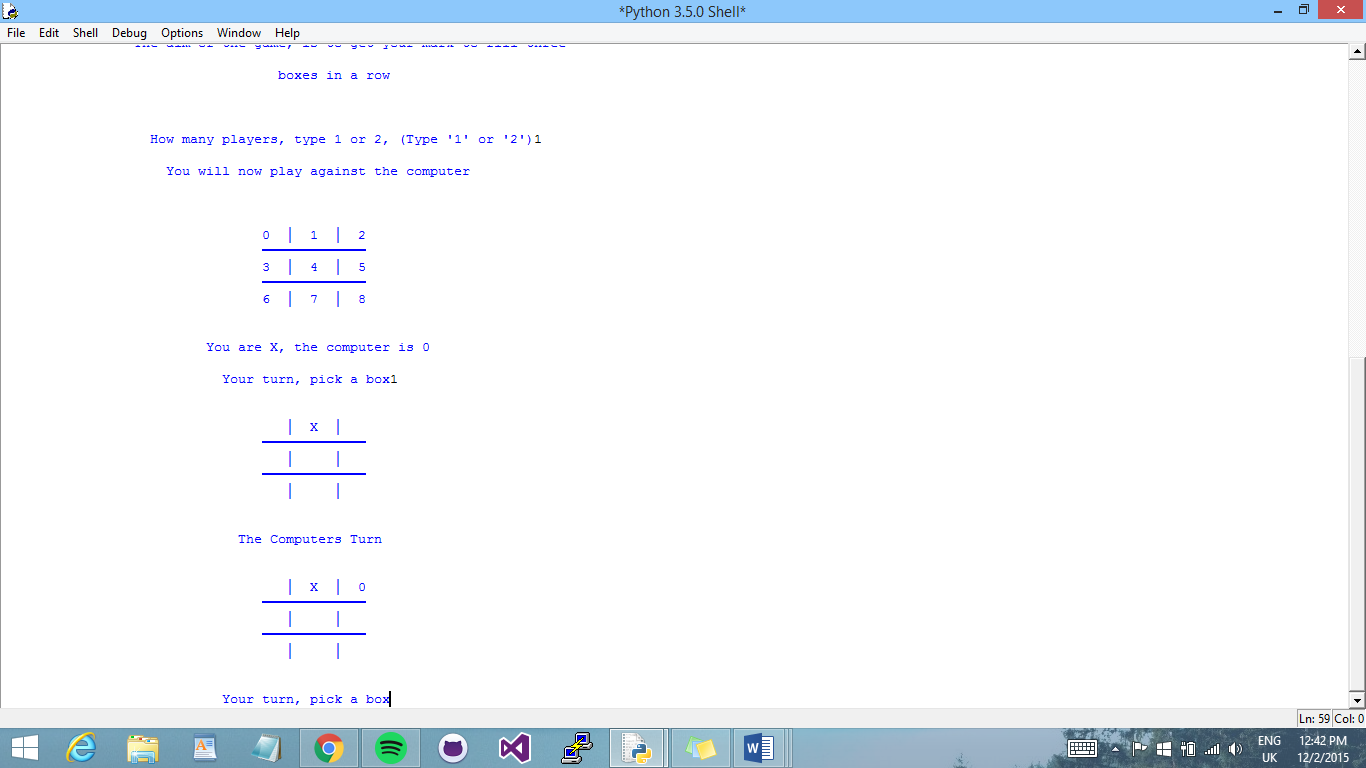
To enter your ‘X’ just type in the number box that you have chosen, and you will see you move displayed on the grid. Player sound will be played. Then it will be the computers turn. There will be a three second delay between the computers turn and yours.

If either the player or the computer, choose a place on the board, that is already taken, it will automatically move on to the next players turn.

**Two Players Offline:**

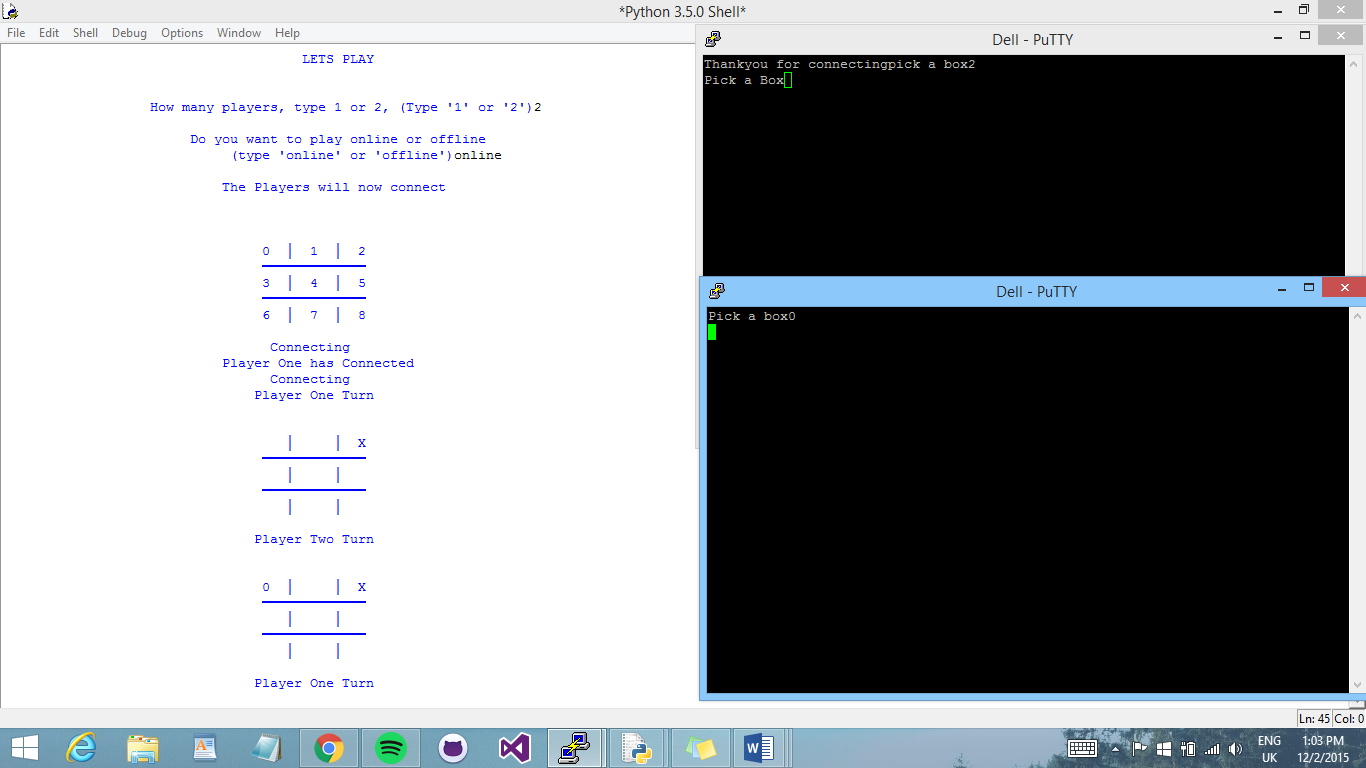
To play, both players need to be at the same computer, because of this the players can decide between themselves who is player one and who is player two.

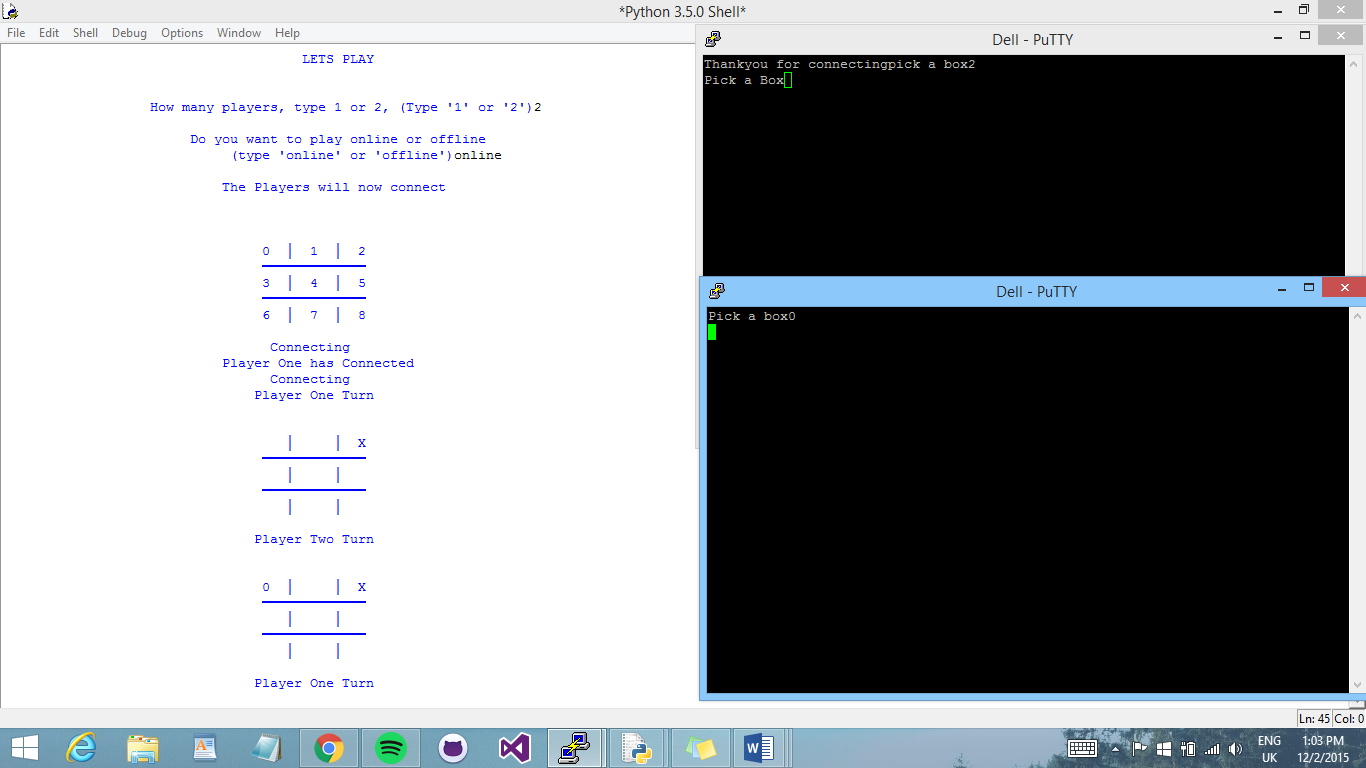
Then the game will start, displaying the reference board. If either player chooses a box that is already taken or unavailable, the next players turn will start.



**Two Players Online:**

To play online each player must have access to PuTTy. This can be downloaded online. When the game shows that it is connecting, each player must connect. The first to connect will be player one, and the other player will be player two.

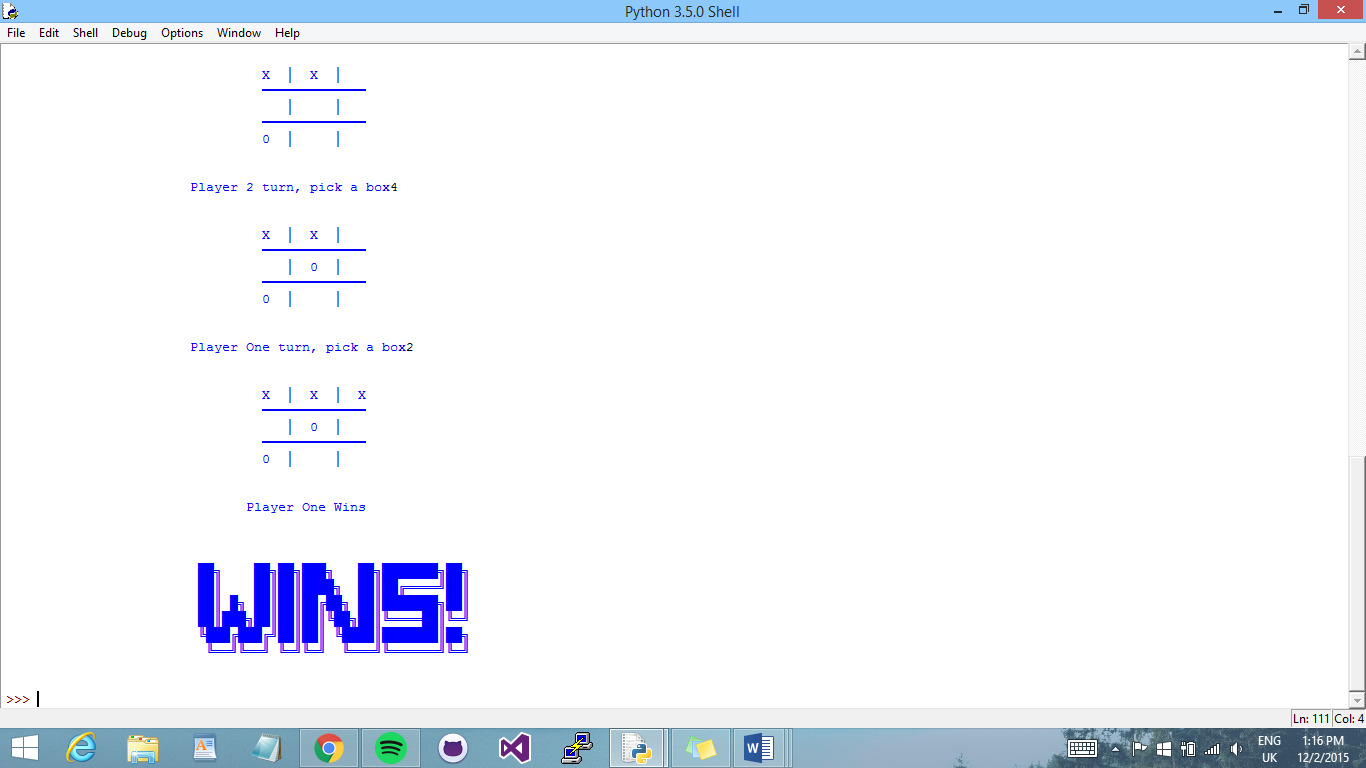
Once both players have connected. Player one will be sent a message to choose a box. Unlike before, when playing online instead of typing input on the game screen, now both players must input there chosen box on the PuTTy window.

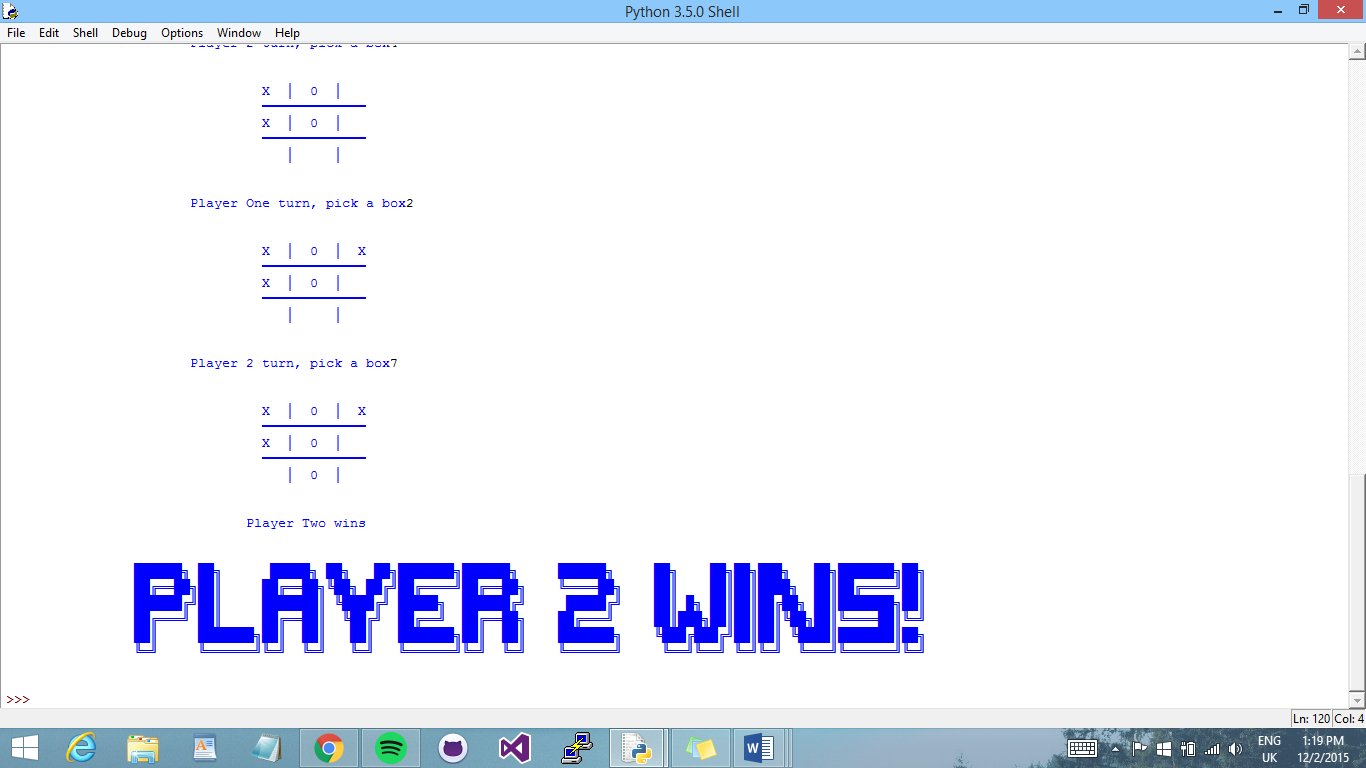


**Winning the game:**

Whether playing one player or two player, the win conditions stay the same. As stated in the instructions for the game, if either player gets there ‘X’ or ‘0’ in three boxes in a row, then they have won the game.

If player one wins then the player one win display will appear:



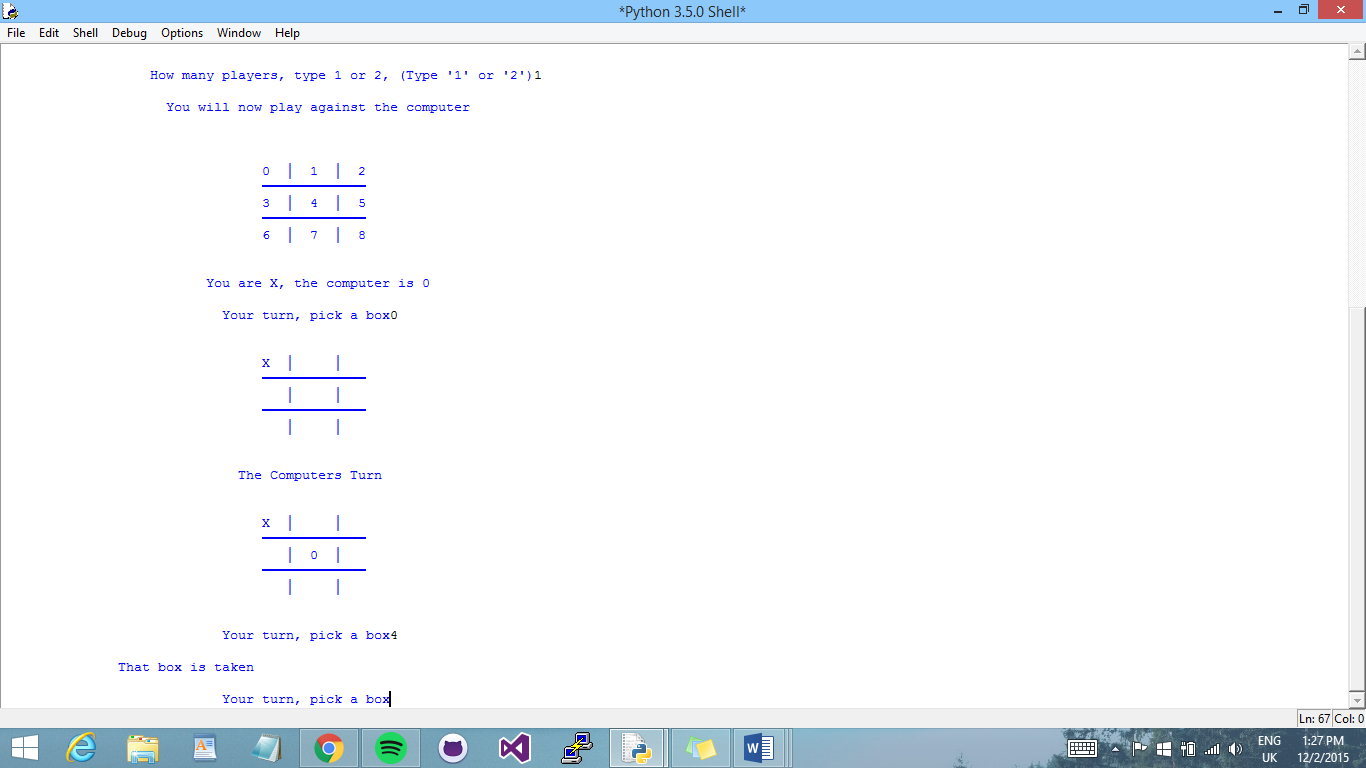
If player two wins then player two win display will be shown:

**Testing:**

The following are tests that we ran and what we did to fix them.

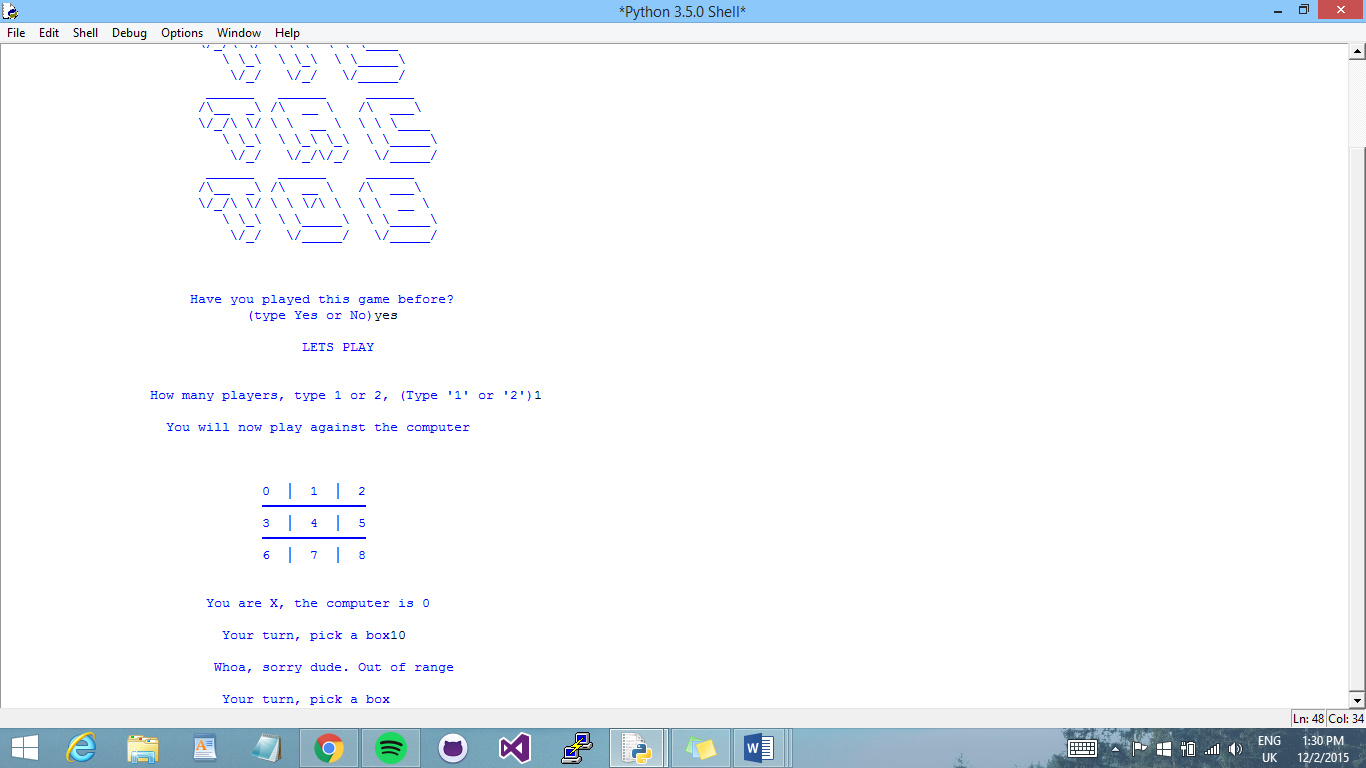
* **Choosing a place on the board that is already taken:**

To ensure that each player cannot put there mark in the same space as another player, we decided to show them the message and then allow them to choose another box on the grid.



* **Choosing a place on the board that is ‘Out of Range’:**

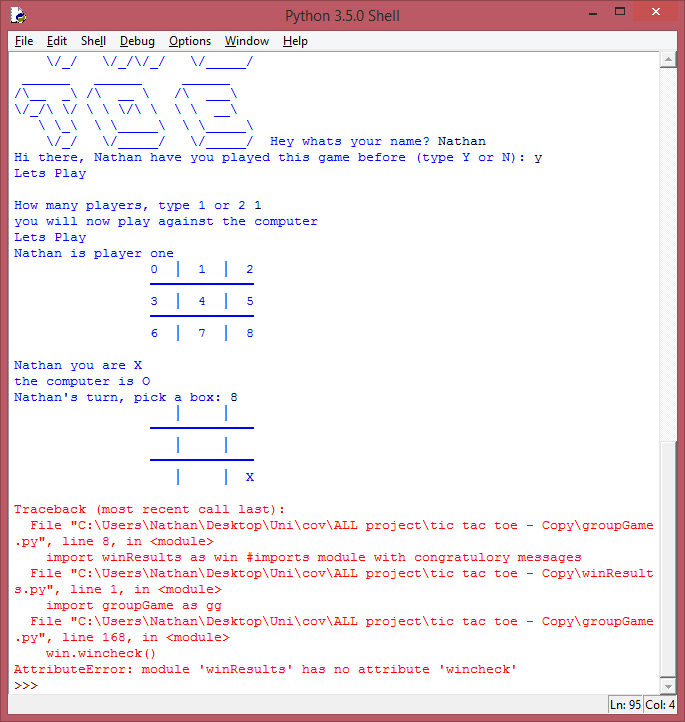
If a player chooses a box that is not on the board, so not in the range of (0-9), it will start the players turn again.



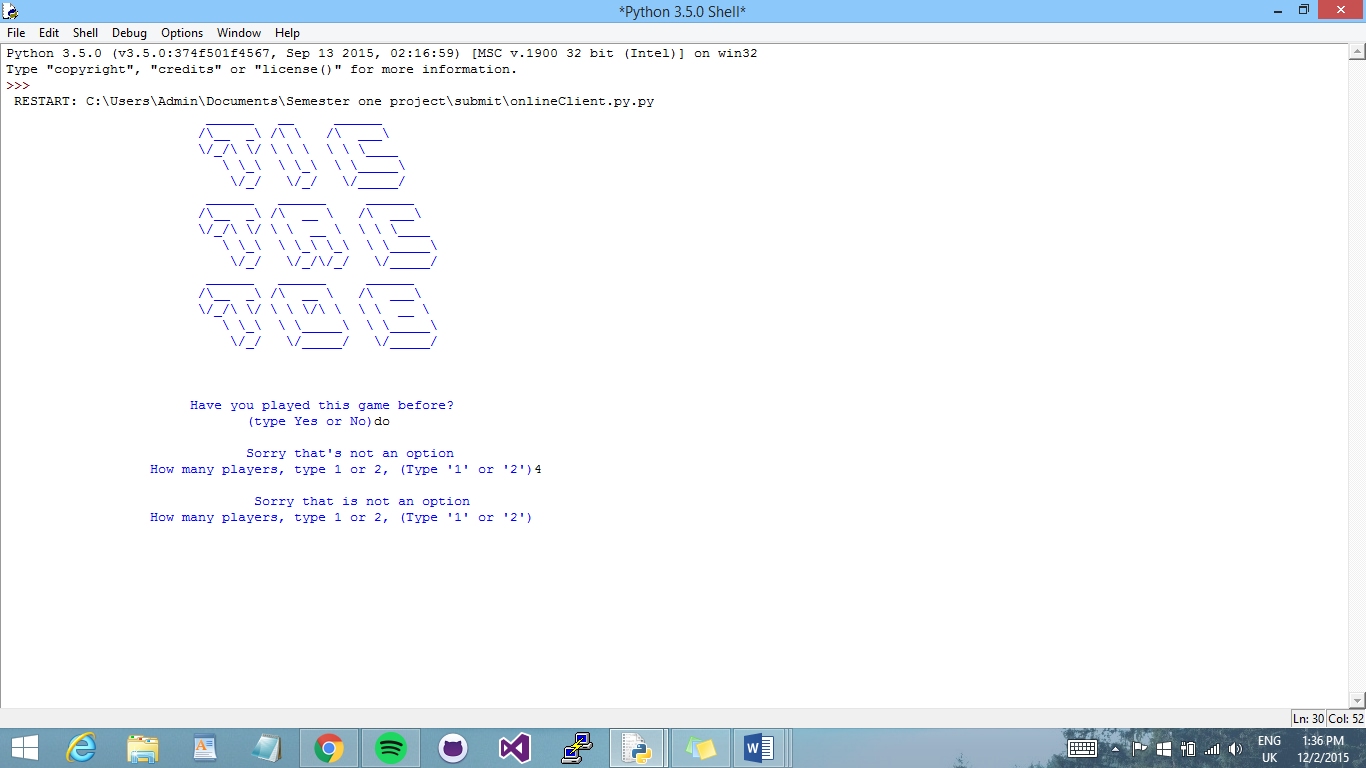
* **Running on different computers:**

When I tried placing the win-condition statements in a separate IDLE module file named *winResults* after encapsulating them inside a function called *winCheck*. I encountered a couple of errors during the course of running the game program. One of which was that the attribute variable ‘name’, assigned within the *winCheck* function, was not found from the separate variable I had made to import into the main game. I approached this error by importing the module file of the main game into the *winResults* module, thus enabling a back-and-forth connection to take place between modules, and the resulting performance of the program found no longer detected this error.





* **Unexpected user input:**



If the user enters unexpected input then, the game will let them know that what they chose is not an option, and continue on instead of producing an error message.

* **Reconnecting players for every turn:**

One of the biggest problems we encountered, was that for online play, the users would have to reconnect to the client every time that it was there turn. To stop this I put the client for each player into two different classes, and defined different methods for each go.

For each players first go, they will both connect but then for each turn after that they do not have to re connect because I used different methods for a first turn and then all other turns.

* **Sending messages and receiving messages:**

Due to the client connections for each player being in different classes. In order to allow the players to send and receive messages in from methods in a different class, I had to define the variable used for the messages to be global, in all classes and methods.

**Work Allocation:**

**Nathan:**

* Win statements
  + Did this by coding an if-statement evaluating if a horizontal, vertical or diagonal row has been filled with an X or O, then either Player 1 or Player 2
* Improved game interface by changing the look of game board
* Adding exit function from sys module to actually exit the game as it did not do so before when a player wins the game; the game continued instead
* Adding a timer so that there’s a delay between users turn and CPU’s turn during the game
* Win Displays in ASCII, (import the module)
* Title in ASCII
* Introductory Game Board.
* Did a few modifications to make the code space more readable
  + Added the string character \n in different places in the code (Figure ?) to separate the display of the content – such as the block of texts, the grid, etc.
  + Rearranged win conditions in different order
  + Used the sleep function from the time module to set a delay in seconds after users turn and before computer puts an input into a random place, to ensure that users are fully aware that the computer’s turn has passed. If this function was not present, the computer will execute its turn in

**Hannah:**

* Client connection
  + - Allowing for two players to connect online, and send and receive messages. Also be able to show the updated game board after each players turn.
* User input onto game board
  + - Showing where the player chose to place their mark.
* Rotation of players turns
  + - Moving from one players turn to the next. Online and offline.
* Allowing for unexpected input
  + - Out of range/ wrong input type and players choosing the same box.
* Play against computer (Ai)
  + - Allow for one player to play against the computer.
* Centred everything to be shown in the play window.
  + - Everything is shown in the middle of the screen, under the grid.
* Comments
  + - All Client comments
* Game start Menu
  + - Instruction select or skip
* Player Selection
  + - Allows player to choose how many players.
* AGILE Plan

**Yavor:**

* Client Connection
  + - Allow players to play online.
* Testing
  + - For unexpected input.

**Joe:**

* Sound affects

**Abdul:**

* Sound effects.

As a team, Nathan, Yavor and Hannah worked well collaboratively, using GitHub and communicating. We knew what parts of the game each other were working on. Once one of us added a segment to the game, the other would always look at the addition and discuss how it was added to the game.