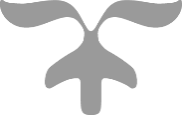


gAME DEVELOPMENT pROJECT

**Software Development – Game: Noughts and Crosses**



June 29, 2018

Student Name : Jason Fernandes

Tutor : Vy Tran   
Line Manager: Katie Hirchovits

Contents

[**Introduction** 2](#_Toc523065764)

[Project Brief 2](#_Toc523065765)

[Purpose of the Document 3](#_Toc523065766)

[Scope 3](#_Toc523065767)

[Who with? 3](#_Toc523065768)

[**Project Overview** 4](#_Toc523065769)

[Solution Overview 4](#_Toc523065770)

[Project timescale and dependencies 5](#_Toc523065771)

[Requirements 6](#_Toc523065772)

[Issues 8](#_Toc523065773)

[Risks 8](#_Toc523065774)

[Process Map 9](#_Toc523065775)

[**Development of the Program** 10](#_Toc523065776)

[Version 1 - Code 10](#_Toc523065777)

[Version 1 – Output of code 11](#_Toc523065778)

[Version 2 - Code 12](#_Toc523065779)

[Version 2- The output of the code 13](#_Toc523065780)

[Version 3 - Code 14](#_Toc523065781)

[Variables used in the final version of the code 20](#_Toc523065782)

[**Testing** 21](#_Toc523065783)

[Test plan (Before carrying out testing) 21](#_Toc523065784)

[Test plan results (After carrying out testing) 22](#_Toc523065785)

[**Documentation** 29](#_Toc523065787)

[Minimum Hardware and Software requirements 29](#_Toc523065788)

[**Reflection of Code** 30](#_Toc523065789)

[Evaluation of the results 30](#_Toc523065790)

[**Deployment** 32](#_Toc523065791)

[**User Guide** 33](#_Toc523065792)

[How to prepare to play Noughts and Crosses? (From Python) 33](#_Toc523065793)

[How to prepare to play Noughts and Crosses? (From BQUANT on Bloomberg) 33](#_Toc523065794)

[How to run Noughts and Crosses? (From Python) 34](#_Toc523065795)

[How to run Noughts and Crosses? (From BQUANT) 34](#_Toc523065796)

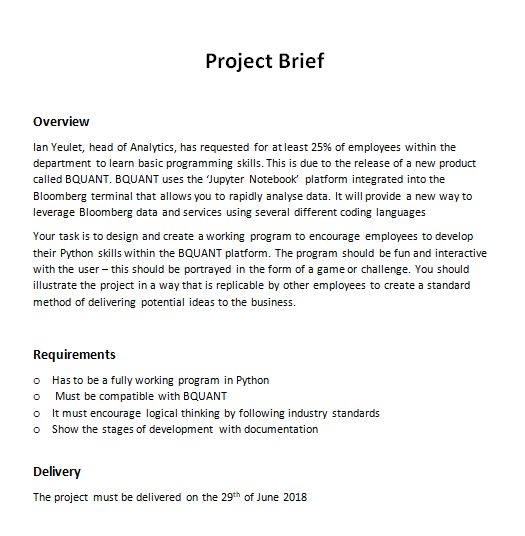
[How to play the game (From Python) 36](#_Toc523065797)

[How to play Noughts and Crosses? (From BQUANT) 38](#_Toc523065798)

[Troubleshooting 40](#_Toc523065799)

[Full Code with Comments on BQUANT 41](#_Toc523065800)

# **Introduction**

Project Brief  
My manager: Katie Hirchovits [Analytics Team Leader] has assigned me to a project that would focus on the release of a new product “BQUANT” - A tool which helps clients evaluate data through programming. However for clients to know about this tool, employees would need adequate knowledge of this product so trainings are able to be conducted. Therefore this project would entail planning, designing and documenting a way in which to make the launch of BQUANT known to other employees, preferably in a fun, productive way so that they are enticed to learn about the product in their own time as explained below:

## Purpose of the Document

The reason why I am creating this document is because it will show the evolution of the game which I have chosen to make due to the reason that it is a fun, interactive way in which will entice employees to learn more about the product. Noughts and Crosses is the game I have chosen to develop on the BQUANT platform and proof test the results on Python aswell. The reasoning behind Noughts and crosses is because it is a game that requires logical thinking in order to win and the same can be said for my role in Analytics, as logical analysis is needed to be able to understand the client concerns and give a correct response. Furthermore, both require being able to predict the next step as in Noughts and Crosses a player has to predict the next move, while in Analytics I am faced to anticipate a client’s next query/follow up questions in order to provide the best customer service

## Scope

This game will be released for 600 employees whom are located in the Analytics department. This is because they will need to familiarize themselves with the BQUANT platform, in order to solve questions from clients about this. Noughts and Crosses developed on BQUANT will not only promote teamwork skills since it is a two player game, but will also provide a way in which employees can take a break to relax their minds from daily tasks whilst enhancing their familiarity with the product.

## Who with?

I will be conducting most of the stages in the software development life cycle alone for this project; however, I will be regularly reporting the progress of this task to my manager, Katie Hirchovits. Furthermore, I will be getting end users within the department to test my game and give me their opinions as to what they like and areas to improve on the analysis/testing phase.

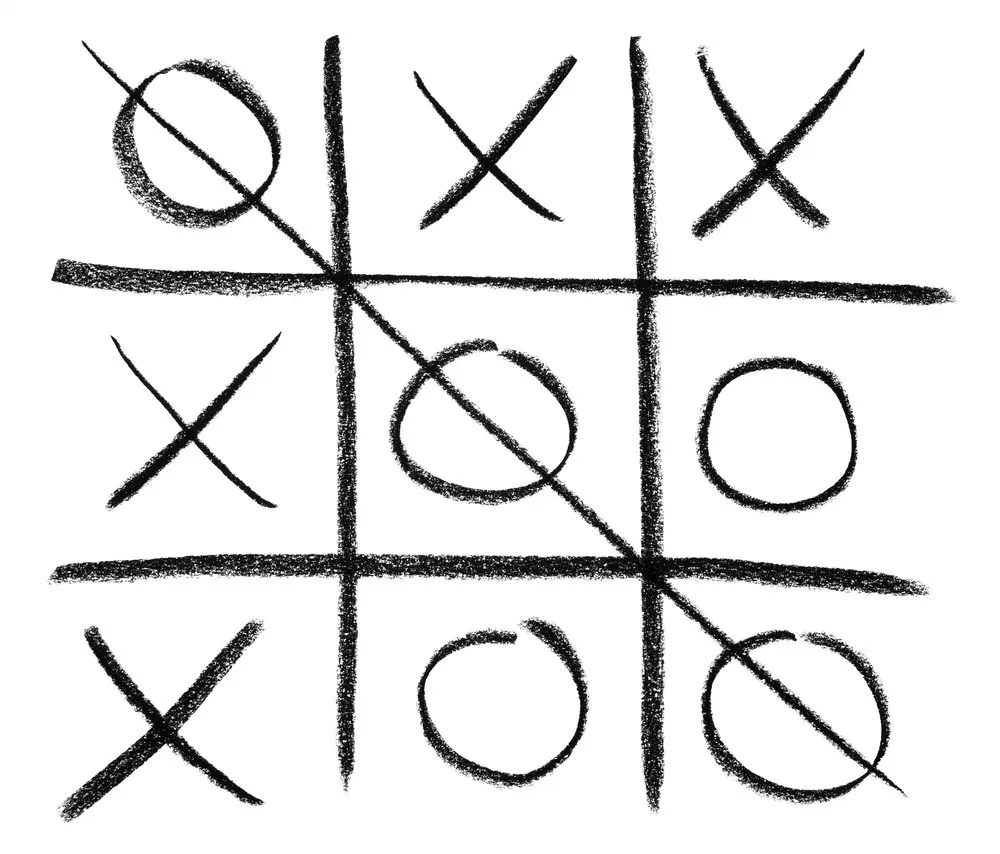
# **Project Overview**

## Solution Overview

* **How the game should work**

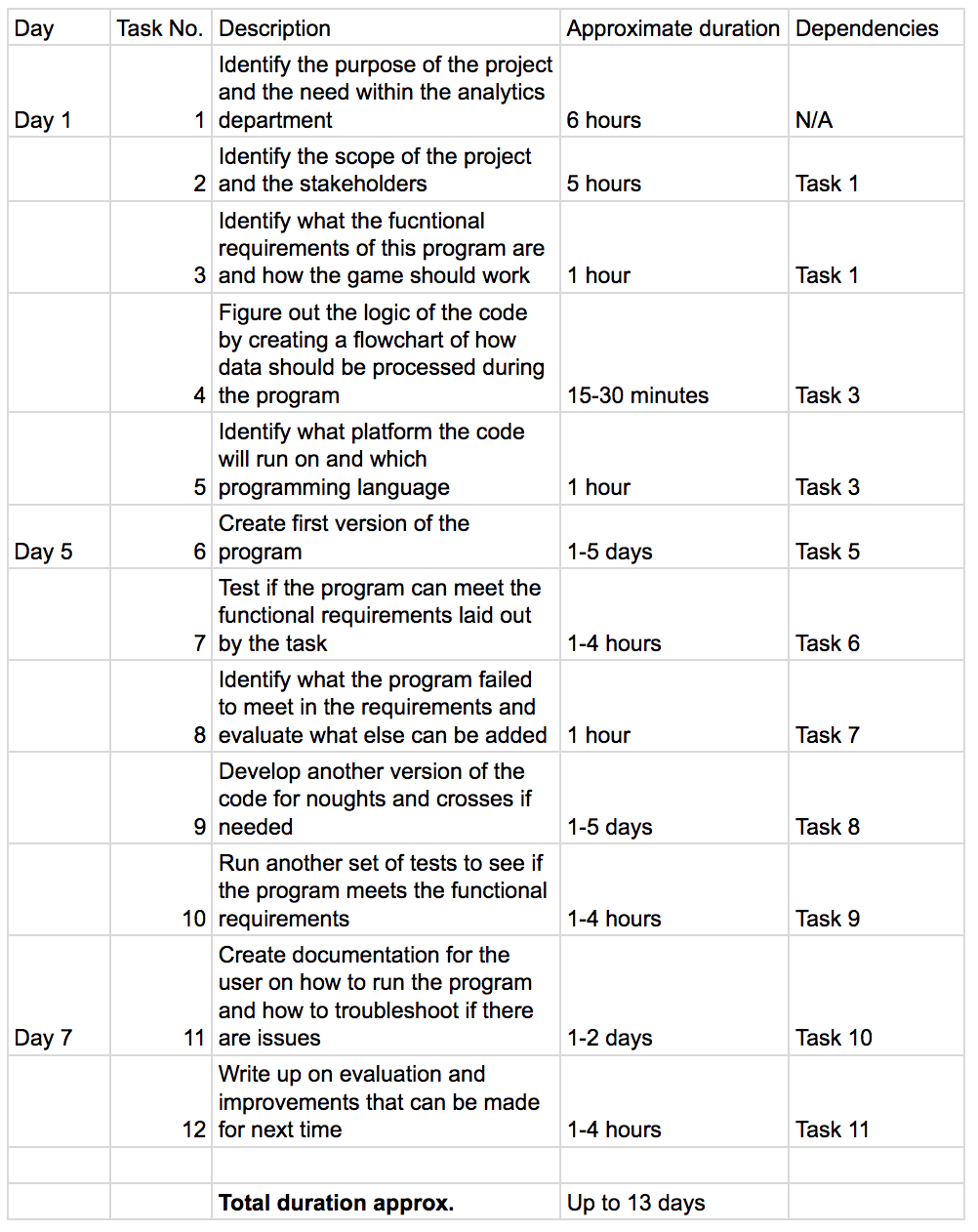
I will be creating a game of noughts and crosses on the new BQUANT platform, which has integrated python as well as other programming languages. How it should work is that when the program is first loaded, a game board is drawn up, a 9x9 square to be precise. It will then ask the user a question as to either selecting X or 0. Once the user inputs either the X or 0, player 2’s symbol will be determined automatically. For example, if player 1 selects X, then player 2 will be automatically allocated the 0 to play with and vice versa.

Once the symbol in which they are playing with are selected [X or 0], player 1 will be asked to input a number from 0-8 because this number will determine which square they want to place their chosen symbol in. Then the result will be outputted in the allocated place on the 9x9 grid. The program will then continue until a winner is able to achieve three of their symbols in a row and the message ‘’Player [1 or 2] is the winner!’’ will be outputted from the program.



## Project timescale and dependencies

I carried out the development of the program over a 2-3 weeks, which involved identifying what the code should do and creating multiple versions of my code and testing if the code met the requirements.



## Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req number | Functional Requirements | Inputs required | Processing Required | Outputs |
| 1 | Run the program and wait for the welcome message where the user will be required to choose a symbol | “X” should be the symbol inputted by the user | * Verifies that “X” is a valid symbol * Store and save Player 1 as symbol, “X” * Store and save Player 2 as symbol, “0” | * Message outputted,   Player 1 is “X”, Player 2 is “0” |
| 2 | Run the program and wait for the welcome message where the user will be required to choose a symbol | “0” should be the symbol inputted by the user | * Verifies that “0” is a valid symbol * Stores and save Player 1 as symbol, “0” * Store and save Player 2 as symbol, “Z” | * Message outputted,   Player 1 is “0”, Player 2 is “X” |
| 3 | Run the program and wait for the welcome message where the user will choose their symbol | “Z” should be the symbol inputted by the user | * Verifies that “Z” is a symbol | * Error message should state that “Z” is not a valid symbol. |
| 4 | Run the game until Player X wins | X,0,1,4,2,8 The inputs above are the positions that Player X could enter to win | * Verifies the player that is assigned to symbol “X” has got three symbols in a row | * Message outputted,   Player X wins! |
| 5 | Run the program until Player 0 wins | 0,0,1,4,2,8 The inputs above show the positions that Player 0 could enter to win | * Verifies the player that is assigned symbol “0” has got three symbols in a row | * Message outputted,   Player 0 wins! |
| 6 | Run the program and game until no one wins | X,0,1,2,3,4,5,6,7  The inputs above show the positions that both players could enter to draw | * Verifies that both players have not been successful to match three symbols in a row | * Message outputted,   It is a draw |
| 7 | Run the program and enter the same position as the last player | X,0,0,0  The inputs above show the positions that both players could enter | * Checks that the position in which the player has chosen has not been already picked/filled. | * Message outputted,   “That position has been taken! Try again” |

## Issues

Developing a python code on the Bloomberg terminal will encounter some issues:

- The programming aspect within analytics is fairly new and has not been fully rolled out, so not many people within the department will be aware of how the code works or what the purpose of the code is at this point

- The python version that is approved to run within Bloomberg may not be the same version that is used to develop the code outside of Bloomberg

- There may be compatibility issues as a result if the python version that Bloomberg has approved is different to the one that is used to develop the program outside of Bloomberg

## Risks

- There are data confidentiality issues at Bloomberg with data, working with programming languages may cause security issues

- It should be ensured that the program does not deal or contain any sensitive information regarding Bloomberg

- The project may overrun if the code cannot be done in an efficient way

- There may be unknown variables that are encountered during development of the code

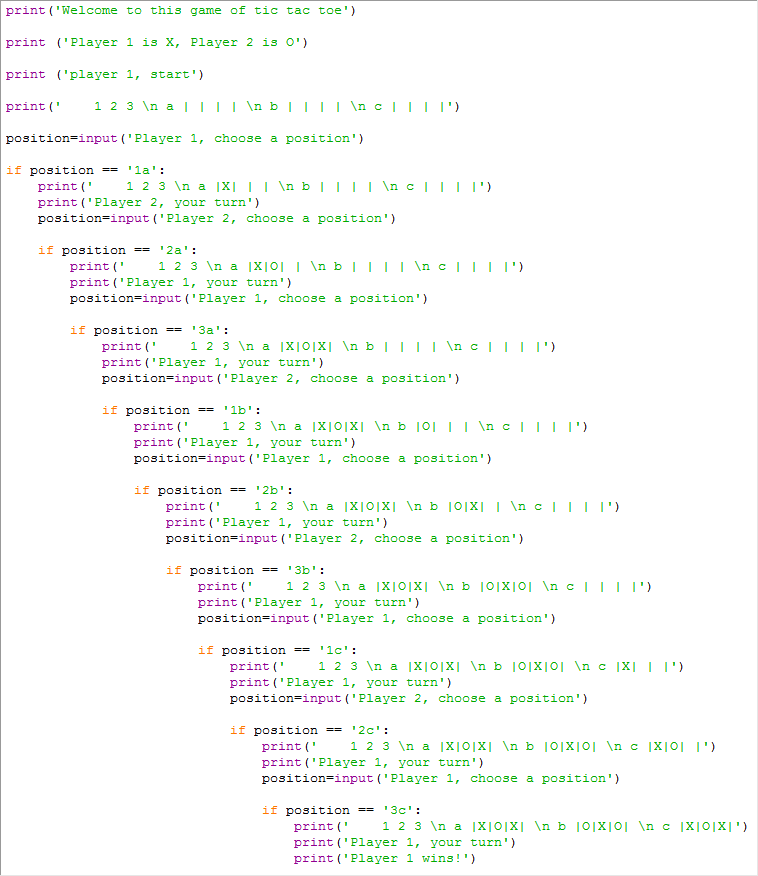
## Process Map

# **Development of the Program**

Whilst I was developing the program, I came up with different ways as to how this game could work. There were three versions in particular which maps out the key stages of the creation upon the final product. Below I will explain each version and why I didn’t settle for that to be the final program – Version 1 and 2 were coded in Python, while version 3 was finally integrated on the BQUANT platform.

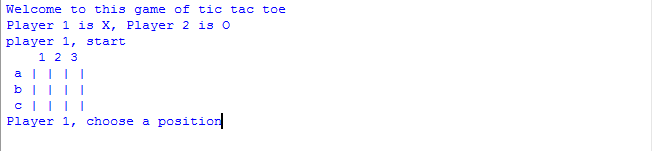
### Version 1 - Code

Version 1 was a very basic code it worked and it eventually led to a winner. However, this was extremely inefficient as it meant that every single combination would have to be coded out. To do this would take many hours, and over 1000 lines would be required. The problem was that there was no verification of the user input nor did the code stop once a winner was determined.

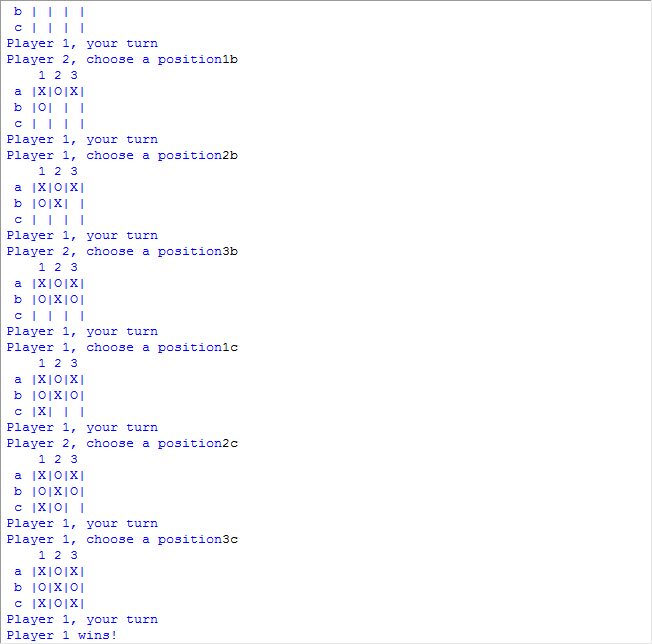


### Version 1 – Output of code

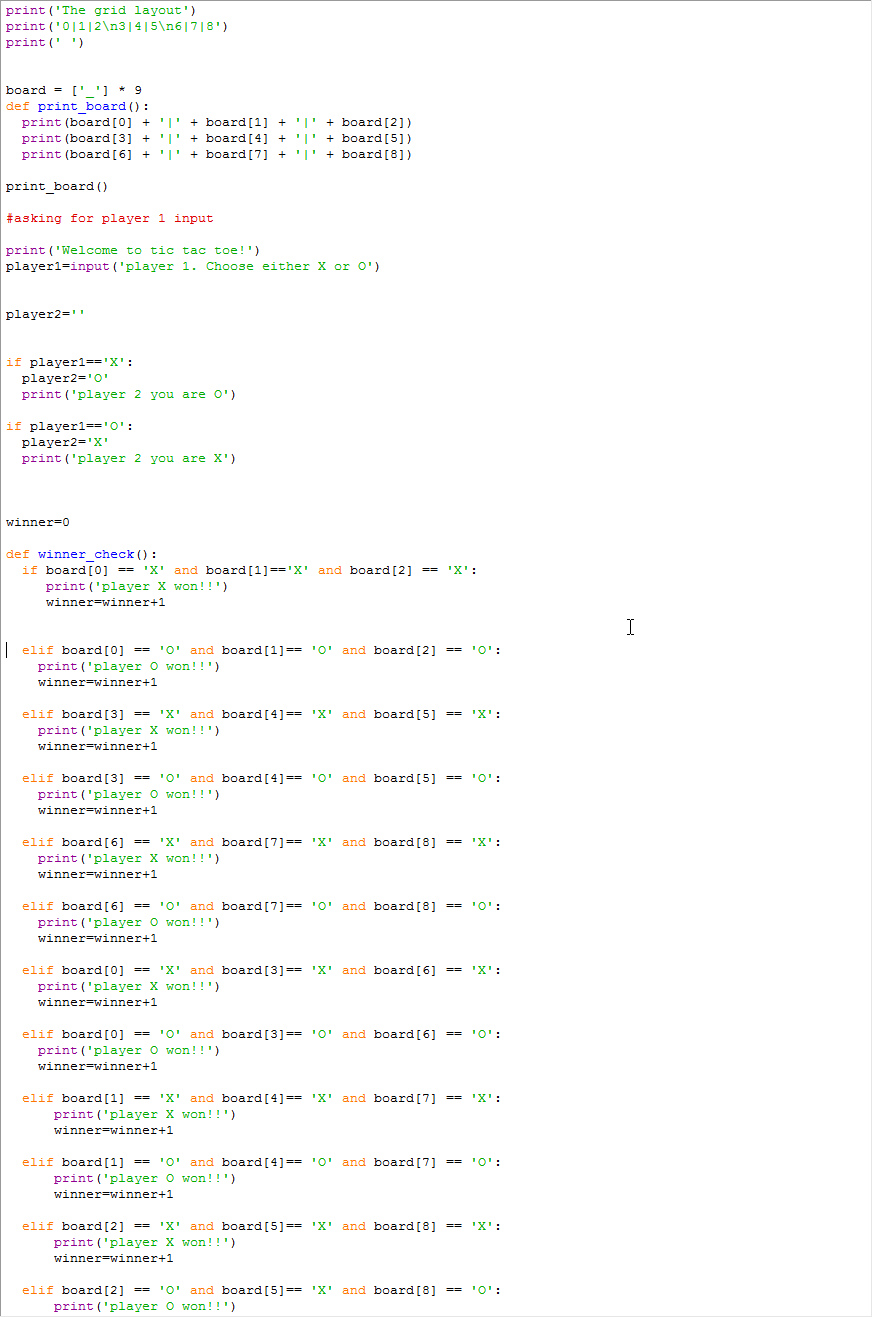
When the program runs, the welcome screen of the code looked like this:



In addition, the winner of the game would be announced like this:



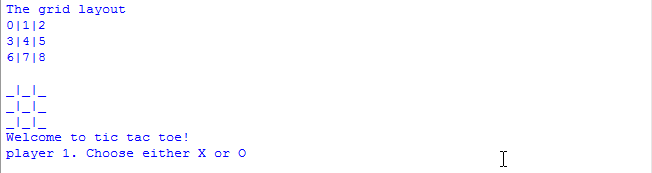
### Version 2 - Code

Version 2 was a lot more efficient in terms of handling the way in which the user would choose a position on the grid to place their symbol. Making use of the “list” function where indexes are able to be used to identify grid positions meant that I would not need to manually code what the play positions would look like each time someone choses a position [Version 1]. Every time a player chose a position, it would print out the grid with the chosen position. It also gave the player a choice on which marker they wanted to be.

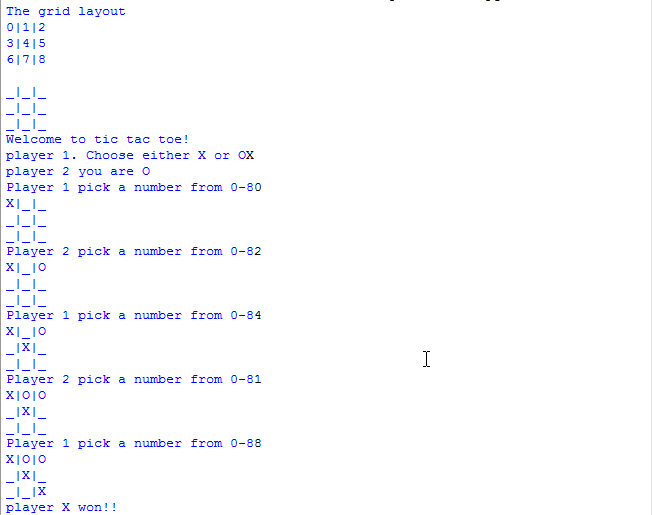
When a winner was found, the winner variable would become 1 and it will be printed onscreen that a winner has been found

### Version 2- The output of the code

When run, the welcome screen of the code looked like this:



In addition, the winner of the game would be announced like this:

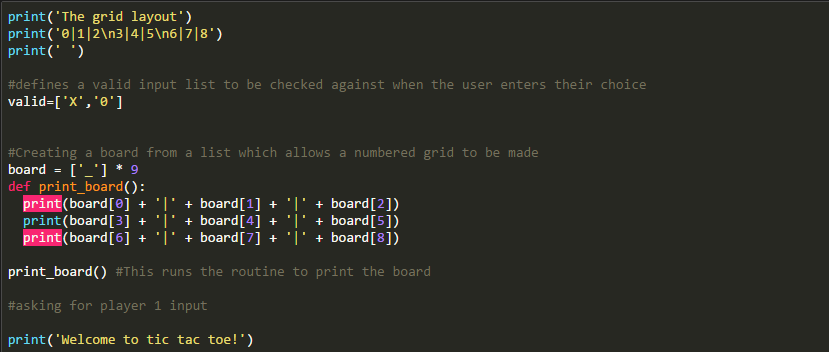


### Version 3 - Code

Version 3 of the code was most efficient piece of code when compared to the other two versions. It built on top of the basis of version 2 and included verification of the users input. For this version I used BQUANT to program the game, as the end goal was to release this in order to promote the new product. The code below has been broken down and will be explained in sections.

**Section 1** - This is where the board is defined and board as a variable is set to ‘\_’ to show the user an empty space. It is multiplied by 9 to create a list that can be indexed from 0-8. The board is printed and shown to the user. Indexing allows the user to have the choice to choose a position that will be printed out on the board.

**Code**:

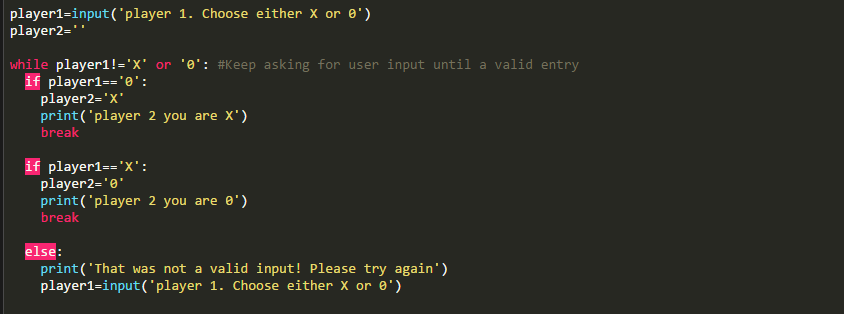


**Output**:



**Section 2** - This is the first user input where the user is asked to choose either X or O. If they don’t put the right input, the code will output saying it was not a valid input!

**Code**:



**Possible Outputs**

**Output 1**:



**Output 2**:

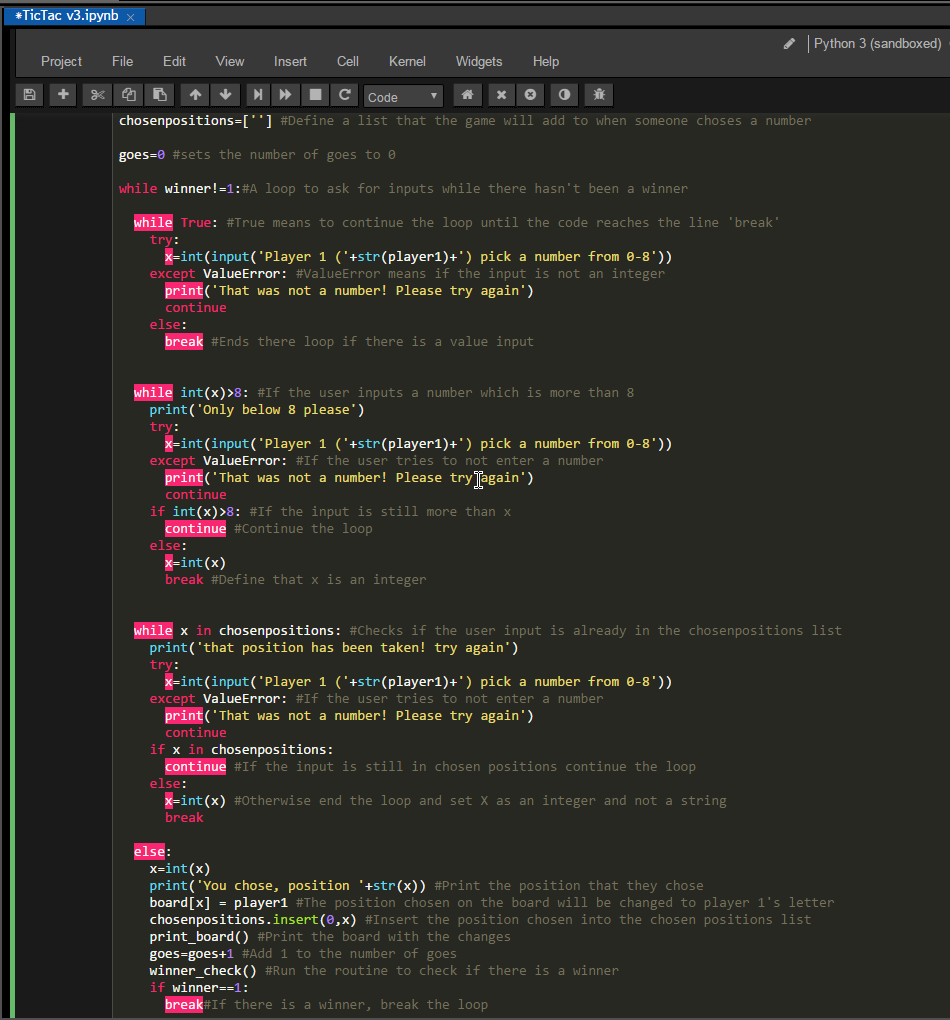


**Output 3:**



**Section 3** - The user will be asked to select their position on the grid, and before it gets placed, the user input will need to go through a number of validation while there is no winner. (1) Ensuring it is a number (2) It is less than 8 (3) It hasn’t been chosen yet by anyone. This is achieved by the code creating a list and recording to it the positions that have already been selected. Once all of the validation is done and the input is correct, it will print out the position chosen, redefine the board grid number with the player’s chosen letter and add the number chosen to the chosenpositions list. It will also then run the winner check routine which is defined before the while loops and will be explained.

**Code**:



**ChosenPositions** is the variable used to define the list that will be added to when a user enters a number. Goes is the variable that defines the number of goes that each Player 1 and 2 have had combined and is increased by 1 after each correct input.

**Possible Outputs**

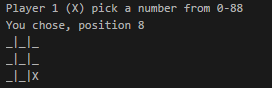
**Output 1**:



**Output 2**:



**Output 3:**

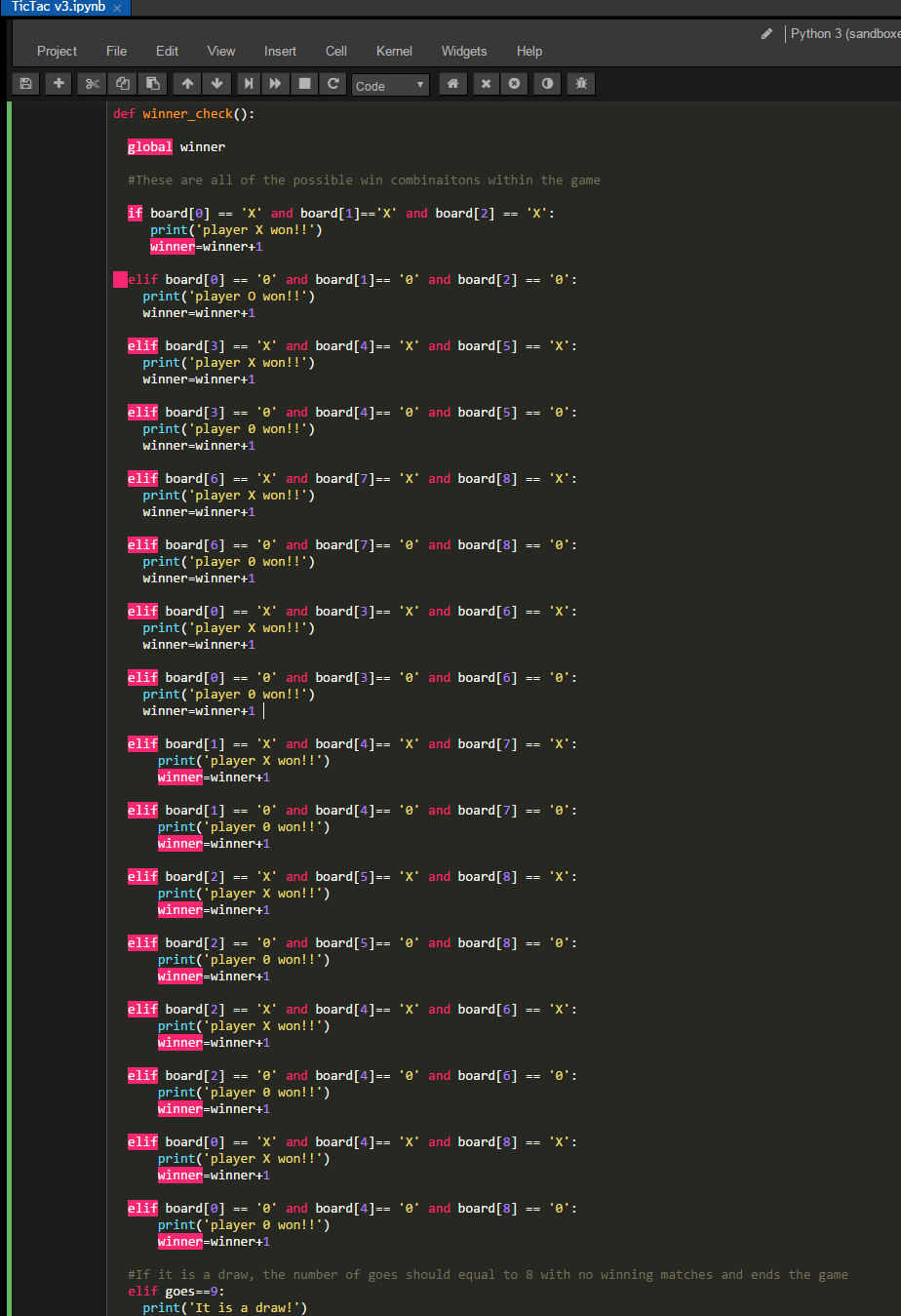


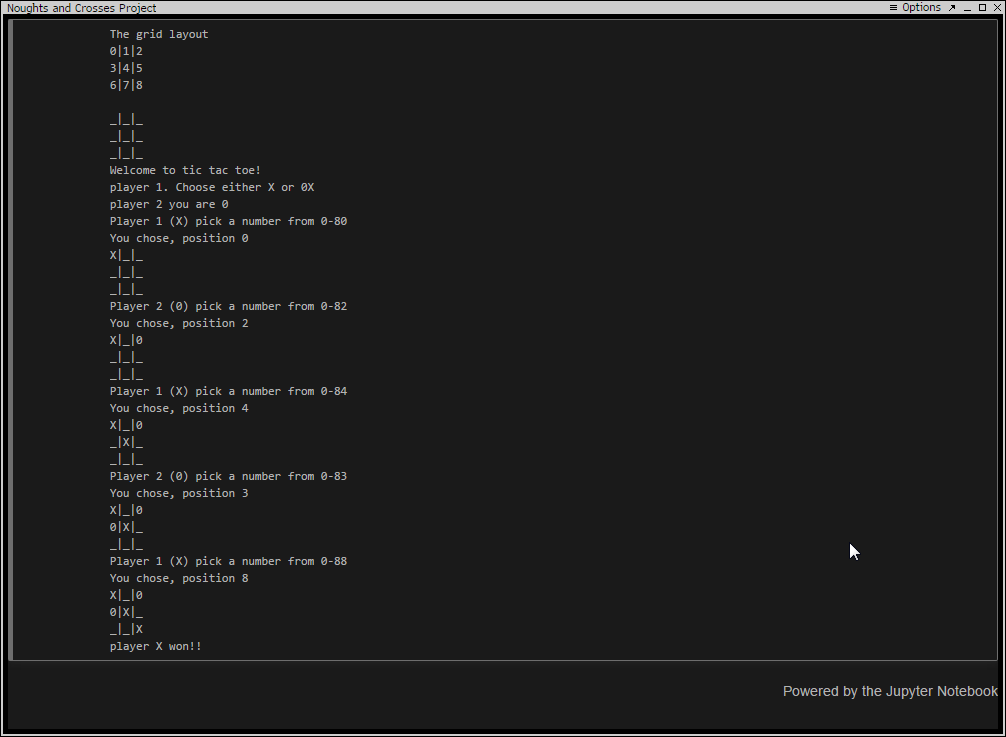
**Output 4:**

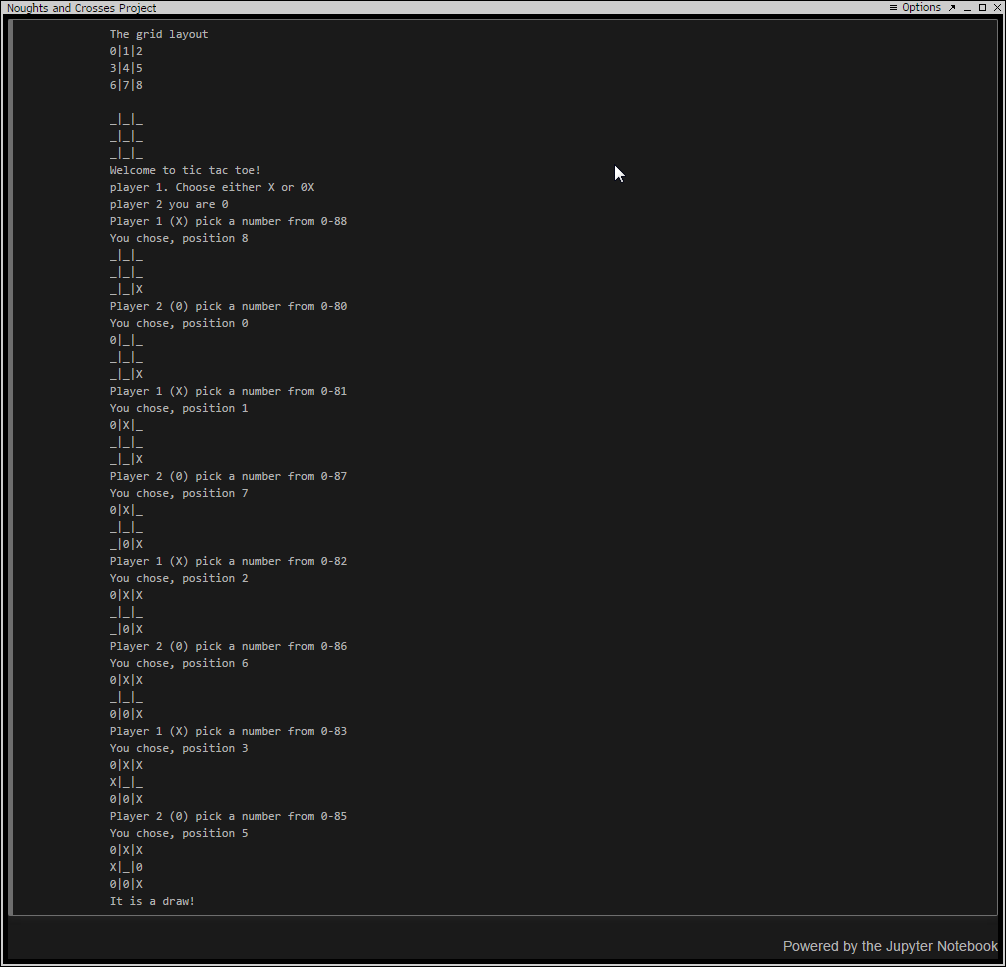


**Section 4** - This section defines all of the possible combinations for wins within the game. If the game does not find any match, then it will go to a draw if the number of goes is equal to 9

**Code:**



**Possible Outputs  
  
Output 1: For a win:**

**Output 2: For a draw:**

### Variables used in the final version of the code

In this code I would use the following variables with specific purposes:

* **board** = This variable defines the empty spaces that the user can select shown as ‘\_’. This then gets multiplied by 9 which allow for it to be indexed by python, where the user can choose a position from 0-8.
* **player1** = This is the variable which is where player 1 defines what symbol they are, either X or 0. This is the symbol in which player 1 will be playing with.
* **player2** = This is the variable that is defined by the opposite symbol that player 1 selects. This is the symbol in which player 2 will be playing with.
* **winner** = This is the variable that determines whether the loop for players to play will continue on or not, if winner is equal to 1 then the while loop will break and declare a winner, and if it is 0 it will continue on. Winner will also turn to 1 when it is a draw.
* **print\_board** = This routine prints out the board after each player chooses their move. When a player chooses a grid position, the empty space becomes replaced with the player’s marker.
* **Valid** = The variable defines the valid inputs the user can choose from to input their symbol into the grid
* **Goes** = This variable counts the number of turns both player 1 and 2 have had combined and increases by 1 after each valid input. The reason for counting the number of goes is how the program identifies when all of the grid positions have been taken up. If the number of goes is equal to 9 and there has not been any winning matches found, it means that it is a draw and the game will end
* **Chosenpositions** = This variable contains a list of all of the chosen positions of each player which gets added to after each valid input. If a player tries to choose a position where the other player has already placed their marker, the game will recognize it and

tell them that the position has already been taken and that they cannot choose it.

* **X** = This variable contains the chosen position of player 1
* **Y** = This variable contains the chosen position of player 2

# **Testing**

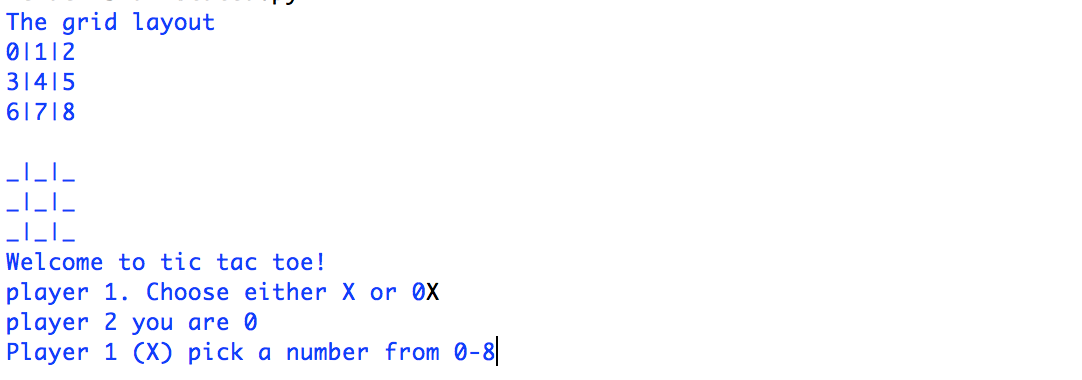
## Test plan (Before carrying out testing)

Before testing the code that I made, I created the following test plan to test my game program.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 1 | Run the program and wait for the welcome message where the user will be required to choose a symbol | X | Player 2 will be assigned the symbol 0 |  |
| 2 | Run the program and wait for the welcome message where the user will be required to choose a symbol | 0 | Player 2 will be assigned the symbol X |  |
| 3 | Run the program and wait for the welcome message where the user will choose their symbol | Z | Error |  |
| 4 | Run the game until Player X wins | X,0,1,4,2,8 | Player X wins! |  |
| 5 | Run the program until Player 0 wins | 0,0,1,4,2,8 | Player 0 wins! |  |
| 6 | Run the program and game until no one wins | X,0,1,2,3,4,5,6,7 | It is a draw |  |
| 7 | Run the program and enter the same position as the last player | X,0,0 | That position has already been taken! |  |

## Test plan results (After carrying out testing)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 1 | Run the program and wait for the welcome message where the user will be required to choose a symbol | X | Player 2 will be assigned the symbol 0 | “Player 2 you are 0” |

****Python**

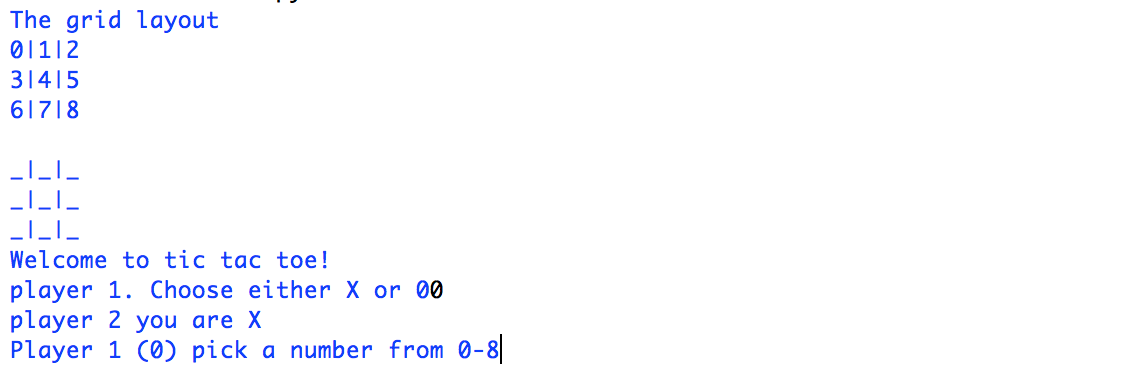


**BQUANT**

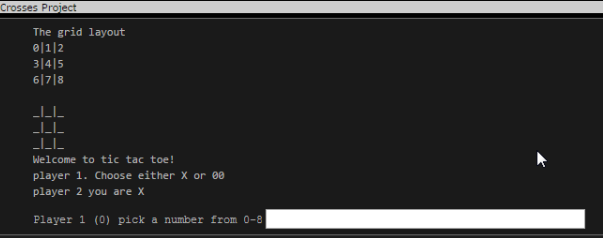
**Test Passed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 2 | Run the program and wait for the welcome message where the user will be required to choose a symbol | 0 | Player 2 will be assigned the symbol X | “Player 2 you are X” |

**Python**



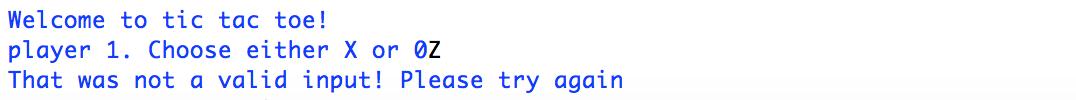


  
  
**BQUANT**

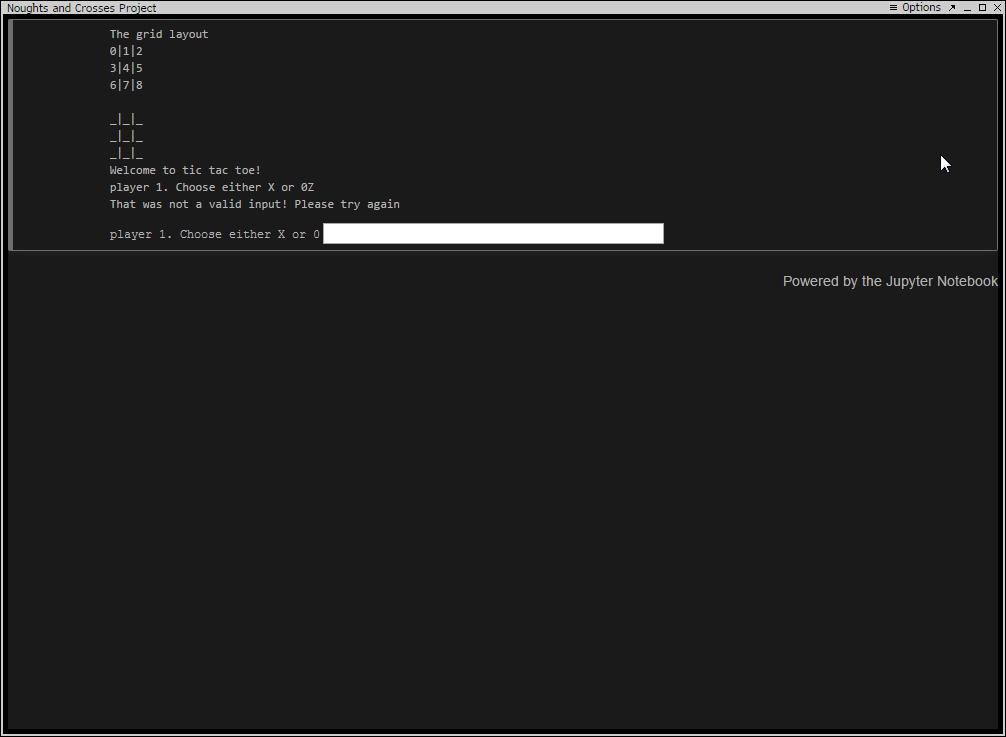
**Test Passed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 3 | Run the program and wait for the welcome message where the user will be required to choose a symbol | Z | ‘That is not a valid input!’ | An error message is outputted:   “That was not a valid input! Please try again” |

**Python**

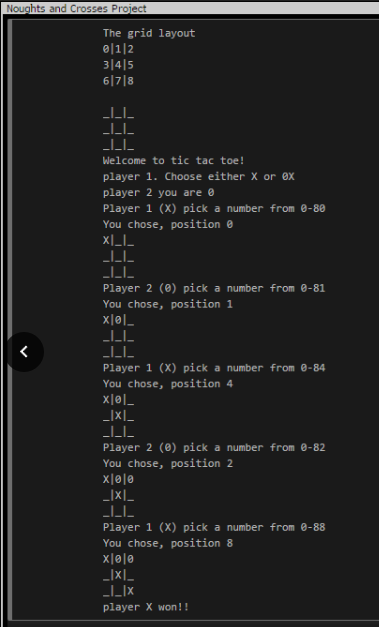
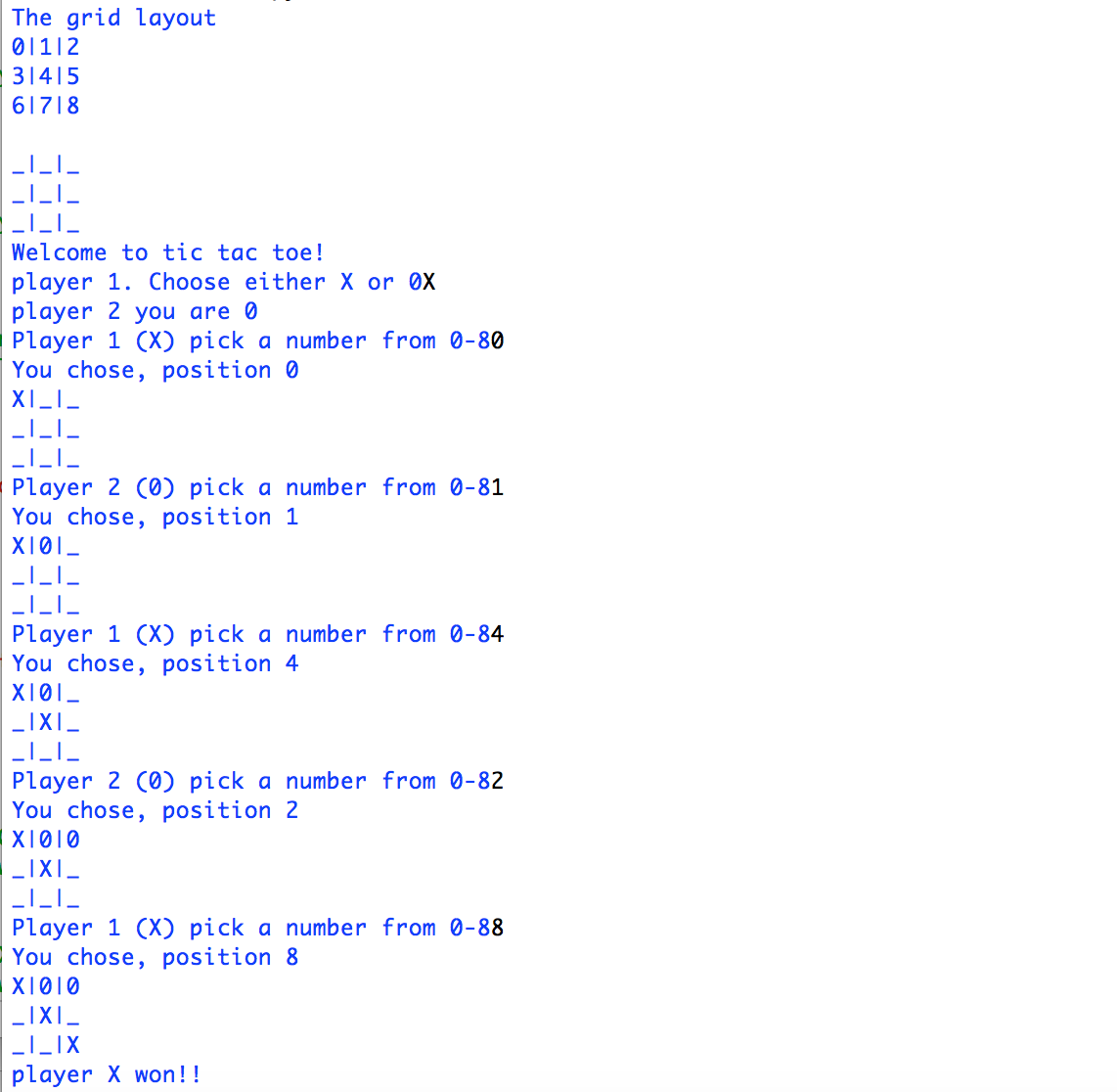




  
  
  
  
  
  
**BQUANT**

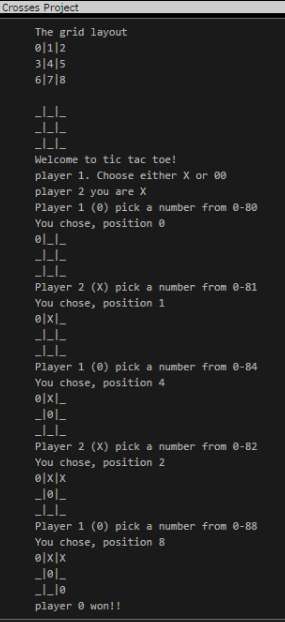
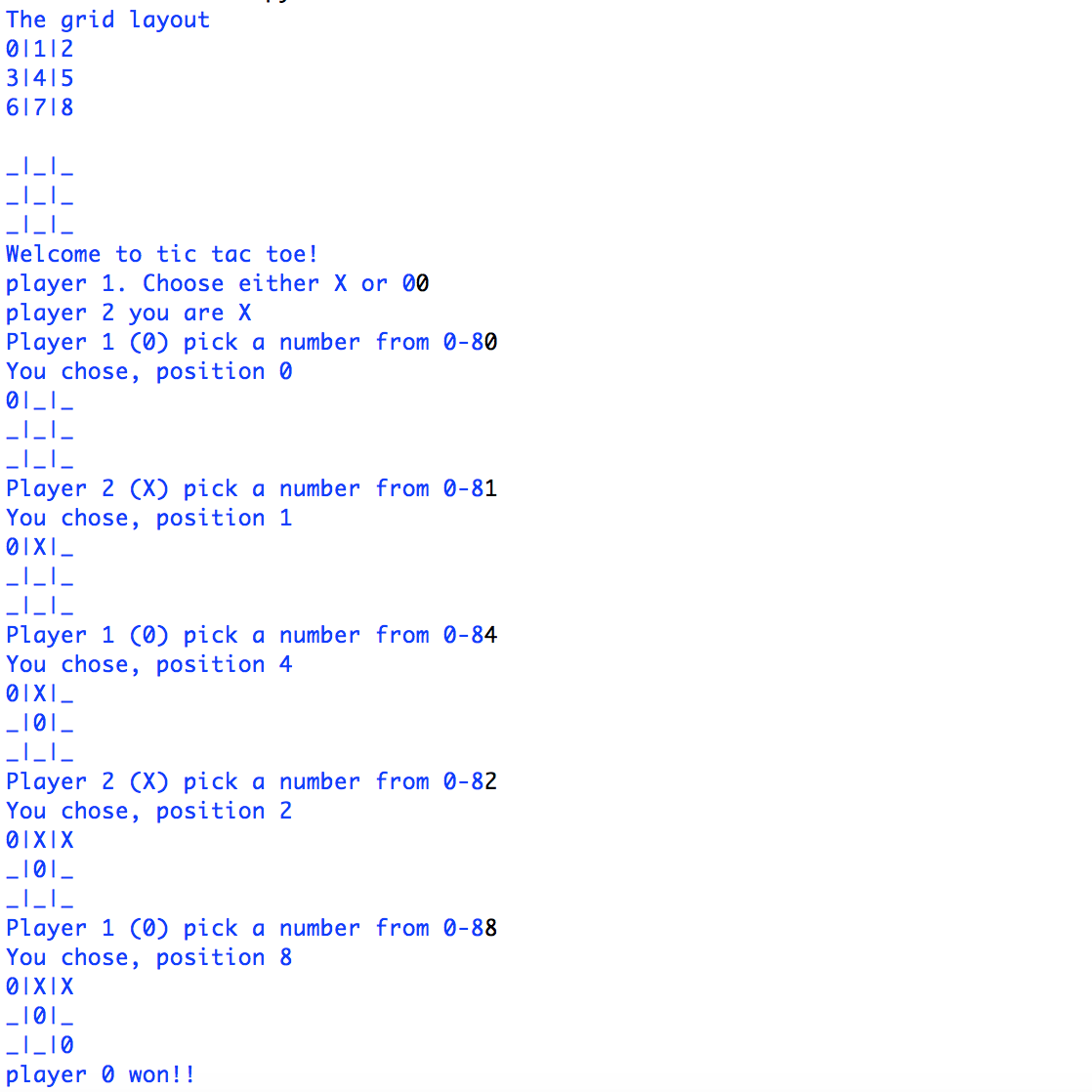
**Test Passed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 4 | Run the game until Player X wins | X,0,1,4,2,8 | Player X wins! | ‘Player X Won!!’ |

**Python**  **BQUANT**

**Test Passed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 5 | Run the program until Player 0 wins | 0,0,1,4,2,8 | Player 0 wins! | ‘Player 0 won!!’ |

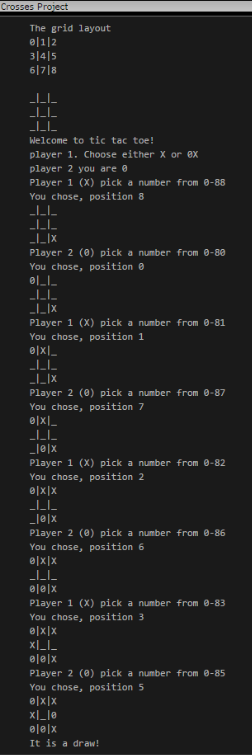
****Python BQUANT**

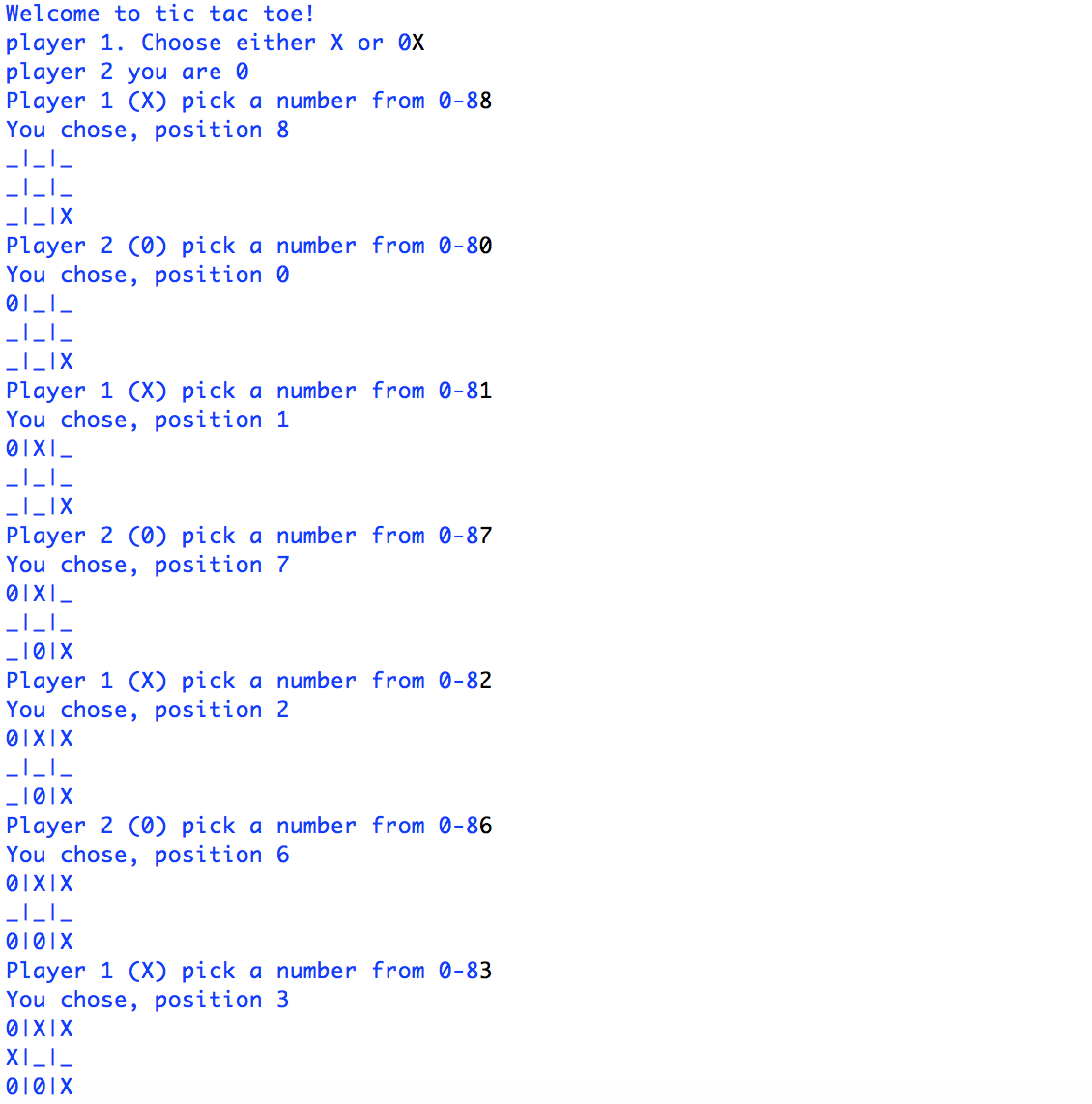
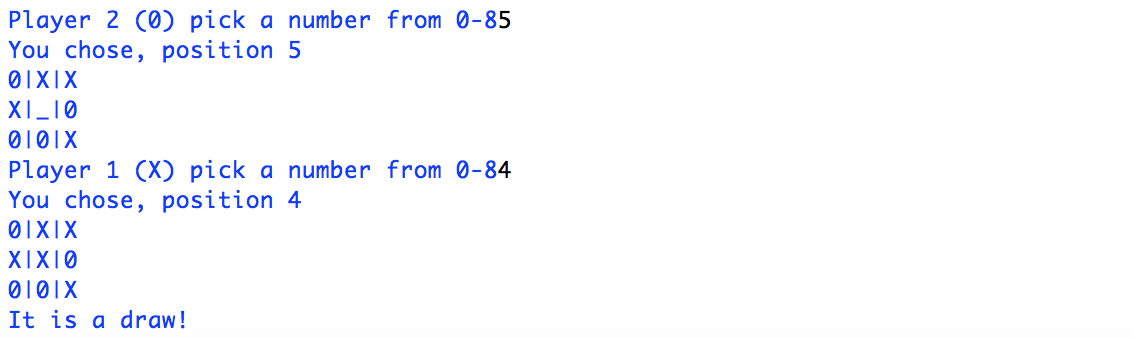


**Test Passed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 6 | Run the program and game until no one wins | X,8,0,1,7,2,6,3,5,4 | It is a draw | A message is outputted stating: ‘It is a draw!’ |

## 







**Test Passed**

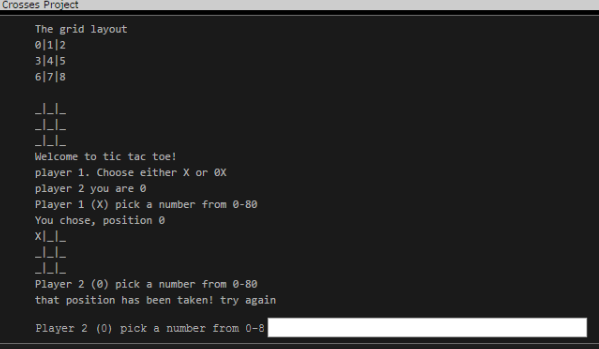
**Python**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Expected input | Expected output | Actual output |
| 7 | Run the program and enter the same position as the last player | X,0,0 | That position has already been taken! | The code outputted ‘That position has been taken! Try again |

## 



**BQUANT**



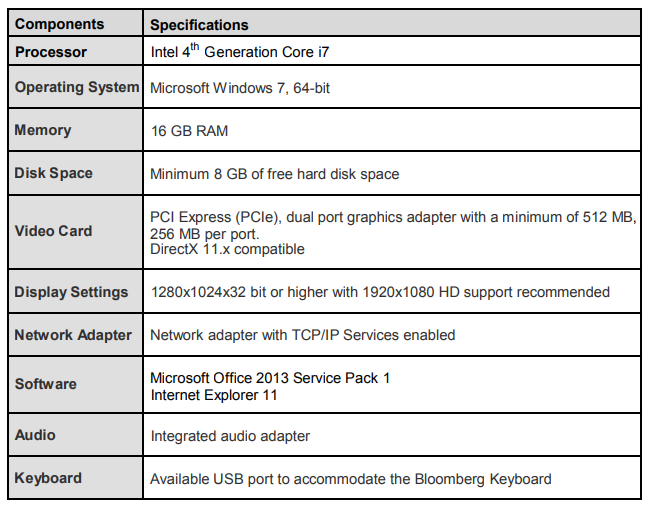


**Test Passed**

# **Documentation**

## Minimum Hardware and Software requirements

To run the new product, BQUANT - every employee and client will need to have access to a Bloomberg Terminal.

**Bloomberg – Recommended Requirements to launch the Bloomberg Terminal**

**Python – Minimum Requirements for Python**

As well as BQUANT. employees are also encouraged to use python when away from the office since both programming platforms use the same methodology. As stated on the python website, below are the minimum requirements needed to run python.

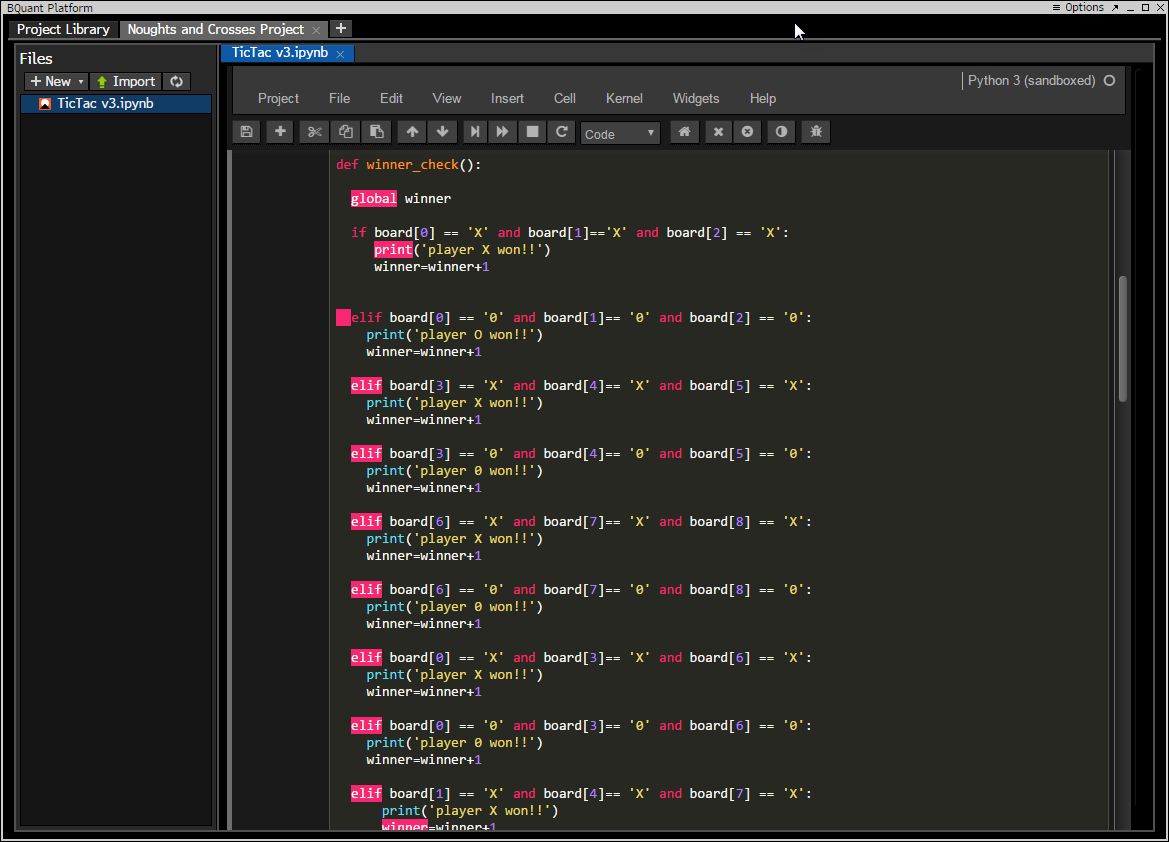
* Processors: Intel Atom® processor or Intel® Core™ i3 processor
* Disk space: 1 GB
* Operating systems: Windows\* 7 or later, macOS, and Linux
* Python\* versions: 2.7.X, 3.6.X
* Included development tools: conda\*, conda-env, Jupyter Notebook\* (IPython)
* Compatible tools: Microsoft Visual Studio\*, PyCharm\*
* Included Python packages: NumPy, SciPy, scikit-learn\*, pandas, Matplotlib, Numba\*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP\*, and other

# **Reflection of Code**

## Evaluation of the results

Overall my code that I created has been able to pass all of the tests that I had created. However, it does not mean that it is perfect since my code consists of 219 lines and if I was given more time, I am convinced that this could be a lot shorter. Here are some potential ways that the 3rd version of my code could be cut down:

1. **Develop an efficient way of finding a winner**

­­­Currently the only way to find a winner is to manually define what a win is by checking the grid number positions. This is the longest part of the code and is repeated constantly, as seen in the screenshot below. The code of winner combinations consists 73 lines and could be significantly cut down

1. **Develop a way where code is not repeated for each player**

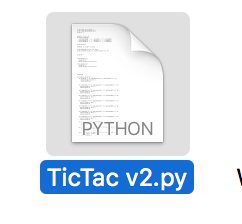
For each player to have their turn, my code is repeated twice and is exactly the same, except for the player variable. Player 1 is variable X and player 2 is variable Y. To improve the code, I could find a way to only have this portion of code repeated once and have each player alternate when it is their turn. In total, the whole section of code for one player consists of 46 lines and therefore if I was given more time, I am convinced that this could be more efficient.



**Deployment**

When the game was finished and the BQUANT platform was ready to be released, the deployment stage began. To start off with, I did many presentations to small groups of employees because I thought it would more effective this way to listen to their feedback, and take on board their opinions. I was able to highlight all the main features of the BQUANT platform in general and also show them how to create a game like mine. The response was very positive that my manager gave me her approval to show the whole department the game in order to familiarize themselves with the BQUANT platform, as well as a show a fun, productive way to entice them to learn more about the product in their own time. Consequently, a mass email was sent out to the whole of the Analytics department, in which a link to multiple games on BQUANT was included too, but also a copy of the user guide that briefly explains how to run and play the game, as well as useful troubleshooting steps. A copy of the userguide can be seen on the next page.

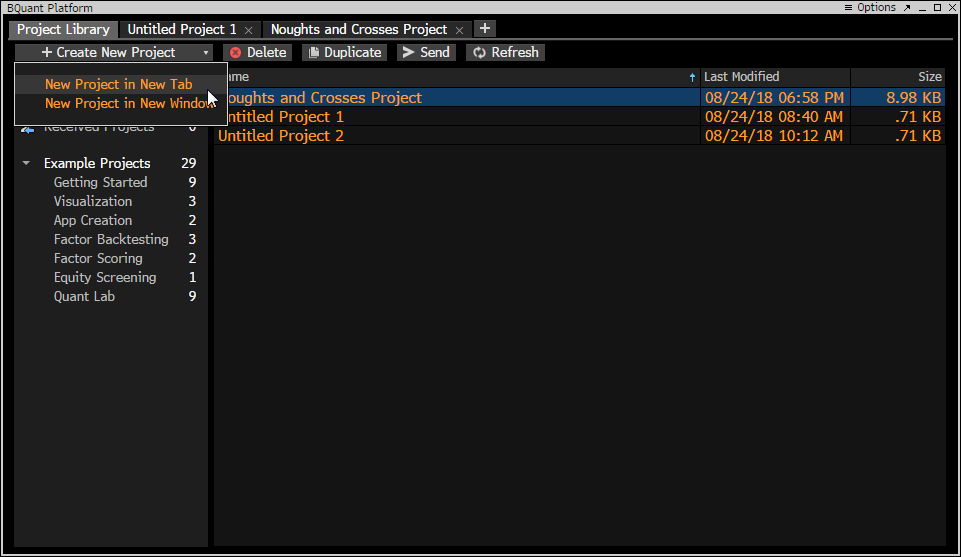
**User Guide**

****How to prepare to play Noughts and Crosses? (From Python)  
  
Make sure you have copied TicTac v3.py file to your chosen folder/directory of your hard drive or USB memory stick.   
In order to prepare this game or open this game in Python, follow the steps:  
• From desktop or start menu, select python 3.6 (or IDLE Python 3.6 32- bit)   
• Click the File menu and Open option and select Tic Tac v2.py file from your chosen directory/folder  
• Now you can see source code of the game program.

How to prepare to play Noughts and Crosses? (From BQUANT on Bloomberg)

Make sure you have copied TicTac v3.ipynb file to your chosen folder/directory of your hard drive or USB memory stick.

From within the Bloomberg terminal, type BQNT <GO> to launch the BQUANT project platform. When it first opens, create a new project in a new tab and then you can import the game code into the project to run



Once loaded, you can import the TicTac v3.pynb file and save it within the project space. You are now ready to run the game!



## How to run Noughts and Crosses? (From Python)

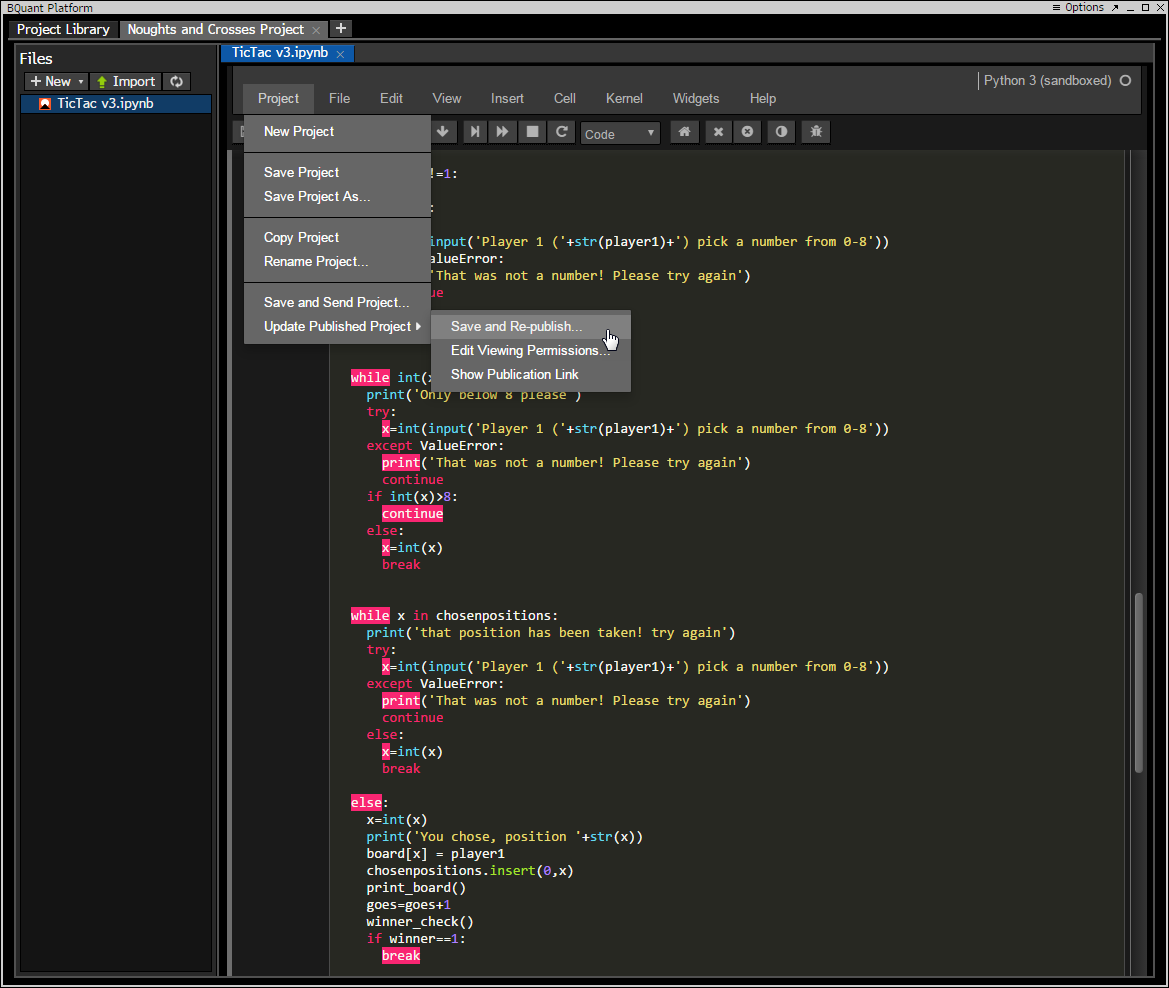
When your game program’s source code is available on the screen, press F5 from keyboard or select Run Module/F5 from Run menu to run the game program.



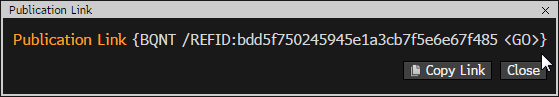
Python may ask you that the source must be saved. Click OK and the program will be opened

## How to run Noughts and Crosses? (From BQUANT)

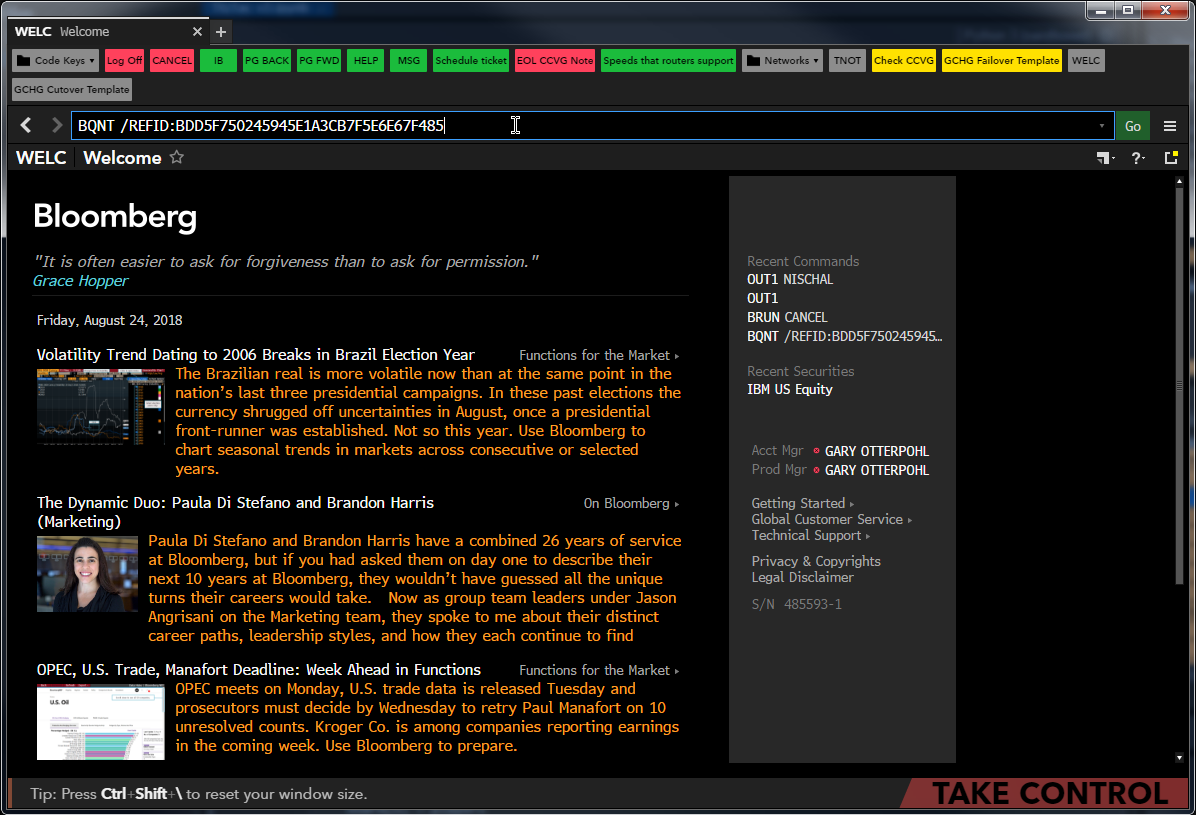
Within the BQUANT platform when you are on the game project file, go to Project -> Update Published Project -> Save a Re-publish.



This will allow you to run the project. The Bloomberg terminal will then inform you of the link of the game to run it. Copy the link and paste it into a terminal window to run.



Once you paste it into the terminal window, the game screen will load with the python game.

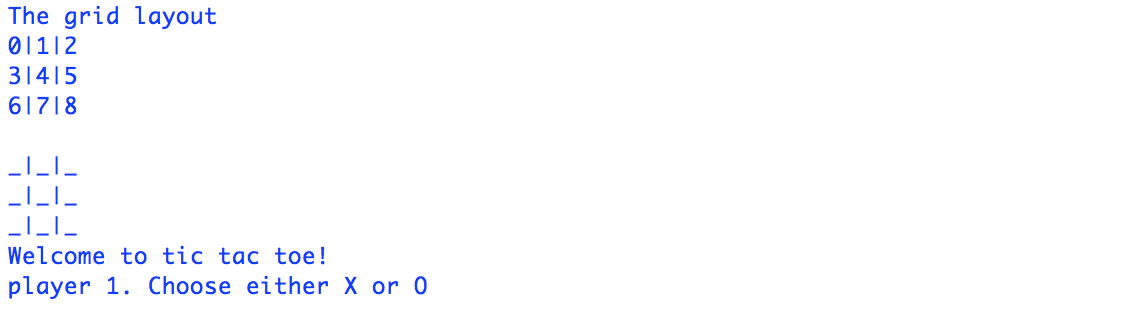




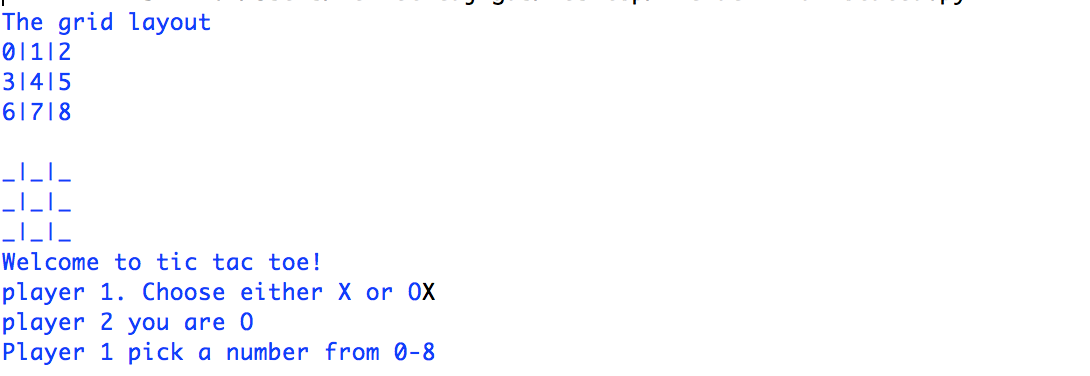
You are now ready to play the game on the Bloomberg terminal!

## How to play the game (From Python)

1. Once the game is loaded, you will be greeted by the welcome text onscreen like below

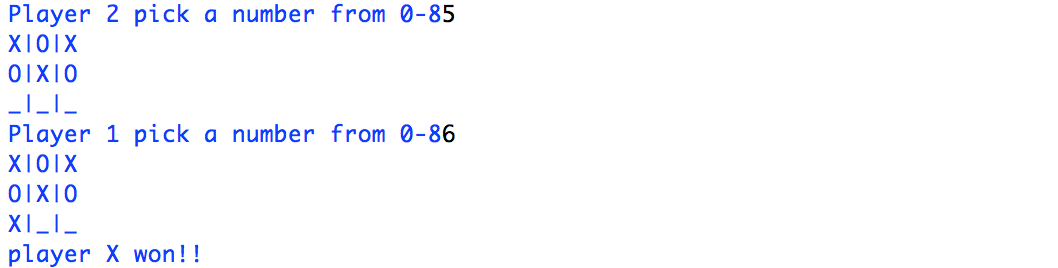


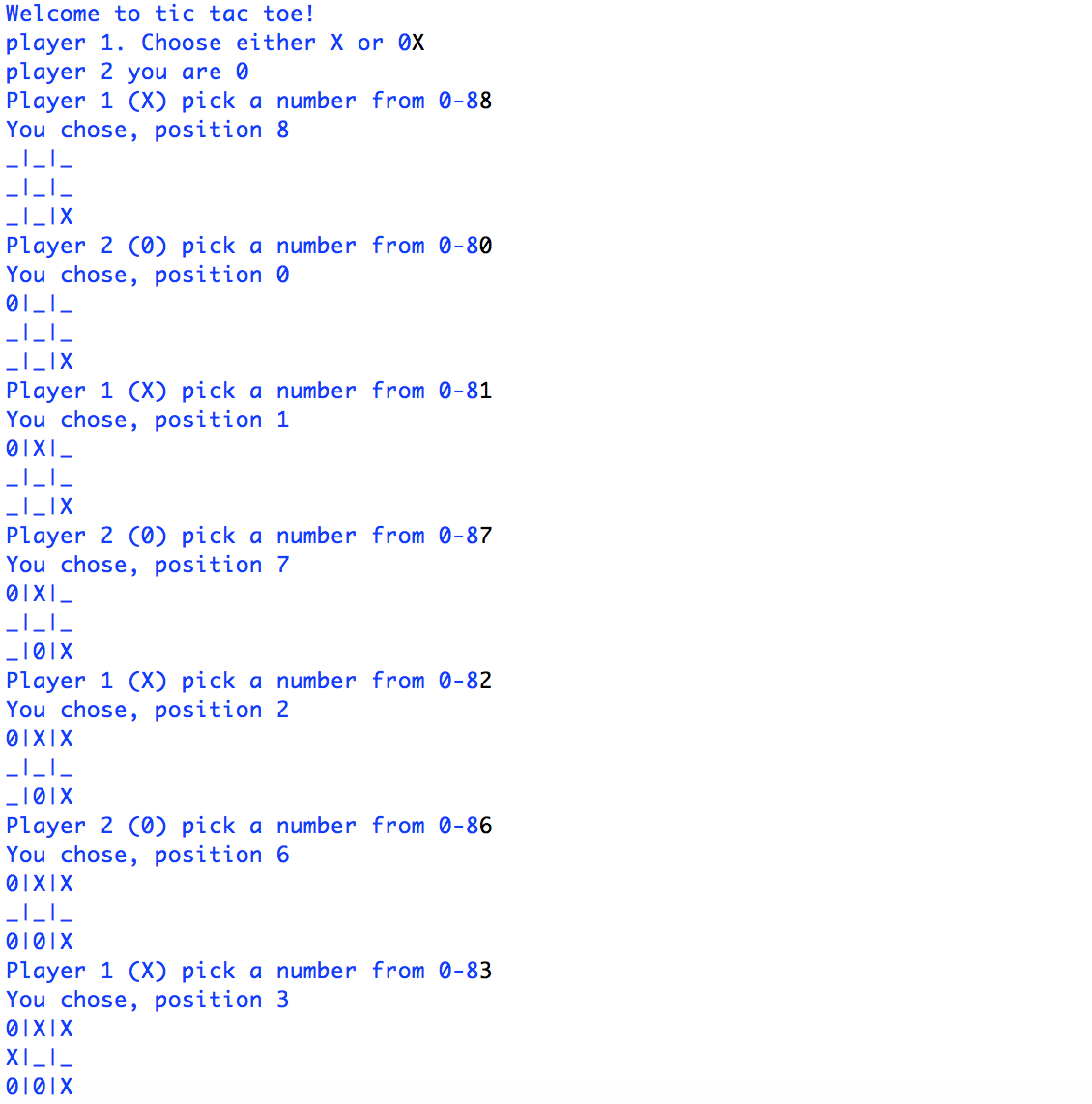
1. Player 1 should input either X or 0, and once entered player 2’s symbol will be determined



1. The grid layout is seen above. Player 1 will have to first turn to input a number from 0 to 8 and their symbol will be outputted on the grid. Player 2 will have their turn and then this cycle continues, until a winner is found.

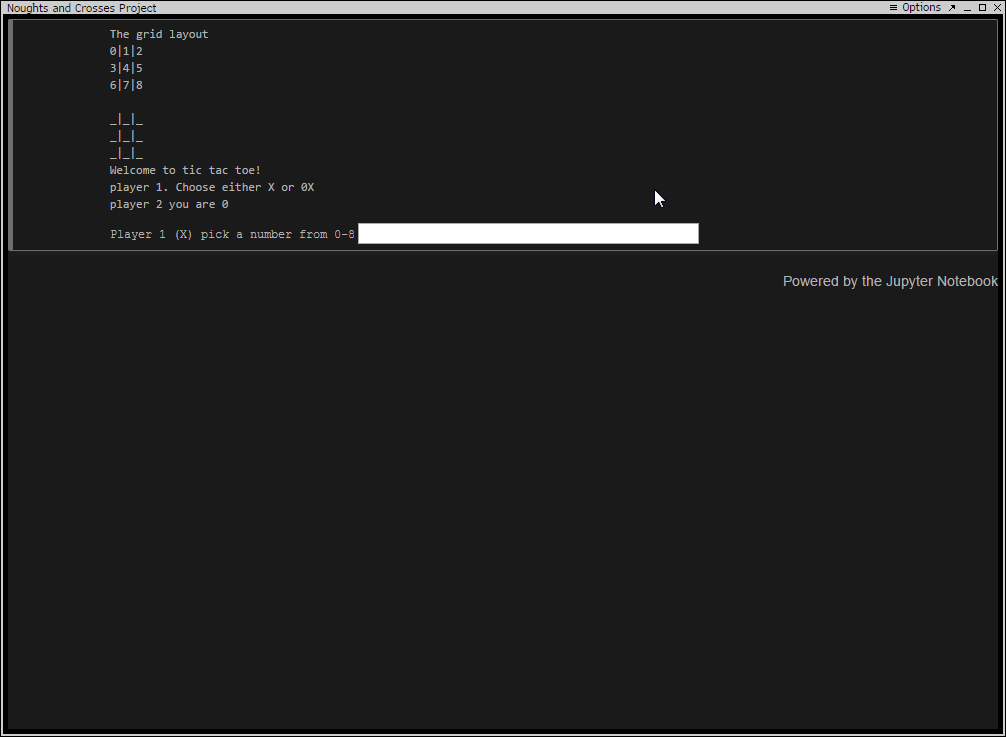


1. When the same symbol is displayed 3 times in a row, a winner is found.

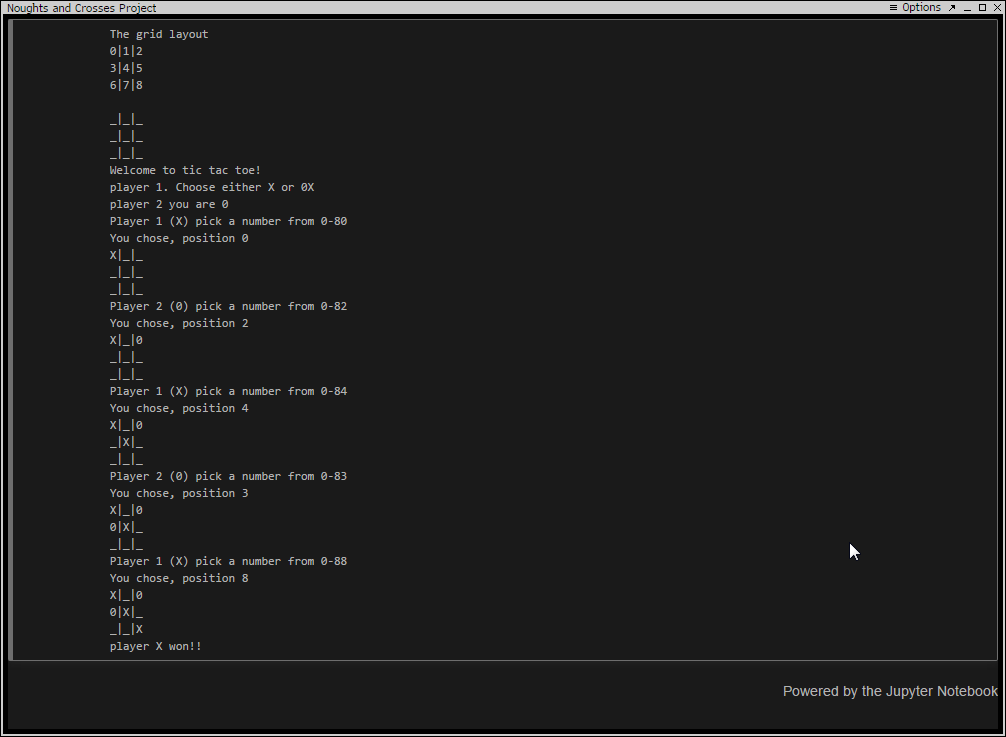
**However, if all symbols have been inputted on the graph, and there have not been a match of 3 symbols in a row, then it is a draw as seen on the right.

## How to play Noughts and Crosses? (From BQUANT)

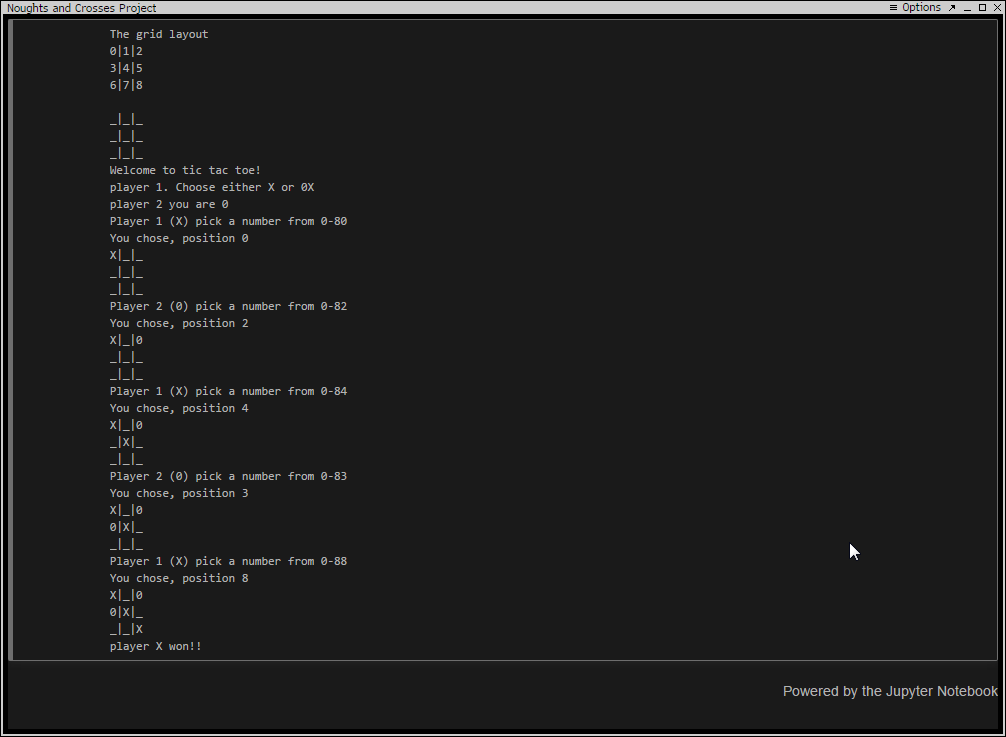
1. Once the game is loaded, you will be greeted by the welcome text onscreen like below

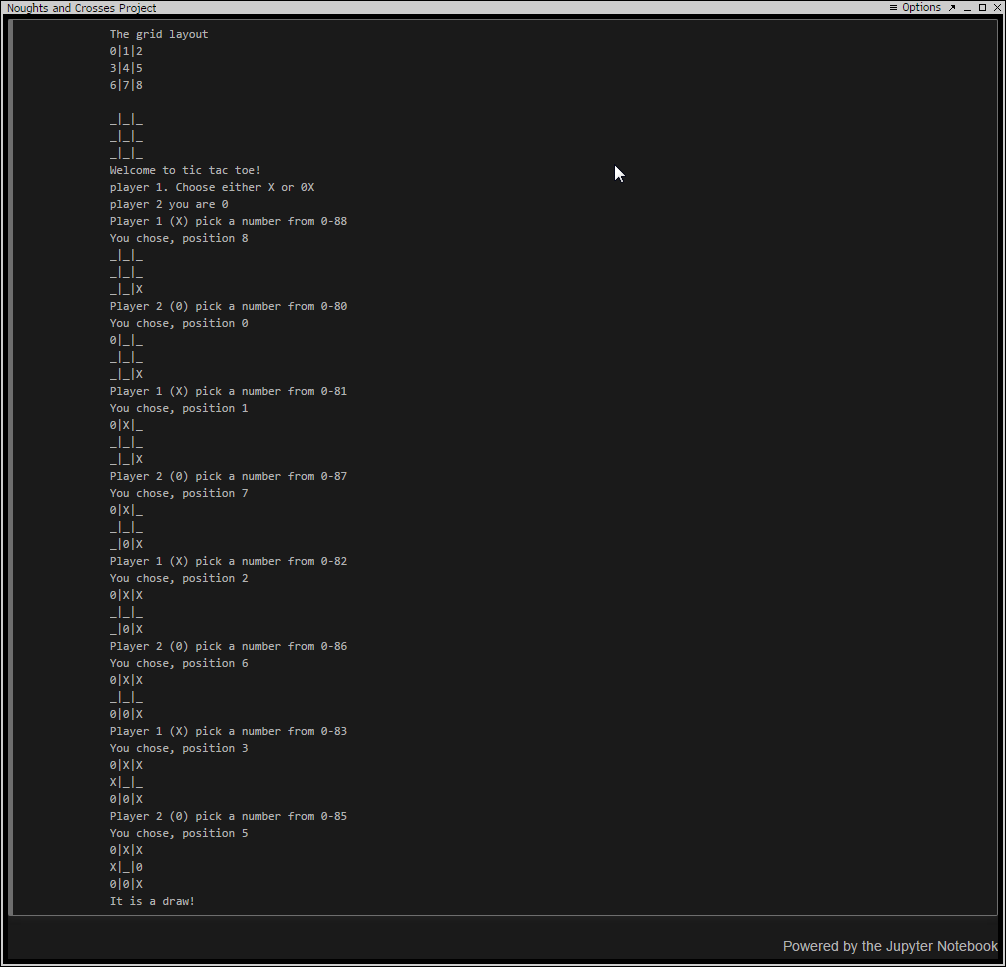


1. Player 1 should input either X or 0, and once entered player 2’s symbol will be determined



1. When the same symbol is entered 3 times in a row, a winner is found.



However, if all spaces within the table have been taken, and there have not been a match of three symbols in a row, then the game will determine that it is a draw. (See below)

Troubleshooting **1. If you get an error to play this game, what should you do?**   
  
If you use recommended Python 3.6, you should not get any error message to run this game. If you use any other version of Python, you may get error message or may not be able to play this game. If you still use Python 3.6 version and get error to run this game, open and run original backup copy. Assume that you have duplicate copy of this game as backup in your hard drive

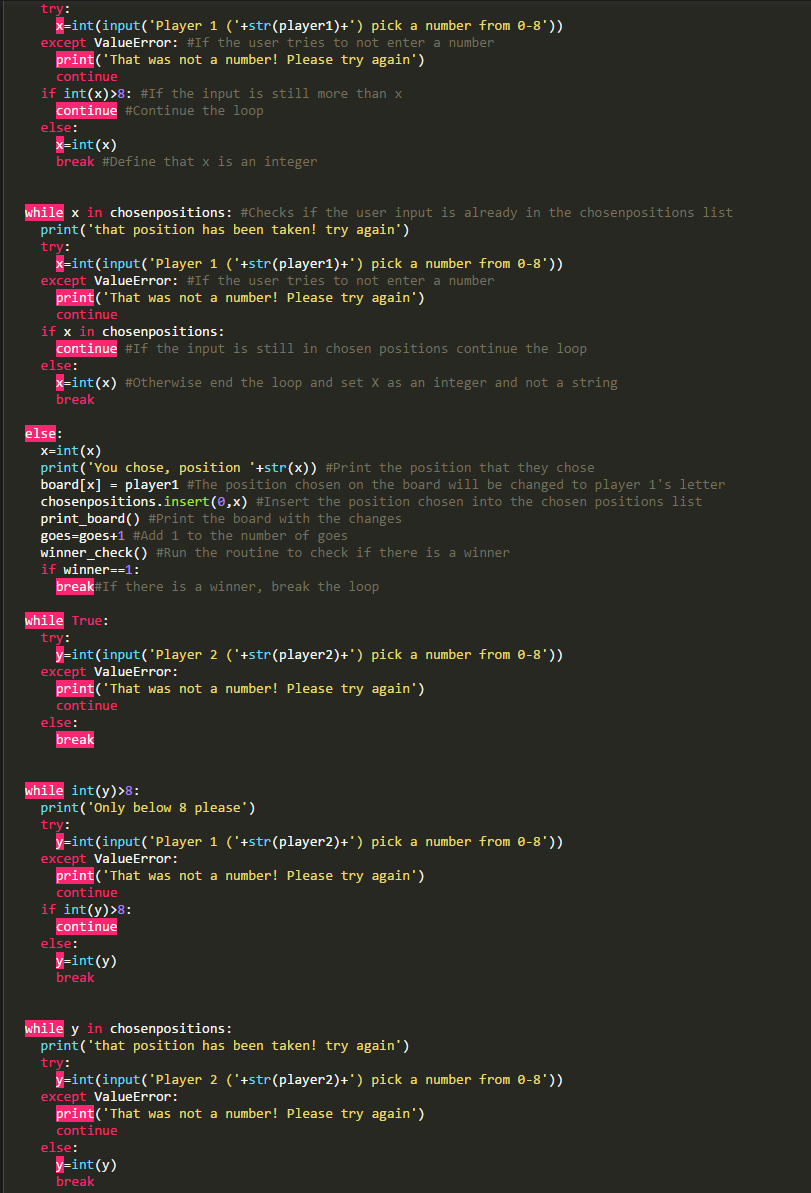
**2. Do you need to save this game after playing?**

After playing this game, you should not save this game. Somehow if this program code is changed or modified, during closing this game, a message would come on the screen to save the program file and click on “No” button to close program file without saving

**3. How to update/modify this game program?**   
  
You should be familiar with Python program and its syntax. For further information, visit www.python.org and https://www.w3schools.com/python/

## Full Code with Comments on BQUANT

**[Part 1]**

**[Part 2]****[Part 3]**

**[Part 4]**

