

Password project

**SD**



September 19, 2018

Student Name : Jason Fernandes

Tutor : Vy Tran   
Line Manager: Katie Hirchovits

Contents

[**Introduction** 2](#_Toc525122497)

[Project Brief 2](#_Toc525122498)

[Purpose of the Document 2](#_Toc525122499)

[Scope 2](#_Toc525122500)

[Who with? 2](#_Toc525122501)

[**Project Overview** 2](#_Toc525122502)

[Solution Overview 2](#_Toc525122503)

[Requirements 3](#_Toc525122504)

[Process Map 3](#_Toc525122505)

[**Development of the Program** 4](#_Toc525122506)

[Password generator 5](#_Toc525122507)

[Section 1 5](#_Toc525122508)

[Section 2 5](#_Toc525122509)

[Section 3 5](#_Toc525122510)

[Section 4 6](#_Toc525122511)

[Password checker 6](#_Toc525122512)

[Section 1 7](#_Toc525122513)

[Section 2 7](#_Toc525122514)

[Section 3: 7](#_Toc525122515)

[Variables used in the final version of the code 8](#_Toc525122516)

[**Testing** 9](#_Toc525122517)

[Test plan (Before carrying out testing) 9](#_Toc525122518)

[Test plan results (After carrying out testing) 10](#_Toc525122519)

[**Documentation** 10](#_Toc525122520)

[Minimum Hardware and Software requirements 10](#_Toc525122521)

[**Reflection of Code** 11](#_Toc525122522)

[Evaluation of the results 11](#_Toc525122523)

[**Deployment** 11](#_Toc525122524)

[**User Guide** 12](#_Toc525122525)

[How to run? (From Python) 12](#_Toc525122526)

[How to run? (From BQUANT) 13](#_Toc525122527)

[How to play the game (From Python) 14](#_Toc525122528)

# **Introduction**

Project Brief

## Purpose of the Document

## Scope

## Who with?

# **Project Overview**

## Solution Overview

* **How the game should work**

## Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req number | Functional Requirements | Inputs required | Processing Required | Outputs |

## Process Map

# **Development of the Program**

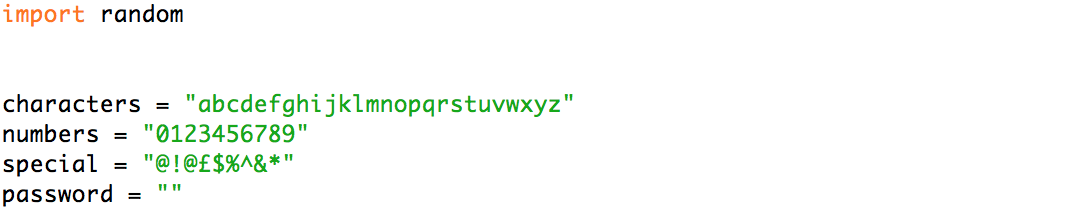
The password generator code came in two parts. The first part was the actual password generator where it would generate a password from a combination of numbers, special characters, upper- and lower-case characters. The second part was the password strength checker, where it would check a password based on a specific set of criteria.

## Password generator

The password generator code was quite simple. I will split it down into 4 sections to explain how it works, along with the output of the code.

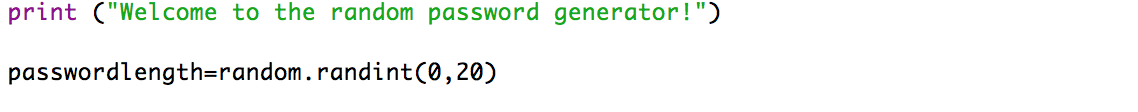
### Section 1

This is the first part of the code. I used the random library, as it would allow me to randomly select from a list or numbers when a password was being generated. Also, in this section are the variables where the random library would be selecting characters from. Password is blank variable as nothing has been added to it yet. There is no output from this section of code.



### Section 2

This part of the code prints out a message to the user welcoming them to the password generator. A new variable is also introduced here called passwordlength. The variable uses the random library function to randomly select a number between 0 and 20. This number forms the length of what a password will be generated to. It should be noted though, that the passwordlength variable does not determine the exact length, but approximately how long the password will be around. This is due to there being 4 different components that make up the password (Upper letter, lower letter, special character, number) and as a result, the length will be a multiple of 4.

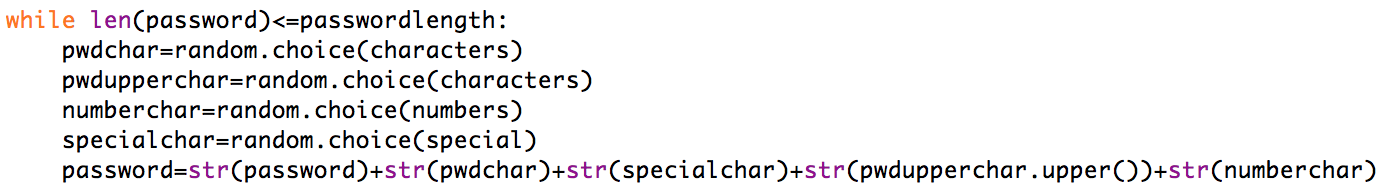


Output:



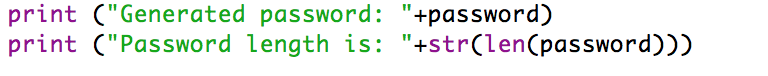
### Section 3

This section of the code is where each component within the password is selected randomly by python and is put together into the empty password variable. Using the .choice function allows the code to select a single character from the variable defined within the brackets. As seen earlier, a choice is made from each of the variables: characters, numbers and special. Once the code has selected a character for each variable pwdchar, pwdupperchar, numberchar and specialchar, they are all converted to string format and added together. In the case of pwdupperchar, it appears to use the same variable ‘characters’ as the lowerchase variable ‘pwdchar’ to get a character, but this is to ensure that the character selected is unique. It is also converted to an uppercase using the .upper() function. This continues to loop until the password generated is longer than the passwordlength number selected by python between 0 and 20. The output is seen in section 4.

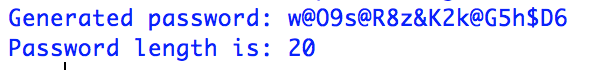


### Section 4

This section of the code outputs the results of the process in section 3 when the password that has been generated is longer than the number generated previously.



Output:



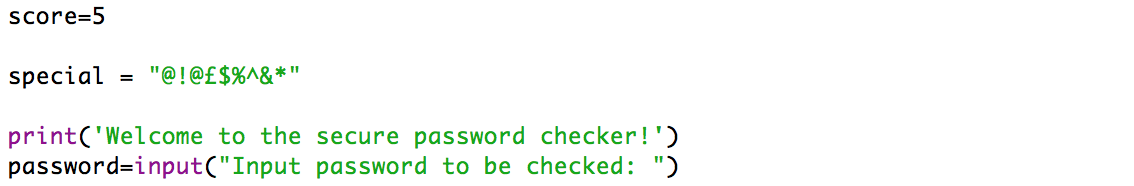
## Password checker

The password checker requests for the user to input a password, and the code checks if it meets the criteia.

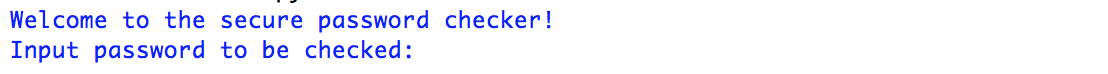
A score is then given to the user out of 5 and it also shows them what they did not pass on.

### Section 1

For the first section of the password checker, I defined the default score of 5. Later on, in the code, if the password that the user enters does not pass a criteria, a point is deducted from this score. Have also define special characters under the ‘special’ variable as python does not have an inbuilt function to check for special characters. This section is also where the user inputs the password that they want to check.



Output:



### Section 2

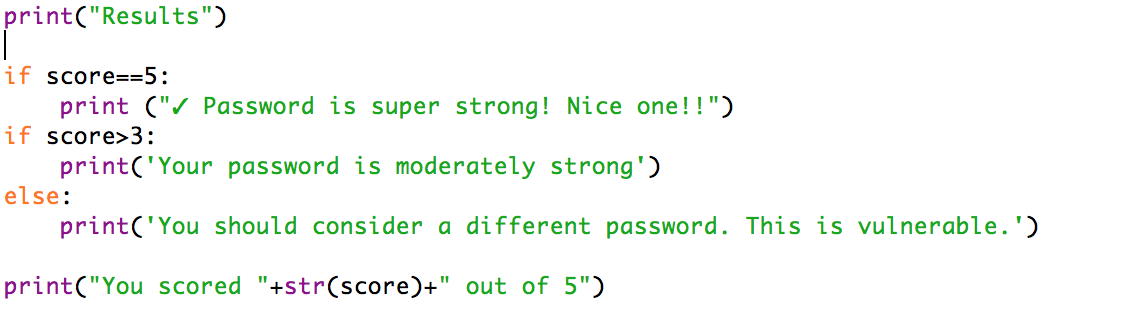
In this section, it is where the password to be checked if it meets the following criteria:

* Length is more than 6
* If there is a digit in the password
* If there is an upper-case character in the password
* If there is a lower-case character in the password
* If there is a special character in the password

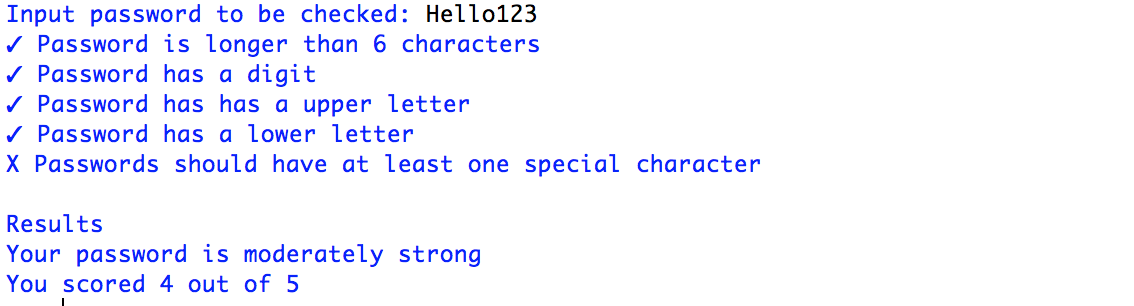
If the criteria is not met, then the score is reduced by 1.

### Section 3:

The final section of the code provides an output of the results. I created 3 thresholds, strong, moderate and weak. As seen in the code below, a perfect score of 5 means the password is strong, 3 and above means that it is moderate and below that is weak.



Output:



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

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

Password generator:

**Characters** = Defines a set of characters that python should select from

**Numbers** = Defines a set of numbers that python should select from

**Special** = Defines a set of special characters that python should select from

**Password** = The variable that will hold the newly generated password

**Pwdchar** = The variable that holds a randomly selected character from the characters variable

**Pwdupperchar** = The variable that holds another randomly selected character from that characters variable and will be converted to uppercase once selected

**Numberchar** = The variable that holds a randomly selected number from the numbers variable

**Specialchar** = The variable that holds a randomly selected character from the special variable

Password checker:

**Score** = The variable that defines and holds the default score in the checker

**Special** = Defines a set of special characters for python to reference from

**Password** = The variable that contains the password that the user inputs

# **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.

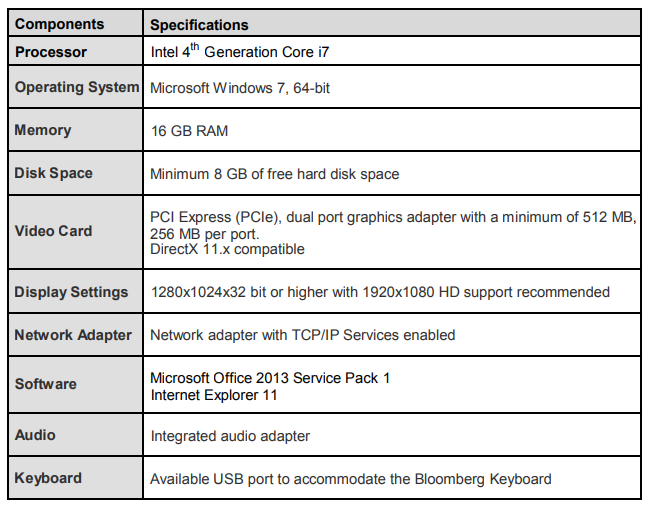
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |

## Test plan results (After carrying out testing)

# **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**

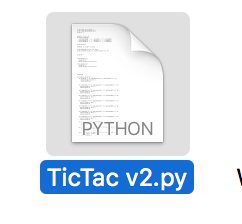
As well as BQUANT. employees are also encouraged to use python when away from the office since both programming platforms use the same methodology. As stated on the python website, below are the minimum requirements needed to run python.

* 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

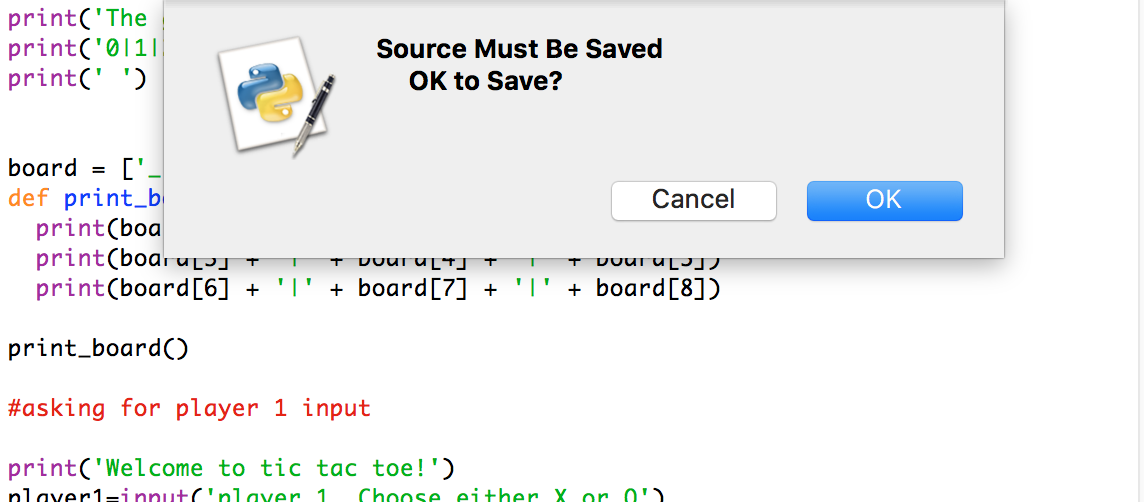
**Deployment**

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 run? (From Python)

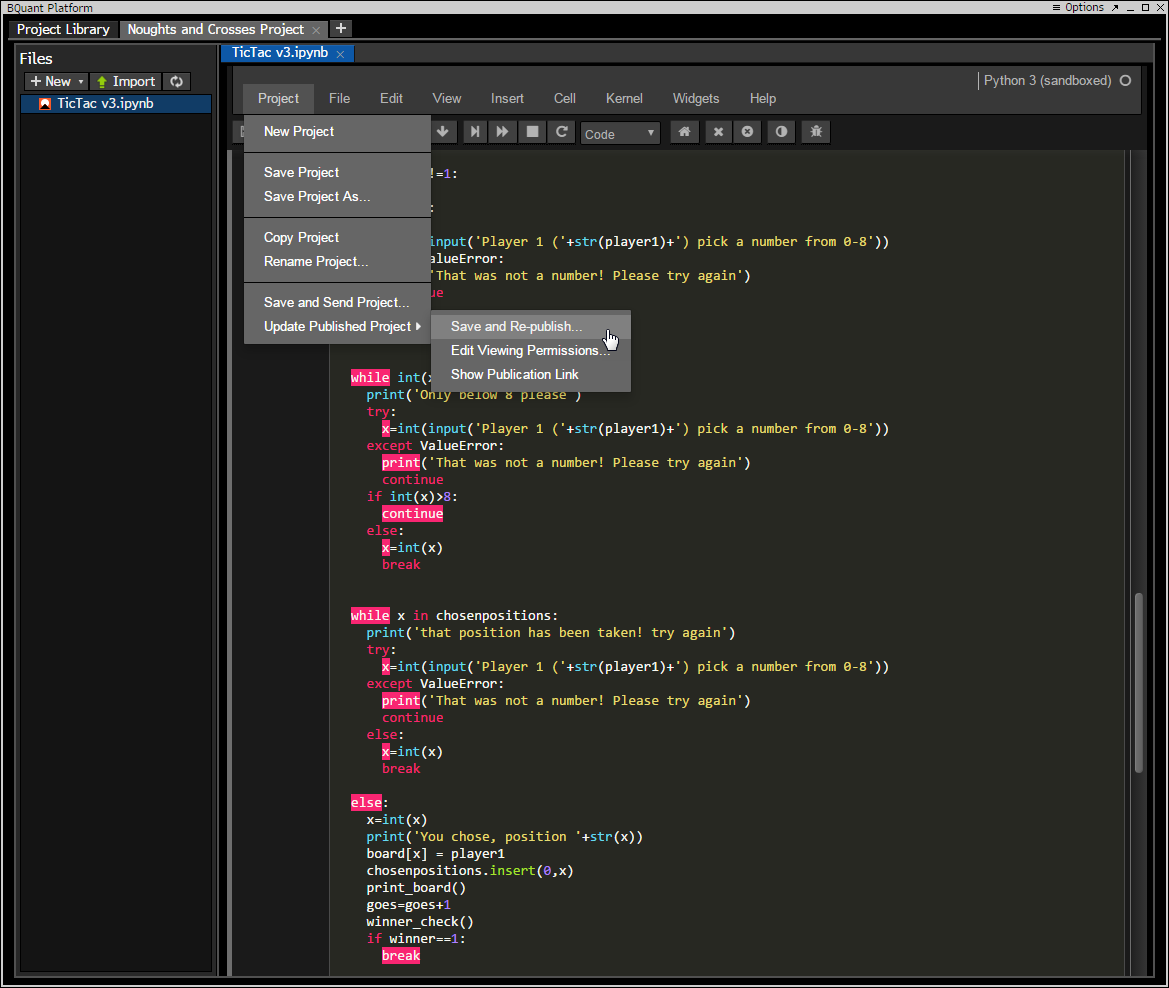
When your game program’s source code is available on the screen, press F5 from keyboard or select Run Module/F5 from Run menu to run the game program.



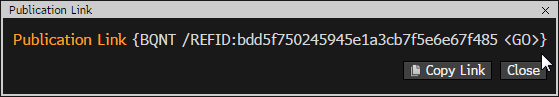
Python may ask you that the source must be saved. Click OK and the program will be opened

## How to run? (From BQUANT)

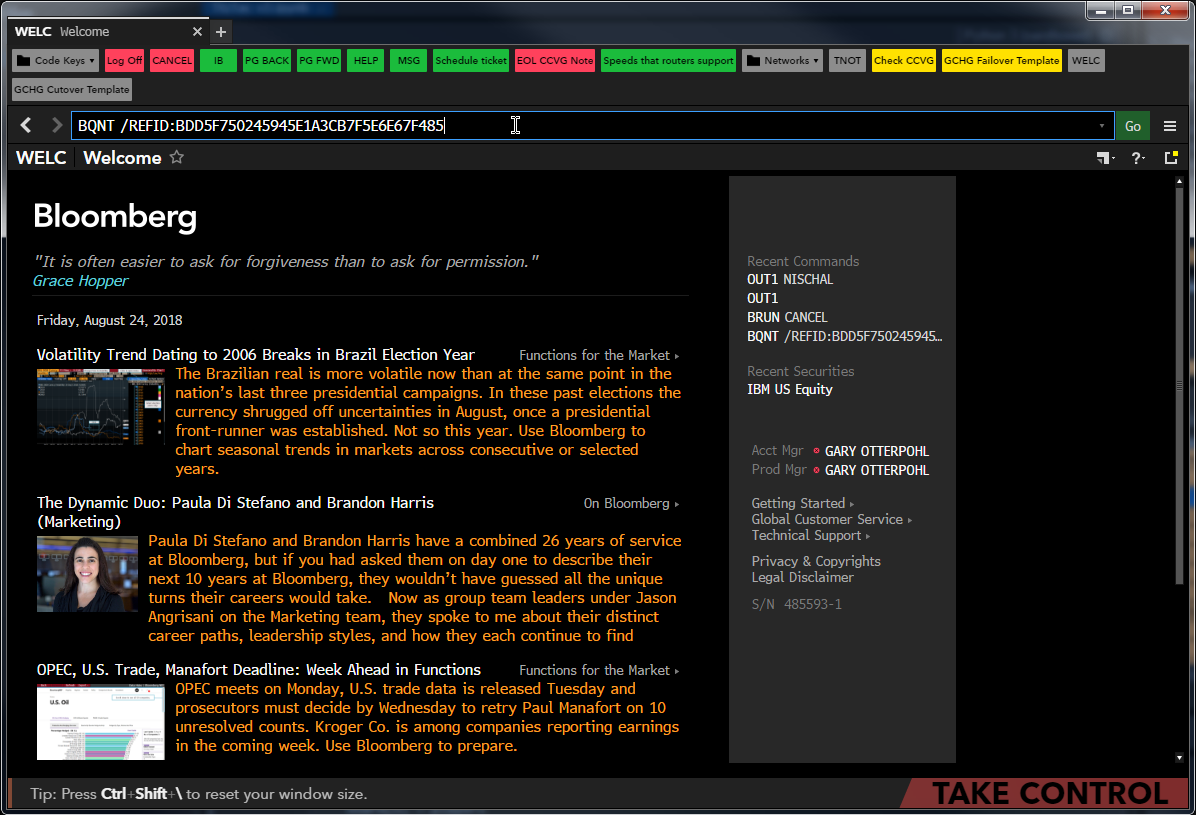
Within the BQUANT platform when you are on the game project file, go to Project -> Update Published Project -> Save a Re-publish.

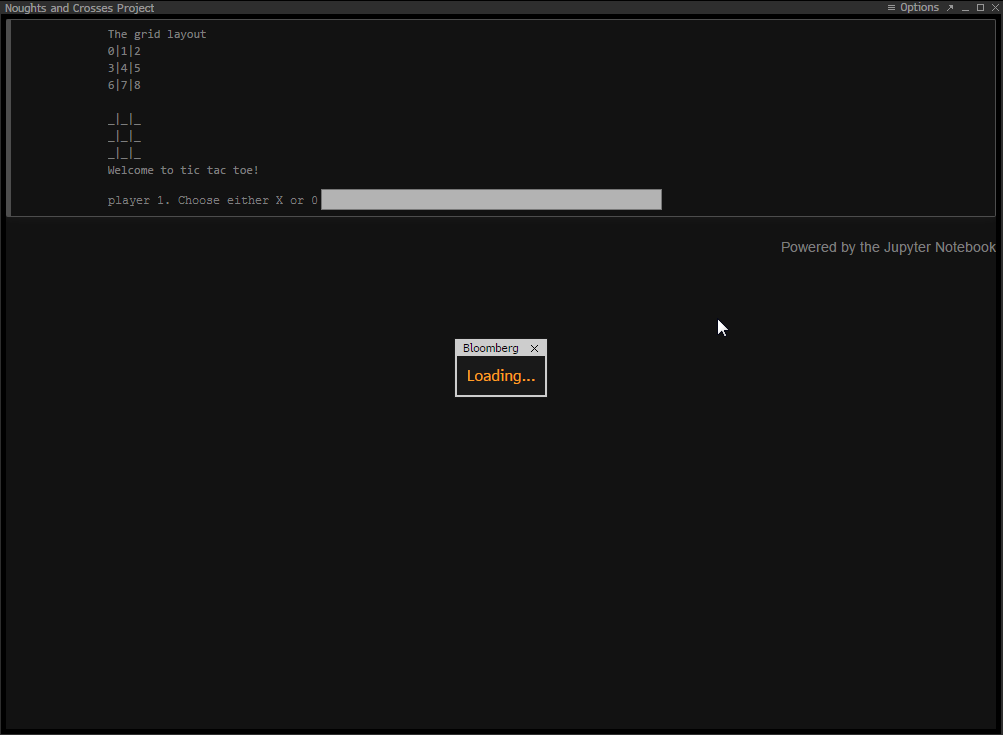


This will allow you to run the project. The Bloomberg terminal will then inform you of the link of the game to run it. Copy the link and paste it into a terminal window to run.



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)