

PASSWORD SECURITY PROJECT

**This project is based around password security, since developers/line managers have been discussing a way in which to integrate a password strength check into Bloomberg, so all accounts will have a level of complex passwords which consequently means it will be harder for non-authorised users to access the data on the Bloomberg Terminal.**



June 29, 2018

Student Name : Jason Fernandes

Tutor : Vy Tran   
Line Manager: Katie Hirchovits

Contents

[**Introduction** 2](#_Toc525152574)

[Project Brief 2](#_Toc525152575)

[Purpose of the Document 3](#_Toc525152576)

[Scope and Who with ? 3](#_Toc525152577)

[**Project Overview** 4](#_Toc525152578)

[Solution Overview 4](#_Toc525152579)

[ Conditional Formatting 5](#_Toc525152580)

[ Data Visualisation 6](#_Toc525152581)

[ Grouping Securities 8](#_Toc525152582)

[ Ease of Writing Notes 10](#_Toc525152583)

[Operational Requirements 11](#_Toc525152584)

[Issues 13](#_Toc525152585)

[Risks 13](#_Toc525152586)

[**Testing**    
Test plan (Before carrying out testing) 14](#_Toc525152587)

[Test plan results (After carrying out testing) 16](#_Toc525152588)

[**Documentation** 24](#_Toc525152600)

[Minimum Hardware and Software requirements 24](#_Toc525152601)

[**Deployment** 25](#_Toc525152602)

[**Reflection** 25](#_Toc525152603)

# **Introduction**

Project Brief  
My manager: Katie Hirchovits [Analytics Team Leader] and the programmers have assigned me a project that would focus on creating a password security check, in order to keep data secure on the terminal. They have said that they would like this password check to be integrated into the release of a new product “BQUANT”. Therefore this project would entail planning, designing and documenting a way in which to make the launch of BQUANT known to other employees, as well as ensuring all employees/clients have a strong password when logging on to the terminal. . As this task deals with keeping all the data within the bloomberg terminal confidential and secure – This program will be integrated onto the bloomberg software so that others are able to choose a password, which is essentially hard to guess/hack.

## Purpose of the Document

The reason why I am creating this document is because it will show the evolution of the password generator program being developed. By having this document, anyone is able to make further changes to the program in the future, to respond to business needs. Bloomberg currently has measures in place such as finger print scanner on keyboards, for both employees and clients to use when logging into the terminal. However if their finger print scanner fails to detect a valid fingerprint, they will be required to input a password which can in turn override the scanner and consequently grant that user with the access of the terminal. Therefore, it is essential for a user to have a level of complexity built in to their password.   
  
Scope

This program will be integrated on to the Bloomberg software and everyone who has access to one, will be able to use the program. Data on Bloomberg needs to be kept confidential as each person within a firm pays £25,000 a year to have access to their terminal, and therefore passwords are essential in order to stop unauthorized users from accessing the software.

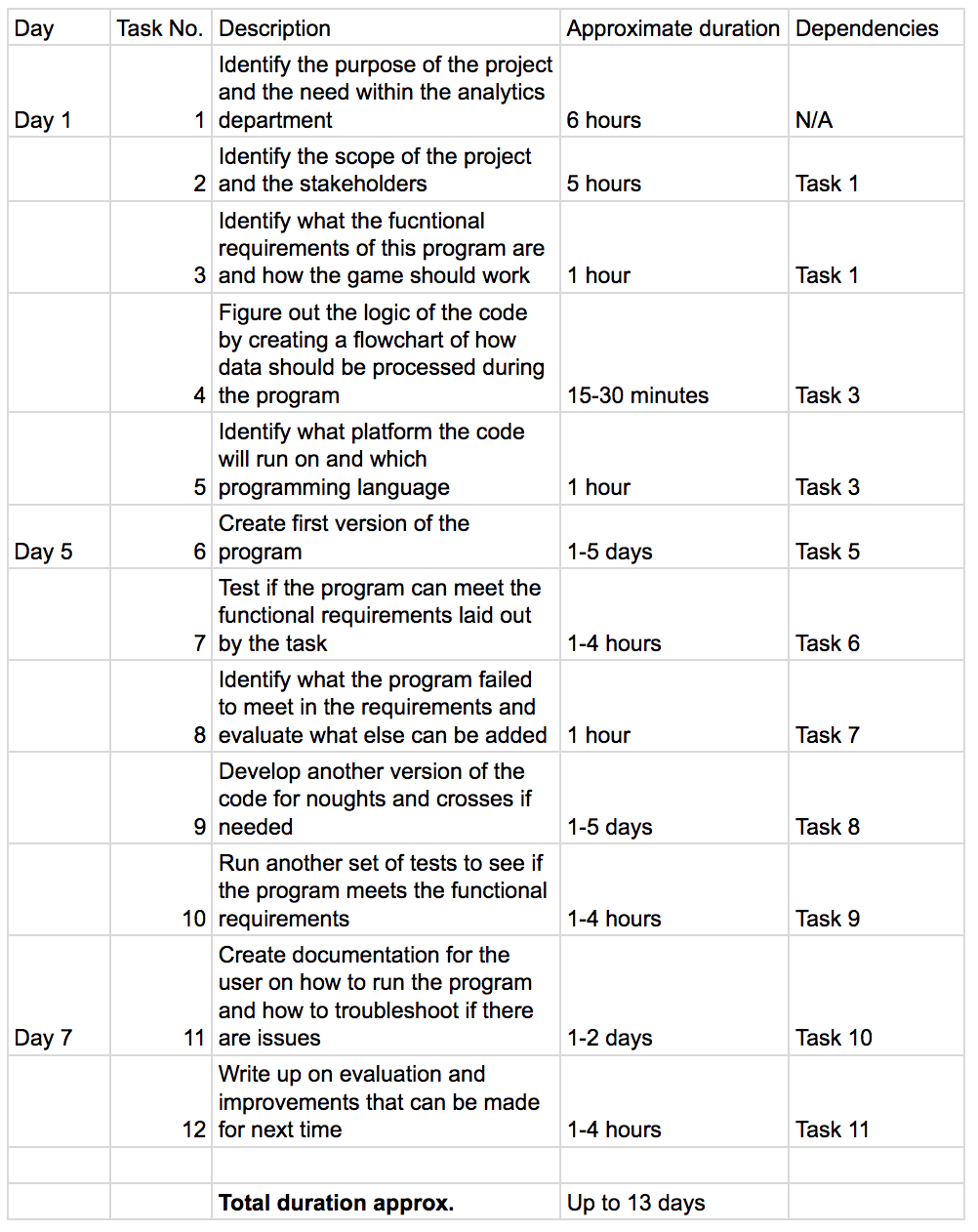
## Who with?

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 to provide feedback on the program, as to what they like and areas in which it can be improved, which will also be outlined in this documentation.

# **Project Overview**

## Solution Overview

#### **How the password programs should work** Password Generator

I will be creating two programs on the new BQUANT platform, which has integrated python as well as other programming languages.  
  
The first program referred to as the “Password Generator”, will make up a password containing symbols, lower/ be asked to input the length in which they want their random password to be. upper case characters and numbers. How it will work, is that when the program is run, the user will be asked to input a chosen digit in which they want their password length to be. Once inputted, a random password will be generated in which the employee could use for their account. The password generated will include the below at a bare minimum:  
  
 - Lower case letter  
 - Upper case letter  
 - Number/Digit  
 - Special symbol  
  
Password Strength Check  
The second program in which I will be creating is one that judges the strength of a password; known as “Password Strength”. How it will work, is that when the program is run, the user will be asked to input a chosen password. Once inputted, the strength of the password will be outputted, as well as it will be given a score out of 5, based on what it contains. This password strength program will be calculated based on the criteria, as stated below:  
  
 - Longer than 6 characters   
 - Contain a number/digit  
 - Contain a capital letter  
 - Contain a lower case letter  
 - Contain a special character   
  
If all the criteria as shown above is, this consequently indicates that the password is strong and therefore the user is able to use it for their Bloomberg login. However, if a user only matches one of the criteria, then the program will indicate that the password is weak and urges the user to try again.   
  
  
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.

## Operational Requirements Password Generator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req number | Operational Requirements | Inputs required | Processing Required | Outputs |
| 1 | Run the program, and wait for a welcome message. | The program should be run by the user | Verifies that the program is able to be played | Message to user  “Welcome to the random password generator!” |
| 2 | Program should prompt user to input a digit as the length of the password | The program should be run by the user | Verifies that the program is able to be played | Message to user  “Please input the length of your password which is multiple of 4:” |
| 3 | User should be able to input the length of the password | “4” should be the digit inputted for the proposed length of the generated password | Verifies that “4” is a valid length | Generated password is outputted to user randomly |
| 4 | User should be able to input the length of the password | “5” should be the digit inputted for the proposed length of the generated password | Checks if “5” is a valid length | Error message is outputted:  “Please enter a multiple of 4 only!  Please input the length of your password which is multiple of 4” |
| 5 | Program should recognize when a character is inputted, rather than a letter and should prompt user to try again. | “a” should be the letter inputted for the proposed length of the generated password. | Checks that “a” is a valid input | Error message is outputted:  “That was not a number! Please try again ...  Please input the length of your password which is a multiple of 4:” |

Password Strength Check

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req number | Operational Requirements | Inputs required | Processing Required | Outputs |
| 1 | Run the program, and wait for a welcome message. | The program should be ran by the user | Verifies that the program is able to be played | Message to user  “Welcome to the secure password checker!” |
| 2 | Program should prompt user to input a their proposed password | The program should be run by the user | Verifies that the program is able to be played | Message to user   “Input password to be checked:” |
| 3 | Program should prompt user to try again if they have a weak password | “abc” is inputted by the user as the password | Verifies that “abc” is a valid input  Checks if “abc” corresponds to 1 or more of the criteria | Message to user X–not longer than 6 characters X – no number X – no capital letter X – no special character ✓ - contains lowercase letter  1/5 Password Weak, try again |
| 4 | Program should recognize if user has a moderately strong password | “123Abc” is inputted by the user as the password | Verifies that “123Abc” is a valid input  Checks if “123Abc” corresponds to 1 or more of the criteria | Message to user ✓ – longer than 6 characters ✓ – contains number ✓ – contains capital letter X – no special character ✓ - contains lowercase letter  4/5 Moderately Strong. Well done ! |
| 4 | Program should recognize if user has a super strong password | “789Abc!” is inputted by the user as the password | Verifies that “789Abc!” is a valid input  Checks if “789Abc!” corresponds to 1 or more of the criteria | Message to user ✓ – longer than 6 characters ✓ – contains number ✓ – contains capital letter ✓ – no special character ✓ - contains lowercase letter  5/5 Super Strong ! Excellent work ! |

## 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

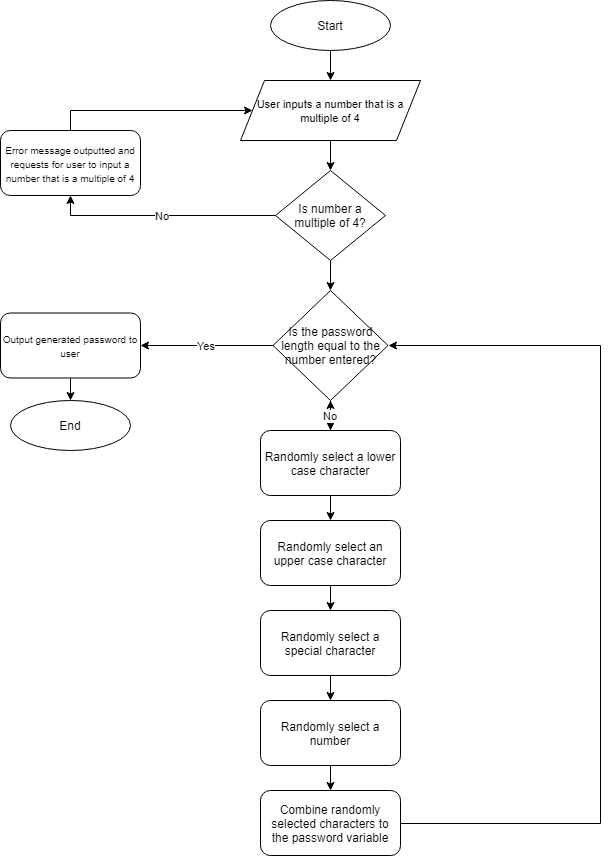
- It should be ensured that the program does not deal or contain any sensitive information regarding Bloomberg

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

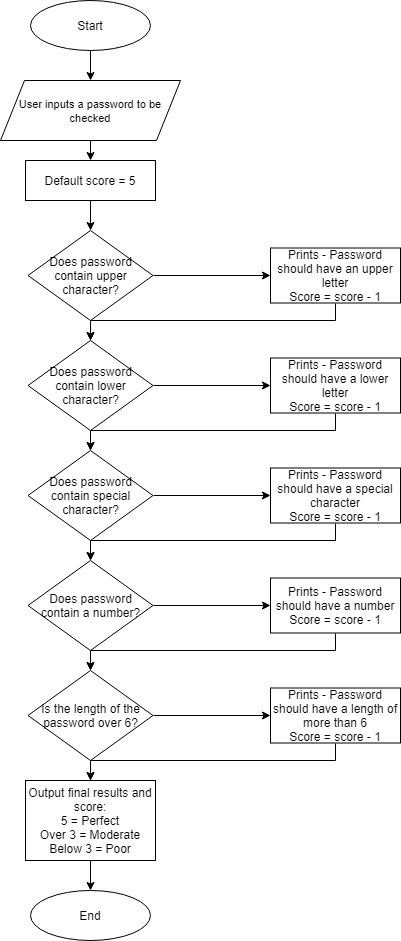
- There may be unknown variables that are encountered during development of the code

## Process Map

Password Generator



Password Strength Check

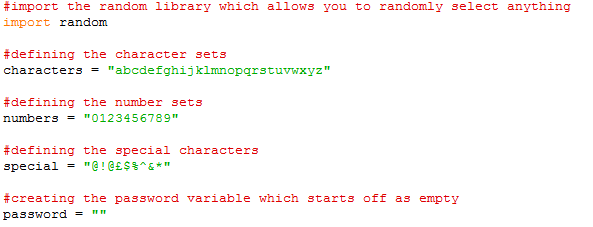


# Development of the Program Since I had developed two different types of password programs, the development phase was split into two main bits. The first bit was the password generator program in which it would generate a password from a combination of numbers, special characters, upper and lower-case characters. The second was the password strength check program in which would check the password based on a specific set of criteria.

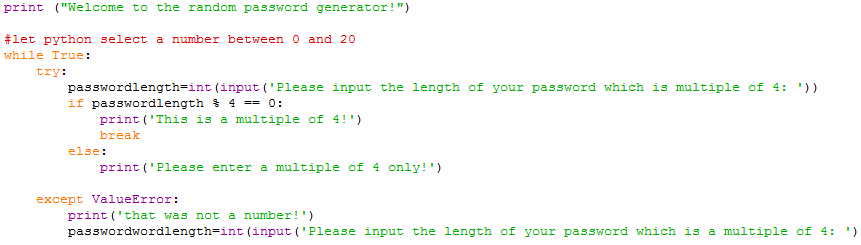
## Password Generator

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 allowed the program to randomly select from a list of numbers/letters/special characters when a password was being generated.   
  
Also, in this section are the variables in which the random library selects the characters from. “Password” is a blank variable as nothing has been added to it yet. There is no output from this section of code.

### Section 2

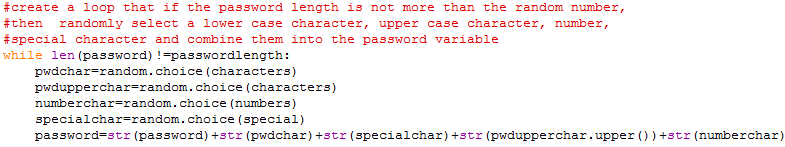
This part of the code prints out a message to the user welcoming them to the password generator. A new variable is also introduced here called passwordlength.   
  
The program asks the user to input a number which is a multiple of 4. It checks the number inputted is a multiple of four and outputs an error number if the user inputs incorrectly.   
  
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.

### Section 4

The final section of the code as shown below, outputs the results of the process in section 3 when the password that has been generated is longer than the number generated previously.

Output:

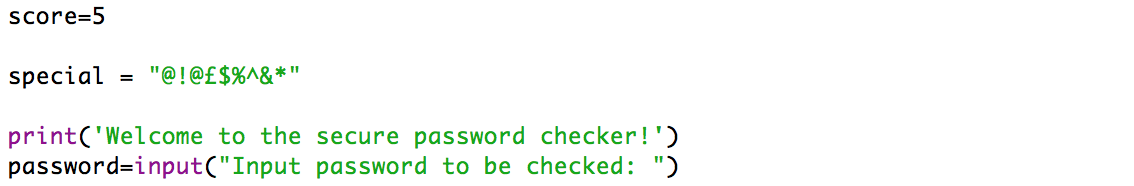
## Password Strength Check

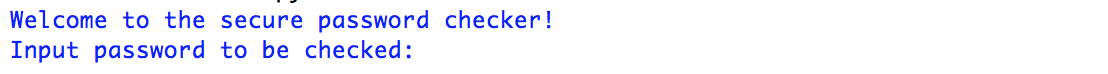
The password strength check requests for the user to input a password, and determines how strong the password entered by the user is, when matching it to the set criteria.

A score is then given to the user out of 5 and it also shows them what they are missing/did not include when they inputted their password

### Section 1

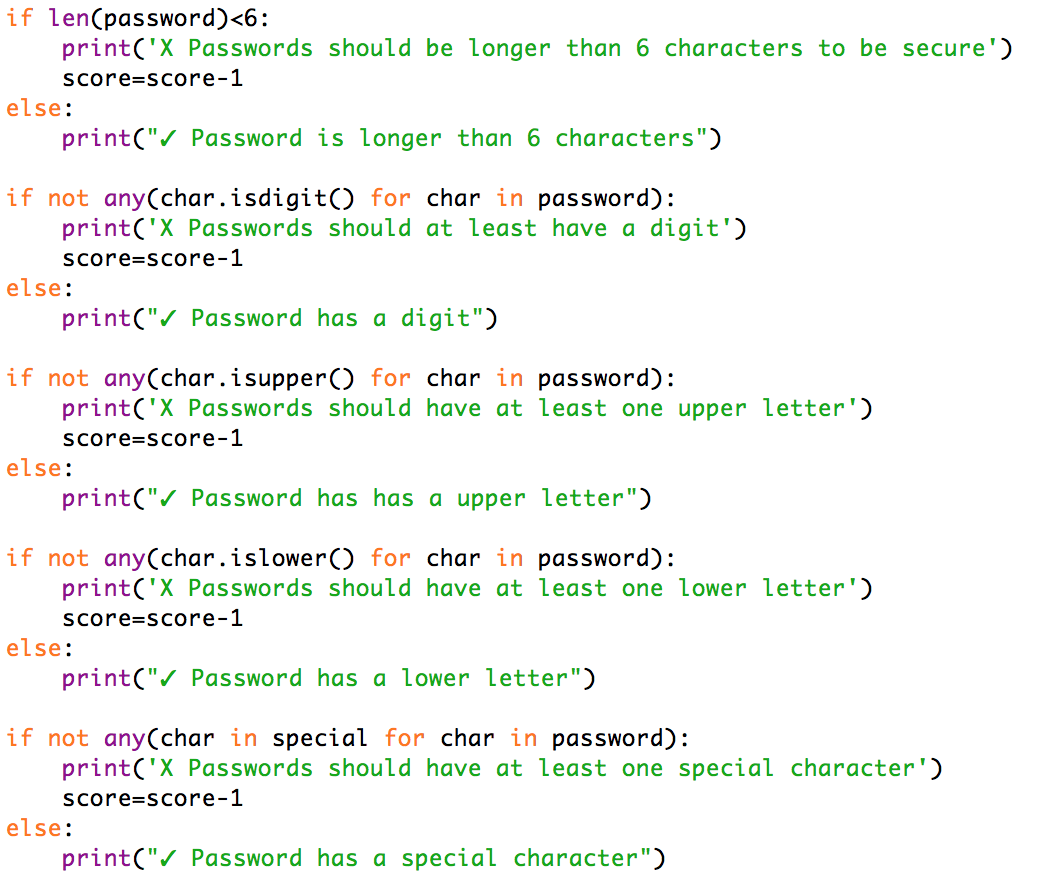
For the first section of the password strength checker, I defined the default score of 5. Later on, in the code, if the password that the user enters does not pass a certain criteria, a point is deducted from this score. I have also defined special characters under the ‘special’ variable as python does 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

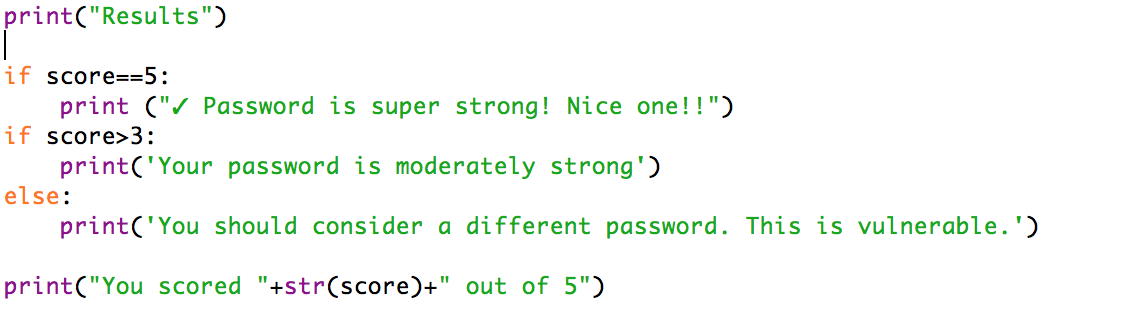
In this section, it is where the password to be checked if it meets the following criteria:

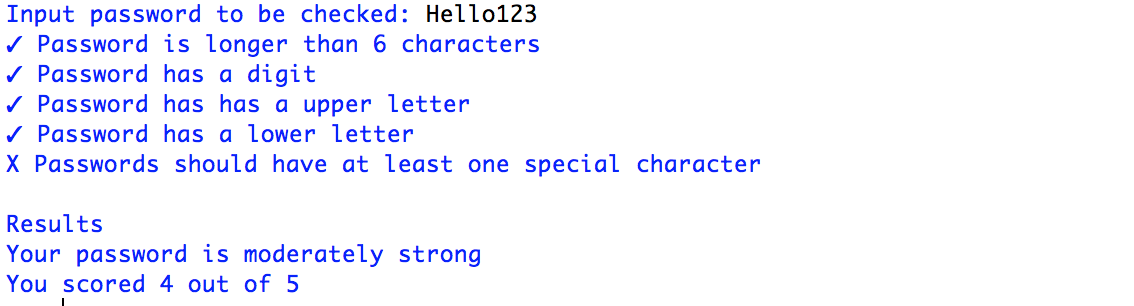
* Length is more than 6
* If there is a digit in the password
* If there is an upper-case character in the password
* If there is a lower-case character in the password
* If there is a special character in the password

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

### 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 3 is where the password is classed as 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 of password generator**

## 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 first welcome message | N/A | “Welcome to the random password generator!” |  |
| 2 | Run the program and wait for the program to ask the user for an input | N/A | “Please input the length of your password which is multiple of 4:” |  |
| 3 | Run the first program and enter a number that is not a multiple of 4 | 5 | “Please enter a multiple of 4 only!” |  |
| 4 | Run the program and enter a number that is a multiple of 4 | 16 | “This is a multiple of 4!” |  |
| 5 | Run the program and enter a character instead of a number | E | “That was not a number, please try again” |  |
| 6 | Run the program and enter a number that is a multiple of 4 and check that it is able to generate a password | 24 | “This is a multiple of 4!”  “Generated password: ” |  |

Test plan results (After carrying out testing)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 1 | Run the program and wait for the first welcome message | N/A | “Welcome to the random password generator!” | “Welcome to the random password generator!” |


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 2 | Run the program and wait for the program to ask the user for an input | N/A | “Please input the length of your password which is multiple of 4:” |  |





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 3 | Run the first program and enter a number that is not a multiple of 4 | 5 | “Please input the length of your password which is multiple of 4:” | “Please enter a multiple of 4 only!” |





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 4 | Run the program and enter a number that is a multiple of 4 | 16 | “This is a multiple of 4!” |  |

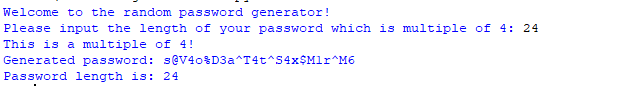




|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 5 | Run the program and enter a character instead of a number | E | “That was not a number, please try again” |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 6 | Run the program and enter a number that is a multiple of 4 and check that it is able to generate a password | 24 | “This is a multiple of 4!”  “Generated password: ” |  |





# **Testing of password strength checker**

## 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 check that the welcome message is displayed | N/A | “Welcome to the secure password checker!” |  |
| 2 | Run the program and check that the program requests for the user to input a password to be checked | N/A | “Input password to be checked” |  |
| 3 | Run the code and enter a weak password and the score should be 1/5 indicating to the user that the password is weak | abc | “X–not longer than 6 characters X – no number X – no capital letter ✓ - contains lowercase letter X – no special character  1/5 Password Weak, try again” |  |
| 4 | Run the code and enter a moderate password and the score should be 4/5 indicating to the user that the password is weak | 123Abc | Message to user  ✓ – longer than 6 characters  ✓ – contains number  ✓ – contains capital letter  X – no special character  ✓ - contains lowercase letter  4/5  Your password is moderately strong |  |
| 5 | Run the code and enter a strong password and the score should be 5/5 indicating to the user that the password is weak | 789Abc! | Message to user ✓ – longer than 6 characters ✓ – contains number ✓ – contains capital letter ✓ – no special character ✓ - contains lowercase letter  5/5  ✓ Password is super strong! Nice one! |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 1 | Run the program and check that the welcome message is displayed | N/A | “Welcome to the secure password checker!” |  |



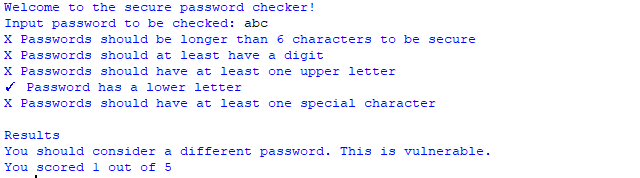


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 2 | Run the program and check that the program requests for the user to input a password to be checked | N/A | “Input password to be checked” | “Input password to be checked” |



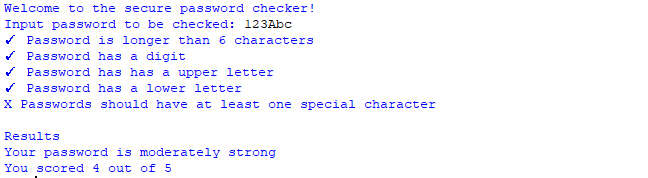


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 3 | Run the code and enter a weak password and the score should be 1/5 indicating to the user that the password is weak | abc | “X–not longer than 6 characters X – no number X – no capital letter ✓ - contains lowercase letter X – no special character  1/5 Password Weak, try again” | “X–not longer than 6 characters X – no number X – no capital letter ✓ - contains lowercase letter X – no special character  1/5 Password Weak, try again” |



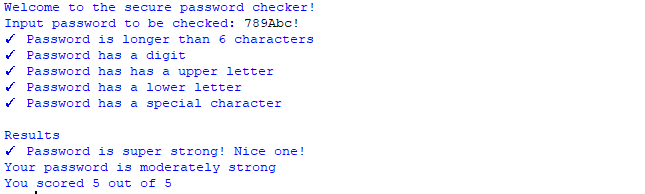


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 4 | Run the code and enter a moderate password and the score should be 4/5 indicating to the user that the password is weak | 123Abc | Message to user  ✓ – longer than 6 characters  ✓ – contains number  ✓ – contains capital letter  ✓ - contains lowercase letter  X – no special character  4/5  Your password is moderately strong |  |





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | What to do | Given input | Expected output | Actual output |
| 5 | Run the code and enter a strong password and the score should be 5/5 indicating to the user that the password is weak | 789Abc! | Message to user ✓ – longer than 6 characters ✓ – contains number ✓ – contains capital letter ✓ – no special character ✓ - contains lowercase letter  5/5  ✓ Password is super strong! Nice one! |  |

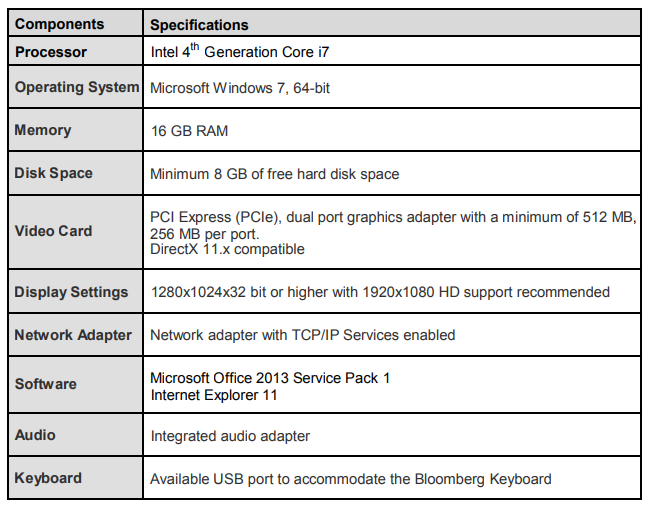




# **Documentation**

## 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

### Password generator

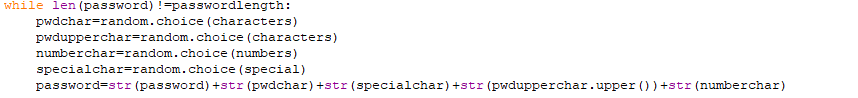
The final version of the password generator was quite efficient. However, there could have been improvements made to the code.

1. **Develop a way to allow the user to enter any number for the length of the password**

The major flaw that the code has at the moment is that it is limited to the user exclusively entering a number that is a multiple of 4. This could be improved on in the future where the code would instantly stop selecting characters and adding them to the password once the length that the user inputs is reached.

1. **Develop a way to randomly order the combination that makes up the password**

The second improvement would be that the pattern of the password generated could potentially be noticed. The password generator appends characters in this order: lowercase character, uppercase character, number and special character. This could lead to a security threat to users, and in future, the code could be made to randomise the order of character selection.



### Password checker

The password checker was quite a simple code which did not require a library to be imported in. The improvement that could be made to the code would be that the checks on criteria could be grouped together into one check and an output made when a criteria was not met.

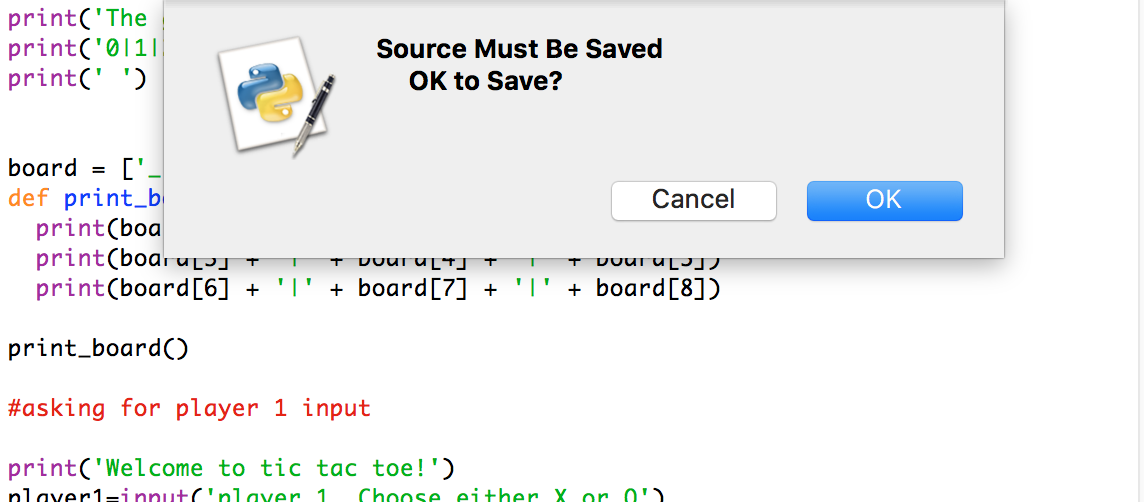
**Deployment**

When both the password programs were 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 release this for all users of the bloomberg terminal on the BQUANT platform. Consequently, a mass email was sent out to the whole of the Analytics department, in which a link to these password programs on BQUANT were included too, but also included was 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)

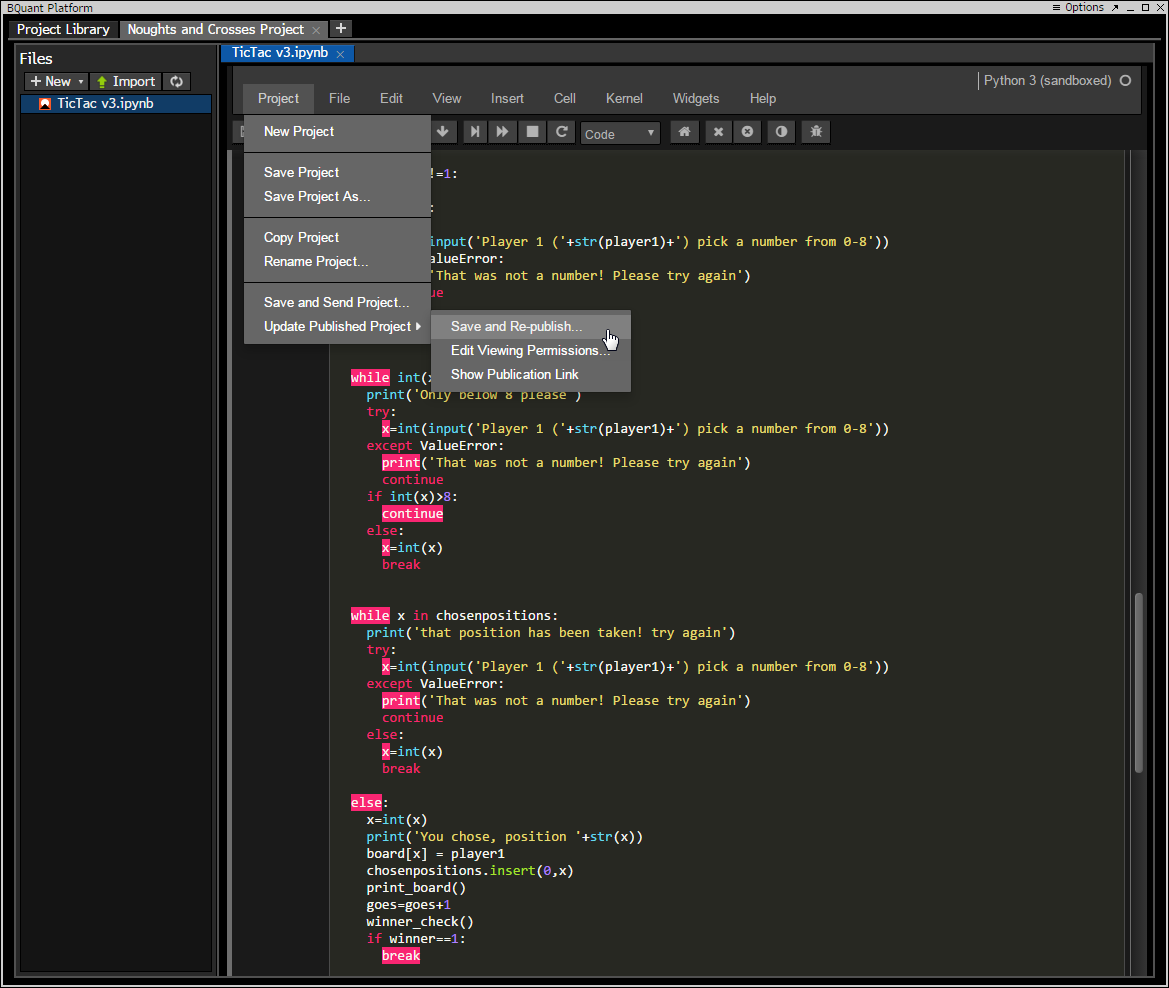
When your game program’s source code is available on the screen, press F5 from keyboard or select Run Module/F5 from Run menu to run the game program.



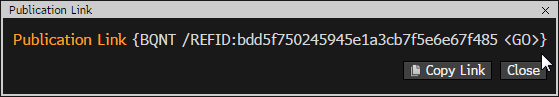
Python may ask you that the source must be saved. Click OK and the program will be opened

## How to run? (From BQUANT)

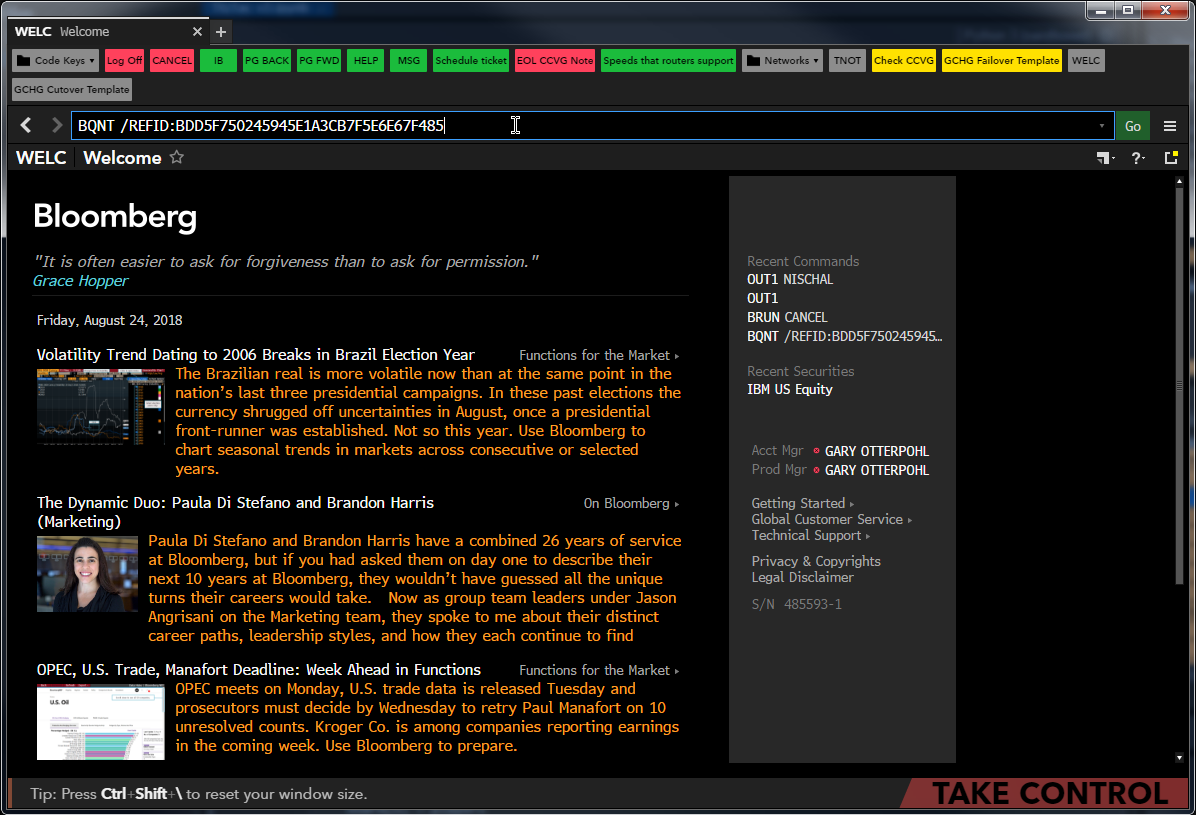
Within the BQUANT platform when you are on the game project file, go to Project -> Update Published Project -> Save a Re-publish.

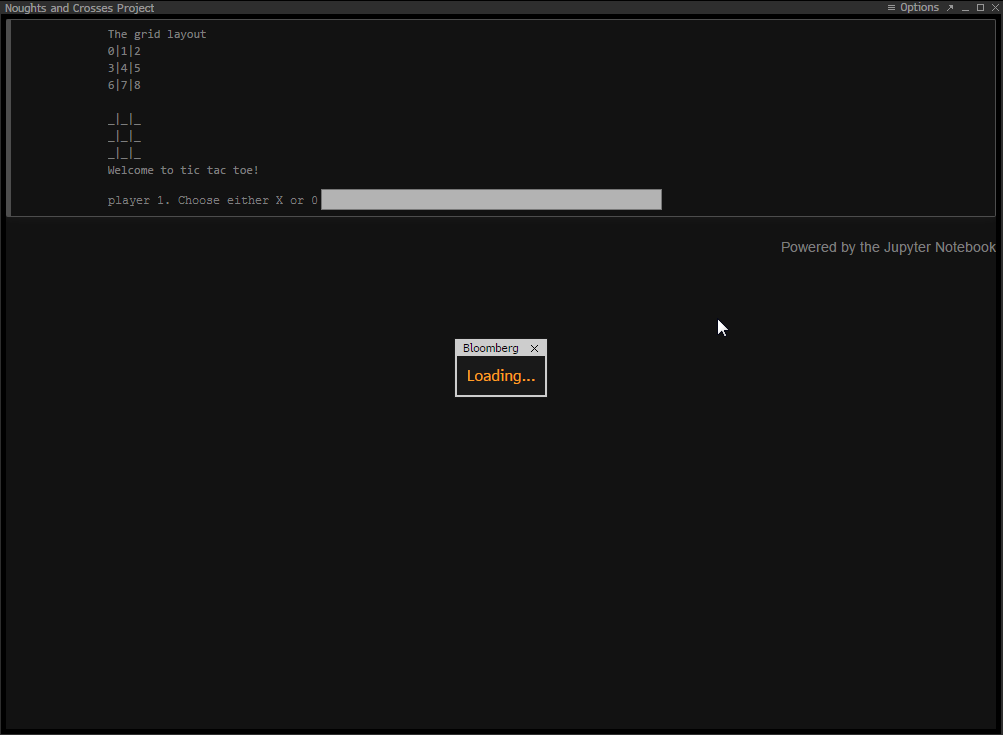


This will allow you to run the project. The Bloomberg terminal will then inform you of the link of the game to run it. Copy the link and paste it into a terminal window to run.



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)