Unit 1: Principles of Computer Science

Python Programming Project:
Tic-Tac-Toe Project

By Janet Voong

Planning

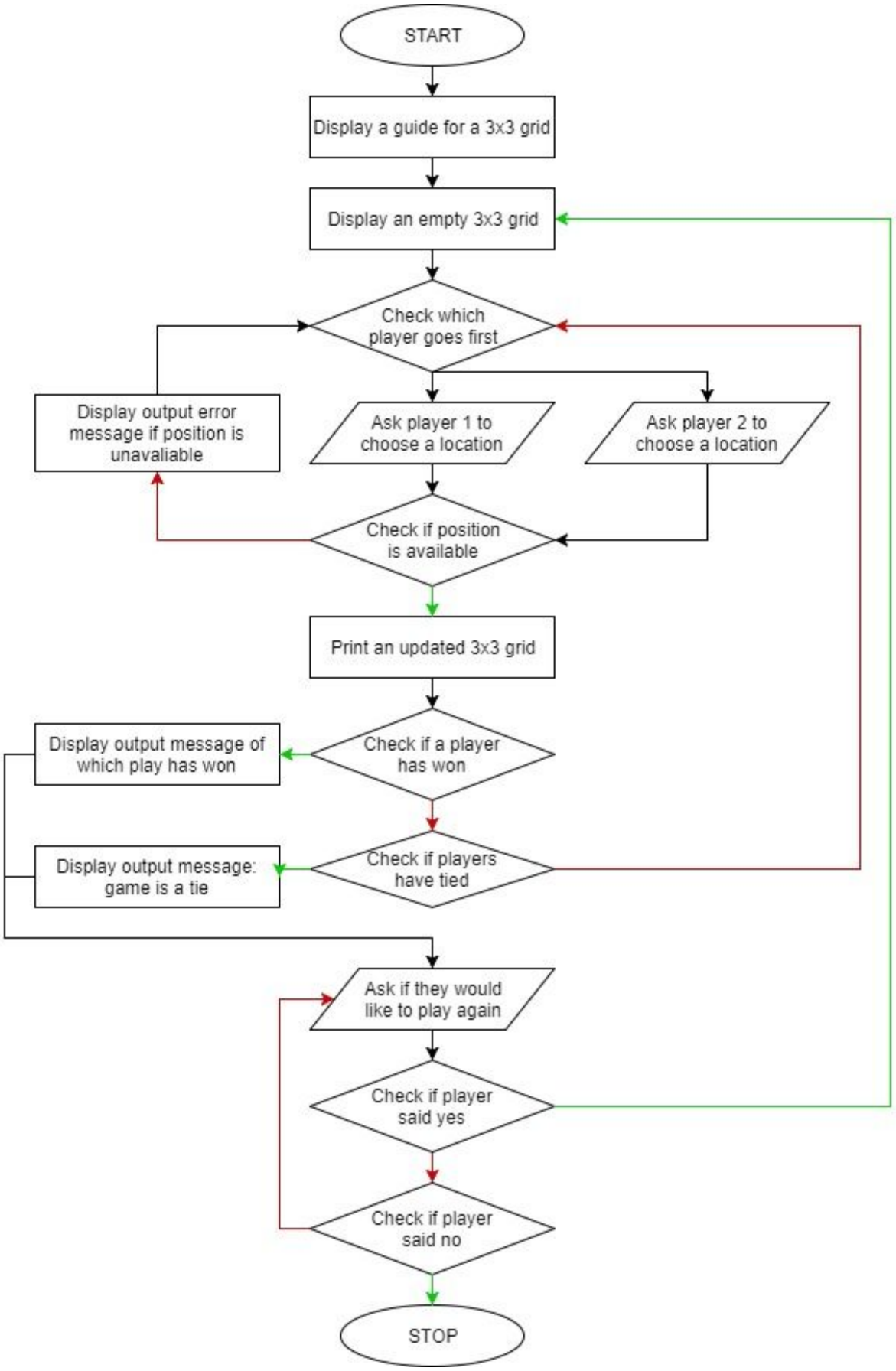
RULES / ASSUMPTIONS:

- There are two players and they are able to choose a character from "X" or "O" that they would like to represent.
- The two players are given or make a 3x3 grid.
- These two players will take turns filling out every free space.
- The game will stop if the same character is seen horizontally, vertically, diagonally, or when every space is no longer available (when no one wins, this is called a draw)

PLAN / VARIABLES:

- Grid (2D list)
- Mark/character ("X" or "O")
- Player ("1" or "2")

ALGORITHM / FLOWCHART: https://goo.gl/JnythU


START
Display a guide for a 3x3 grid
Display an empty 3x3 grid
Check which player goes first
Display output error message if position is unavaliable
Ask player 1 to choose a location
Ask player 2 to choose a location
Check if position is available
Print an updated 3x3 grid
Display output message of which play has won
Check if a player has won
Display output message: game is a tie
Check if players have tied
Ask if they would like to play again
Check if player said yes
Check if player said no
STOP

TEST PLAN:

Test Num	Description of Test	Test data	Expected outcome
1	The game will check if a player has won: to do this, it will see if the same character is displayed horizontally, vertically, diagonally.	Have player one place an "X" in the same column.	The game will say that player one has won.
2	The game will check if a position is already taken on the board.	Have player two place a character in the same place as player one.	The game will print an error message, stating that the position is already taken and so, the player can try again.
3	The game will check if the user's input for the location of their mark/character is out of range: this is because the only positions available are between 1 and 9.	Have a player type "15".	The game will print an error message, stating that the index is out of range and so, the player can try again.
4	The game will check if the user's input for the location of their mark/character is an erroneous data type as an index.	Have a player type a letter or word, such as: "Hello".	The game will print an error message, stating that the index is an erroneous data type and so, the player can try again.
5	The game will check if the user's input to play again is invalid.	Have a player type a number instead of "Y" or "N".	The game will print an error message, stating that the user's input is invalid and so, the player can try again.

PLANNING	Total For Task = 10 Marks

Develop

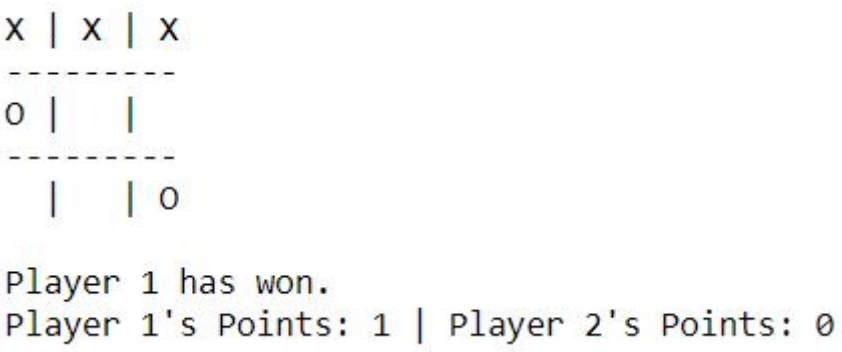
GitHub Repository Link: https://github.com/JanetVoong/Tic-Tac-Toe

Code Version 1: https://github.com/JanetVoong/Tic-Tac-Toe/blob/master/Version-1.ipynb
Code Version 2: https://github.com/JanetVoong/Tic-Tac-Toe/blob/master/Version-2.ipynb
Code Version 3: https://github.com/JanetVoong/Tic-Tac-Toe/blob/master/Version-3.ipynb

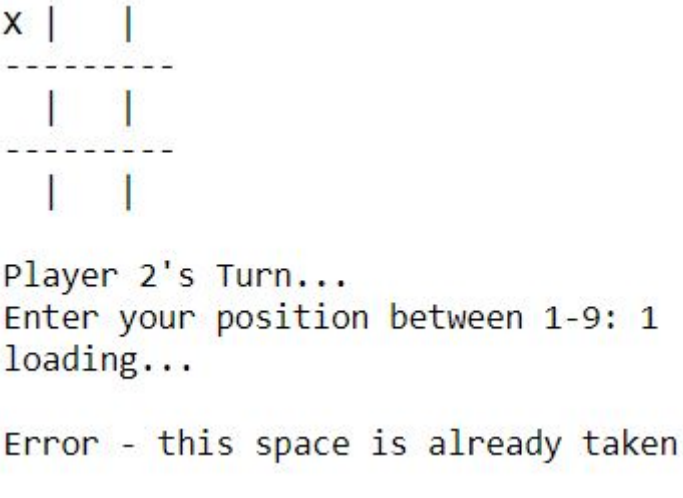
DEVELOP	Total For Task = 14 Marks

Test

Test Num	Actual Outcome	Comments and fixes
1	Game output message: "Player 1 has won."	Test was successful, no changes were needed.



2	Game output message: "Error - this space is already taken."	Test was successful, no changes were needed.



3	IndexError: list index out of range.	I used an if statement and a while loop to ensure that the user would only type numbers between 1 and 9.

Before:

```
Player 1's Turn...
Enter your position between 1-9: 15


---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-6fc1c8a6f93e> in <module>()
    132 print("Player 1 = X and Player 2 = O") # indicates to the players which character they will represent
    133 print()
--> 134 RunGame(GameRunning)
    135 print()
    136 CheckResult(board)

<ipython-input-2-6fc1c8a6f93e> in RunGame(GameRunning)
     85
     86         print()
---> 87         if(CheckMark(choice)):
     88             board[choice] = Mark # game will input the player's chosen position onto the board
     89             player+=1

<ipython-input-2-6fc1c8a6f93e> in CheckMark(x)
     30
     31 def CheckMark(x):
---> 32     if(board[x] == " "): # checks whether there is a space availiable
     33         return True # if so, the game will carry on running
     34     else:

IndexError: list index out of range
```

After:

```
Player 1's Turn...
Enter your position between 1-9: 15
Error - choose a number between 1 and 9.
```

4	ValueError: invalid literal for int() with base 10: 'Hello'	I used try and except within the previous while loop.

Before:

```
Player 1's Turn...
Enter your position between 1-9: hi

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-6fc1c8a6f93e> in <module>()
    132 print("Player 1 = X and Player 2 = O") # indicates to the players which character they will represent
    133 print()
--> 134 RunGame(GameRunning)
    135 print()
    136 CheckResult(board)

<ipython-input-3-6fc1c8a6f93e> in RunGame(GameRunning)
     82             print("Player 2's Turn...")
     83             Mark = "O" # player 2 will always represent "O"
---> 84         choice = int(input("Enter your position between 1-9: "))
     85
     86         print()

ValueError: invalid literal for int() with base 10: 'hi'
```

After:

```
Enter your position between 1-9: hi
Error - that was not a valid number.
Error - choose a number between 1 and 9.
```

5	Game output message: "Something went wrong, please try again." and stops game.	I placed this into a function and returned it in order for the question to loop back and not stop the game entirely.

Before:

```
Would you like to play again? Y/N: 147
Something went wrong, please try again.
147
```

After:

```
Would you like to play again? Y/N: y

  |   |
---------
  |   |
---------
  |   |

Player 1's Turn...

Enter your position between 1-9:
```

GitHub Repository Link: https://github.com/JanetVoong/Tic-Tac-Toe

Code Version 4:
https://github.com/JanetVoong/Tic-Tac-Toe/blob/master/Version-4-Testing.ipynb

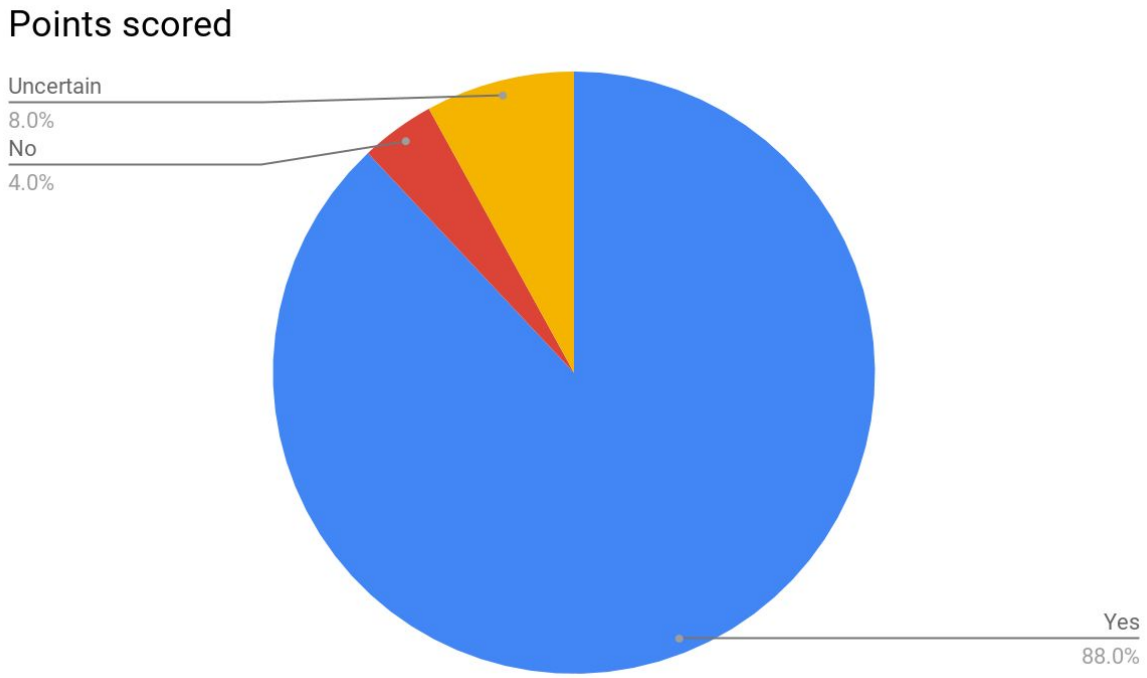
TEST	Total For Task = 7 Marks

Evaluate

Throughout the duration of this project, the naming conventions are not consistent: this is because some variables use camelcase and other variables are all in lowercase. Due to the fact that my game is for two players, I did not require to use external modules. Moreover, I utilised numerous parameters when calling my functions; for example, within the 'CheckResult' function, I have used the parameter 'board'. During the testing phase of the project, I had taken unexpected events into account: I conducted error handling with the ValueError and the IndexError, which had responded correctly.

In order to analyse the quality of my code for users, I have conducted a survey that consisted of 25 volunteers who have tested the code by playing my game and so, they were asked whether it was easy to use:

Survey: Is it easy to use and play?				
Options	Yes	No	Uncertain	Total
Frequency	22	1	2	25
Percentage	88%	4%	8%	100%



The features my game provides:

- Guide - This allows players to locate the position of the board.

- Multiplayer - This allows multiple players to play against each other, meaning the game would be more interactive.
- Point System with a Winner - A point will be awarded each time a player wins the game, which would engage players to play more.
- Play Again - This enables players to repeat the game with a new board as well as add to their current point system.
- Try Again - This would allow players to type another option if their current option is taken or incorrect.

Overall, my code is organised into functions and I have used correct indentation: this means that I am able to add more functionality to the code due to the fact that it is well organised and any user would be able to easily navigate any function that has been embedded in the code. Regarding functions, I have also used parameters and global variables appropriately. As a result, my code is able to deal with errors during execution however, I could potentially improve the error messages by making it more efficient and less repetitive.

EVALUATE **Marks**	**Total For Task = 6**

TOTAL	**Total For Task = 36 Marks**