Brandon Moy
4/10/19
Comp 150
Prof. Streeter

Design

- Text based game that runs in the computers terminal(Mac OS)/ Command Line (Windows/Linux)
- Game will also run the same way inside python's idle shell
- The game board prints out the variables with various symbols as strings in a specific way to look as if its a real game
- The beginning of the game will read a text file that prints the instructions on how to play the game with examples
- The program will then prompt the user on what mode they will like to play, their name, and what symbol they wish to play as
- After the game finishes between users or against the computer, options will be given to play again, quit, or change modes

How It Works / Methods

This program works with similar duplicate methods, so almost every method will be paired with another on how they function

nameSet()/ soloNameSet(): Grab global variables of player1 and player2 names from their initialization and re-defines them into holding new values. These variables are strings and consists of the users name or nickname.

playerSet()/playerOrder/soloSet()/soloOrder(): All grab global variables of players symbols, names and turns and changes them to determine which player plays as X or O and which player makes the first move.

soloPlayer()/player1()/player2()/comp(): All three of four methods work in similar ways, manipulating the global variables of the game board converting them into either an X or an O depending on the player of the Computer. The changing of these variables happen through a series of if statements through process of eliminations based on the users input changes the variable and makes that variable inaccessible from being changed again until the gameboard is prompted to be reset again. The computer method however works on a random drawing of a variable where it draws its choices randomly of a constantly shrinking list of variable options. This list is shrinking because for every move by both the player and the computer the list removes that used variable so it will not have a rand recurrence. These methods also give the

user a prompt if a variable is already used and gives them a retry limit of 3 before it goes back to the other players turn.

Play()/soloPlay(): Work exactly the same resetting all turn counts and gameboard back to their original initializations and then runs a while loop if the turn count is under the max move limit then it will either run the players specific turns in order.

subComp()/CheckWin()/compCheckWin()/subTallys()/subTallyM(): All work collectively running through all 8 win scenarios that are possible and prompts which player wins and what the tally is at. The tally keeps track of how many wins each player has. These methods also will prompt you if there is no winner and adds no tally to either score. These methods also give you an option to keep playing or not or ask if you want to change game modes.
BONUS: uses a timer of waiting 1 second before performing its next action

Draw(): prints out the game board with a sequence of variables and strings in a specific order and this board constantly changes throughout the game.

Appearance:

| X | 2 | 3 |

_____

| 4 | 5 | 6 |

_____

| 7 | 8 | 9 |

menu(): Prompts the use for which game mode they want to play and tells them to type the mode they want to play and sends them to that set of corresponding methods

clearBoard(): resets all changed global variables except for the user specific ones and resets them to their default constructors

main(): implements a time import and prints out the README text file and if the user understand how to work the game than can quit from the start or play the game. If they dont input the correct character they need to run the game it will tell the user to re-read the instructions and in 3 seconds will re-run the method again and will do this until one of the two correct options are picked.

BONUS:
Demo(): calls for two computer methods with parameters of a character X and O to play until one wins or the max amount of moves has been reached. Uses a timer to separate the distribution of moves between the computers moves.

DemoWin(): is a simpler copy of the other win scenarios

CSD(X): is the method that controls the computer's movements to play against itself

Implementations:

- Imports time
- Imports random
- Uses while loops
- Uses if, elif, else logic
- Frequent calls between regular methods
- Uses methods with parameters
- Reads outside files
- Prints information for users to see
- Global variables
- Lists manipulation
- Uses try and except