TicTacToe Game

Connects to the Twitch chat and takes in votes for which square that chat wants to move. Twitch Chat will be X and the computer move will be O. After intentional delay the X will be drawn and another delay for the automatic computer move which is a simple robot that picks a random square from the remaining squares. You can win by Row Column or Diagonal if both rows and columns are equal, in the occasion that you want to test the board with different rows and column sizes then the only way to win is from the greatest of the rows and columns. Once there is a winner or a tie then the board will reset and another game will continue with the computer move starting.

TicTacToe Board

Use turtle to draw the board with a dynamic amount of rows and columns which will dictate the size's of other aspects. The sizes of the X's and O's is dependent on the row and columns and the number of squares and square numbering as well. Each square has it's number on the upper left-hand corner of that square and that is what Twitch chat will be voting on for their next move.

```
SIZE = 300
# For Tic Tac Toe ROWS and COLUMNS should be the same
ROWS = 3
COLUMNS = 3
PENSIZE = 10
HALF_CIRCLE = 180
X_ANGLE = 60
FONT_SIZE = 12
```

These variables can be used to customize the size, the number of rows, number of columns. how thick the lines are, and how the O and X are draw.

Test TicTacToe

This is where the unit testing is done for the game. Feel free to use it as a way to demo all the ways to use this program

```
BOARD_SIZE = 2
BOARD_INCREMENT = 7
SLEEP = 1
```

These variables can be changed to show the TicTacToe board work for different size boards and the sleep can be used to see the boards for a longer amount of time.

Twitch Plays hackRU

twitch_plays_hackru is a Python library for twitch chatters to be able to vote on commands in a game.

Installation

Use the package manager to install twitch_plays_hackru.

Usage

Importing the classes:

```
from twitch_plays_hackru import TwitchPlaysOnline, TwitchPlaysOffline
```

Initializing the TwitchPlaysOnline or TwitchPlaysOffline object:

```
voteingOptions = ["1","2","3","hi","bye"]
tPlays = TwitchPlaysOffline("irc.twitch.tv", 6667, "oauth:YOUR_OATH_CODE_HERE", "TwitchBot", "YOUR_CHA
```

The object will take in **7 parameters**:
SERVER: the server that the bot will be interacting with. In this case it will always be "irc.twitch.tv"
PORT: the port you would like use.
PASS: the OAuth code for the twitch channel you would like this bot to listen on. Use twitchapp.com/tmi/ to generate the OAuth code for your twitch channel.
BOT: the nickname of the bot.
CHANNEL: the name of the channel you would like this bot to listen on.
OWNER: the username of the owner of the channel you would like this bot to listen on.
OPTIONS = []: an array of options that you would like the bot to keep track of.

Chatters voting:

The initialization of the bot will also startup the bot. This means that the bot will send a "Hello World" message in the chat and then begin to listen to all of the chat responses.
Chatters will preface their vote option by using **play_** . For example, a vote for **hi** from the code above must be typed **play_hi**. It is not case sensitive.

Functions:

vote_results():

```
result = tPlays.vote_results()
```

vote_results() returns the majority vote since the last call of vote_results(). The number of votes for all options are reset to 0 after the call. If there are no votes since the last call of vote_results(), None is returned.

License

GNU