Battleships-t2a2

Battleships is a multi-player text based game written in python demonstrating development capabilities with networking, use of classes and object oriented programming.

Installation:

To download the source code of the application execute the following command at your command line prompt: `git clone https://github.com/l-j-g/battleships-t2a2.git`

Dependencies / Packages Used:

This program runs on Python. To install Python, check out https://installpython3.com/'

Before starting the application ensure that all of the system environment requirements have been met.

To install the necessary dependencies, navigate to your type src folder and execute the following command in your command line prompt: `pip install -r requirements.txt`

Battleships requires the following python packages to be installed.

- `random` : python standard library, used to randomly generate numbers
- `numpy` : external python package, used to create an array and manipulate data that represents a battleships game board
- `pdb` : python standard library, used for debugging
- `socket`: python standard library, for networking and establishing connection between server and client.
- `sys` : python standard library, used to exit the program.
- `art` : external python package, used to display game title.
- `traceback` : python standard library, used to display debugging information upon error.
- `literal_eval` from asl - python standard library, used to evalute strings communicated over network connection as a Python expression.
- `os` - python standard library, used to evoke a function that will clear the terminal screen, regardless of users operating system.
- `ipaddress` - python standard library, used to check if a string is a valid ip address.

How to play:

Battleships is a two player game where players place up to 5 ships of different lengths on a 9x9 grid.

Players place all of their battleships on the board and do not disclose to the other player the location of their battleships.

Players then take turn to guess the location of each others battleships one element at a time.

When either player has guessed the location of all of the other players battleships the game is over.

Input

Input is taken from the user during three distinct processes:

- Setting up connection: port, address and role
- Placement of ships: row and column
- Co-ordinates of an attack: row and column

Output

Output to the user is provided by `draw()` function of the `Battleships` object. which visually represents a `numpy` array of data according to the following legend:

Logical	**Numerical**	**Visual**
An empty element	0	`~`
A friendly ship	1	green `x`
A damaged shit	2	red `x`
A missed attack	3	blue `o`

Error Handling

Errors are handled through the program by type and logic checking input.

E.g is the input of a guess of type int? and, are the provided inputs within the dimensions of the game board.

As well as a turn based control flow that ensures two way communication before the game is progress.

E.g. During each turn one player will enter co-ordinates of their attack and send these co-ordinates to the other player.

The program will check if the co-ordinates of the attack correspond to the location of where the players ships are placed and will respond the player who send the attack to indicate if the attack was a hit or a miss.

Until this two-way communication is completed each turn the game cannot progress.

Control Flow:

At a high level the control flow of the program can be described as follows:

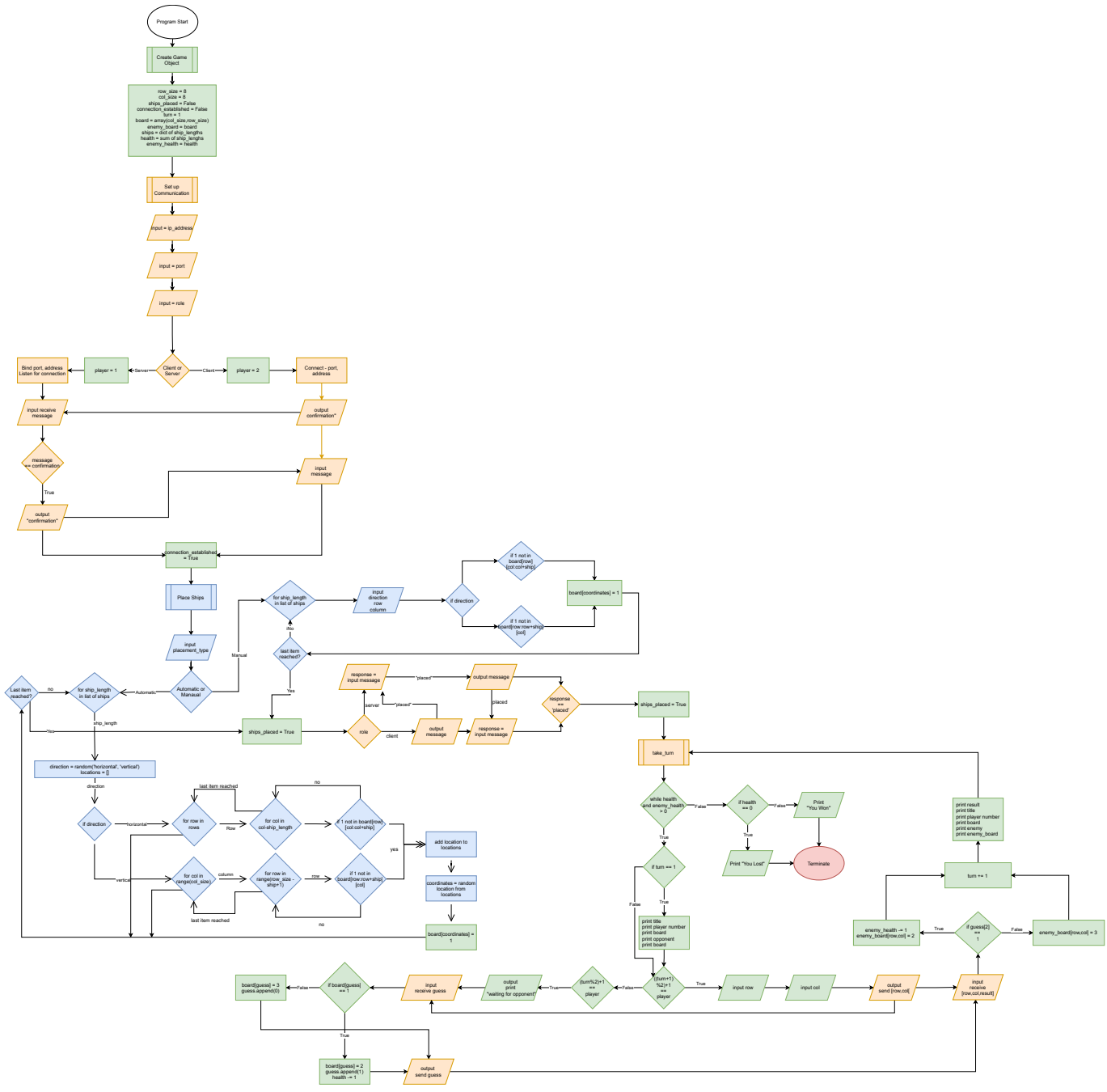
- initialise battleship class
- initialise connection class as either server or client
- place ships on to the board
- while ships remain on both players boards, players take turns guessing the location of enemy ships.

The programs control (without error checking for brevity) flow is described in the attached flowchart.

The different processes of the program are colour coded as follows:

- `green` : relating to battleship class functions and variables, used to provide input and output to the user of the application.
- `blue` : relating to the placement of ships on the game board

- `yellow` relating to Communication class functions and variables, used for two-way communication between a server and a client.



Classes

`classes.py` utilises three different Python classes.

- Battleship class: to represent one players game of Battleships
- Connection class: to establish a two-way connection required for a game of Battleships
- Format class : to represent ASCI escape codes