*CPP IT5014 Programming Principles Project*

**Project Report: Battleships in Python**

By Mark Cooper - 92018808

Development Method:

I used a **Scrumban** agile method with an iterative/user story backlog approach. I worked on one user story at a time.

I implemented the following development process:

Create/pick a few user stories -> Write some code -> Test some code -> Write some more code -> Test more code -> Create/pick a few user stories…..

For time, I spent about 45% coding, 45% testing and 10% eliciting requirements/documentation.

It took me about 25 hours so

Development: 10 hours

Testing 10 hours

Requirements, documentation and miscellaneous: 5 hours

Requirements Specification:

**User Stories:**

User story 1:

“As a player, I want to be able to be notified of the correct format to enter in my ships orientation and coordinates so that I can place my ships correctly with minimal frustration.”

-*Complete, user is prompted with correct display of the format – changed example coords to valid ones to avoid confusion.*

User story 2:

“As a player, I want to able to see a visual representation of my sea aka battlefield, with coordinates, so that I can get a clear idea where I have placed my ships.”

-*Tried a wide array of coordinates to see if it displays properly, no overlapping cords etc. Find a way to clarify SxR SxM SxF (r = rear) etc.*

User story 3:

“As a player, I want to see how many turns that a game has been running, so that I can appreciate when my ship placement has worked out well on my longer rounds.”

* *User story complete, at the end of each game it tells you how many rounds you lasted.*

User story 4:

“As a player, I want a clear visual representation of when one of my ships have been hit so that I can see clearly how the round is going for me”

* *User Story complete, visual representation in game.*

User story 5:

“As a player playing a turn-based game, I want to be in control of the speed of the game so that I can enjoy the game at my own pace.”

* *User story 5 complete, player is prompted to push enter in between computer guesses.*

User story 6:

“As a player, I want to be able to exit the game easily so that I do not get frustrated when trying to end my game session”

* *Option to opt out after each round, can also close console mid-game.*

User story 7:

“As a player, I want the game to display game stats and “hang” when the game is over, so that I can appreciate that round I just played.”

* *Previous game remains visible until the player chooses to proceed.*

User story 8:

“As a player who likes to control his game time, I want the game to prompt me before starting a new game so that I can more easily control the time I spend on the game.

* Prompted to start a new game at the end of each round with a message notifying the user of consecutive games played.

User story 9:

“As a player who wants an authentic experience vs AI, I don’t want the computer asking me the same coordinates more than once, so that the experience better and is close to playing a human”

* *Complete, the computer does not ask the same coordinates more than once, sets of coordinates are pulled from a list/array and deleted after.*

User story 10:

“As a player who wants an impactful experience, I want the console messages to be meaningful and have some punch so that I get an epic experience”

* *Console messages are impactful*

User story 11:

As a player who has just finished a game, I want to be able to seamlessly start a new game so that I don’t have to waste time and effort closing down the console application and rebooting the program.

* *User story fulfilled with an if/while statement combo in the Main Program Python file.*

User Story 12:

As a user who is playing a game of battleships, I want to be able to see what guesses I have made without having to scroll up through the console so that I can avoid wasting a turn by guessing the same co-ordinates twice.

* *User story fulfilled, player guesses are stored in a list and printed to the player every few turns*

User Story 13:

As a user who wants an intuitive experience, I want the application to handle the incorrect input gracefully, without crashing the game.

* *All input is validated, I used mechanisms such as loops and if statements to ask the user to re-enter any invalid entries.*

User story 14:

**Optional Advanced Tasks that I completed**:

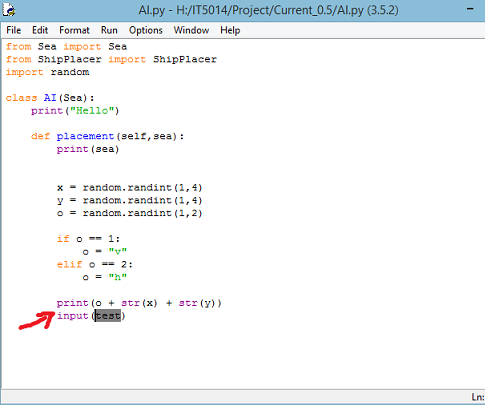
* Validation that prevents the user from entering an out of bounds ship
* Validation that prevents adding a ship if it creates overlapping positions with an existing

ship

* A user friendly console representation of ships at sea e.g.

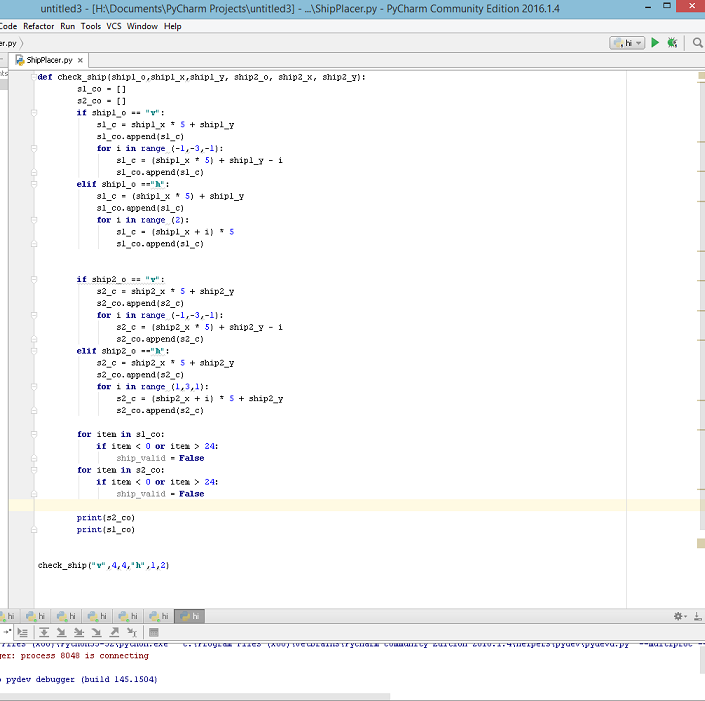
Testing:

* The testing I carried out included unit tests, such as this one I did to play around with a few values to determine how I would build my place ship method etc…
* As shown in some of the pictures below I used PyCharm for its inbuilt debugger and to have a quick way of testing code.

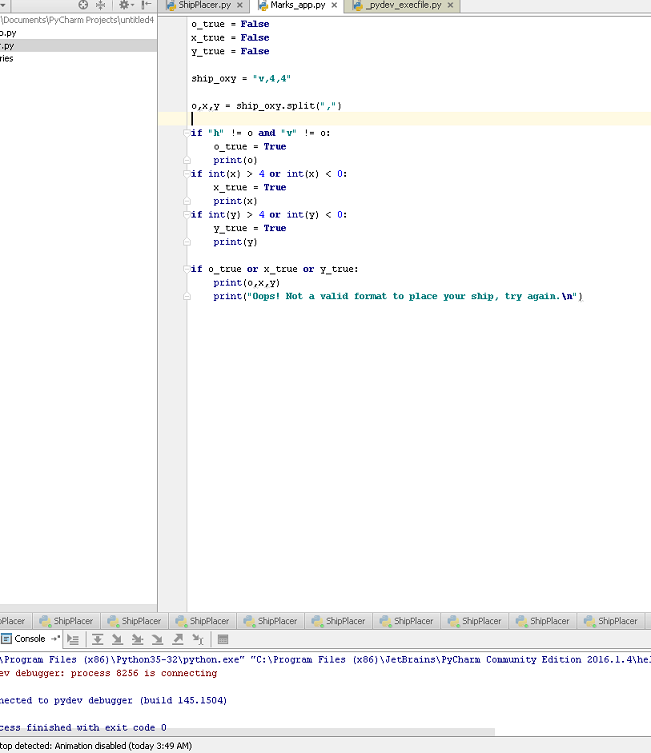


*A test method I used was adding an input prompt where I wanted to program to stop executing to inspect some variables (in this instance, o, x and y)*

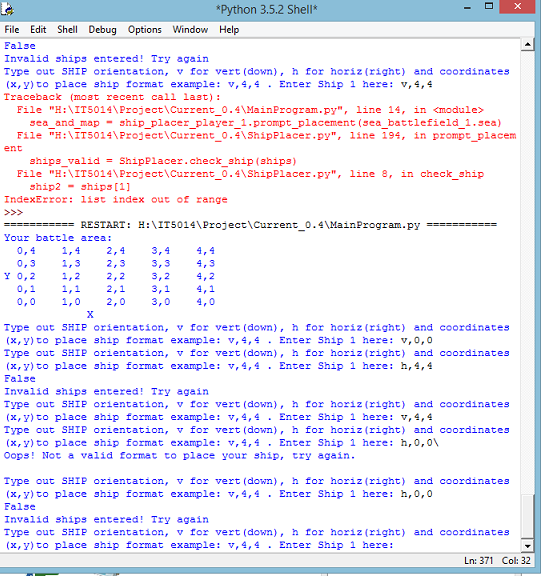
**MORE IMAGES BELOW**



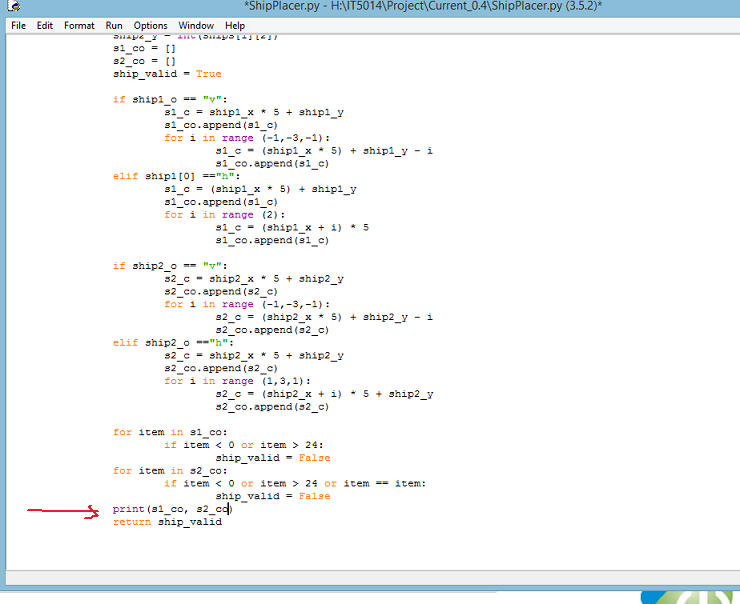
*A unit test that tested if my ships were valid and did not overlap*



*Another unit test done for the feature of validating user input.*

**

*An example of my testing input/output with the console.*

**

*Here, I add a print comment to the end of the function to see if the values are what they should be.*

* Another testing technique I used was thousands of console input/output tests
* Also in my testing toolbox, I used a lot of print() methods to check what some of my variables were doing, commenting code to be tested again later, usually print() statements.
* I did a lot of agile testing against the user requirements (user stories). This typically involved just playing out the game to ensure that the requirement was satisfied.