

Task description:

Battleship The task is to create a simple version of an imaginary game. The game Battle Ships Battle Ships game is basically as follows:

- It consists of two boards, one for each player, on 10x10 squares -Each player has 5 ships, one with length 5, one with length 4, two with length 3 and one with length 2
- Each player must place his ships on the board as he wishes. The ships can stand either horizontally or vertically. The ships cannot overlap or cross each other.
- Then each player should try to shoot down the ships of the other player. In turn, the player says which route he / she is shooting on. The opponent says if he hit a ship and if the ship sank. A player can only shoot on the same route once. The player basically does not know where the opponent has placed his ships.
- A ship sinks if all the squares it is in have been shot at by the opponent.
- A player loses the game when all his / her ships are sunk. The other player wins.

This exercise is to create a one-person version of the game where the computer places the ships randomly, and where the goal of the player is to shoot down all the ships with as few shots as possible:

Exercise 1)

Make a class for a game board. This class should have the following properties.

- A. variable that counts the number of the shooter has used
- B. A list of ships, starting empty
- C. A matrix or list of lists with Boolean values that say whether this route has already been shot at or not. This matrix should have 10x10 elements where all start with the value False. For a list of lists you must first create the outer list, insert 10 inner lists, and for each inner list insert 10 False values. If you want to use a 10x10 numpy array, you can use 0 for False and 1 for True. A matrix or list of references. This matrix should have 10x10 elements where all start with the value None. A list of lists is made as in the previous subtask. To create an array, you must specify the parameter "dtype = object" to np.zeros to create a numpy array that accepts objects and not just numbers.

Exercise 2)

Create a method to check if the placement of a ship is legal. A location is legal if the entire ship is within the game board and none of the routes the ship is going to cover already contain a ship. Use the matrix from problem b–d to check if the routes already contain a ship. The method should take the start coordinates, the length and direction of the imagined ship as parameters. The method should return True or False.

Exercise 3)

Create a method that places a ship. It must first use the method from the previous subtask to check that the placement is legal. If the location is legal, it should create a new Ship object and put it in the list of ships. In the reference matrix (subtasks b -d)it should set all the routes that the ship covers to

refer to the ship (instead of being None). The method should return True if the placed ship and False if the location was illegal.

Exercise 4)

Create a method that shoots on a route. The method should take the coordinates of the route as parameters. The method must first check whether the user has already shot at the route. If not, it should check if the route contains a ship. If it contains a ship, it shall call the hit () method of the ship. Regardless of the shell cell of the square the user has shot at in the matrix over hits is set equal to True Exercise

5)

Create a method that prints the board, which for example can look like the figure below. Exercise

6)

Create a method that checks if the user has won by going through the list with ship and check that everyone is sunk. Exercise

7)

Create a function or if `__name__ == «__main__»` block that plays the game. It shall place one ship of length 5, one ship of length 4, two ships of length 3 and one ship of length 2 in random places and in a random direction. it finds a legal place. Then it should let the user shoot by entering coordinates until the user has shot down all the ships. Then it should print the number of shots the user used.