day26 battleship game
Due: Friday 10/20/23

Open the attached repl. Fork it. This contains the base of a battleship game.

When you run it, you get this:

```
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

This may not look like much, but it's the ocean, and we're going to make a Battleship game.

The repl I gave you has code that creates a "board" for playing the game. The board as it comes to you has five rows and five columns. Because this is a computer program, the rows are numbered 0 through 4, as are the columns.

If we add this code on line 8…

```
board[4][0]="X"
```

… the program prints the following:

```
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
X ~ ~ ~ ~
```

The board[4][0]="X" command tells Python to access row 4, column 0 of the board and to make that spot an "X". The first number in brackets is the row, the second number is the column. Change that line to board[2][3]="X" and you get this:

```
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ X ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

Remember, the rows go 0, 1, 2, 3, 4, so row 2 is the third row down, then column 3 is the fourth column over.

Remove line 8, that was just so you could see how the rows and columns work.

Today's task is to make a game of battleship. Details are on the next page.

Requirements:

- Put your name and the date in a comment at the top of your program.
- Make your game 8 by 8 by changing the boardSize variable to 8.
- Add "import random" at the top of your program, and before the while True loop randomly choose the ship location as follows: Use random.randint(0,boardSize-1) to get a random integer between 0 and 7 (boardsize-1) for the row, and again for the column of the ship. Remember, although your board is going to be 8 by 8 (or boardSize by boardSize if you use larger numbers), the first item in a Python list is item 0, which is why you need a random number between 0 and 7 (boardsize-1) for the row, and another random number in the same range for the column. You would do:

  ```
  shipRow = random.randint(0,boardSize-1)
  ```
  and
  ```
  shipColumn = random.randint(0,boardSize-1)
  ```

  to create a random location for the ship stored in row shipRow and column shipColumn. We use boardsize-1 instead of hard-coding the number 7 so that you can make the board larger or smaller by simply changing a single variable (boardsize) up at the top.
- Print the ship location once at the start of your program to help with debugging. See my sample run below for an example.
- Remove the "break" from the while True: loop I gave you. The code that is already in the loop prints the board. This should be how each turn starts, with the board printing. So after this code, ask the user to enter a row and a column (both integers). After they do, check if the row the user entered is equal to shipRow and if the column the user entered is equal to shipColumn, and if they are, the battleship has been sunk. If they are not, put an X in the spot the user torpedoed and tell them they missed.
- When the user nails your ship, say "You sunk my battleship!" or something fun like that, then quit. Again, see my sample output below.
- Once you get the program running, try it out. Does it print the location of the ship at the start, just once, so you don't have to try 64 turns to get a hit? Try taking some shots. Does an X get put in each miss spot? Does the board print before the user is asked to enter a row and column on each turn? This is phase 1 of today's program. This actually gets a working battleship game running, which is no small feat. Well done if you get this far.

  Some things to test for at this point:
  - Does your program print the ship location clearly once at the very start?
  - Does the board print out right before the user is asked to enter a row and column on each turn?
  - Does an X get put into the spot of your shot when you miss?
  - Are you told you missed when you miss?
  - If you hit the ship, does it say "You sunk my battleship!" or something like that and then quit?
- Get all of the above working before you move on to part 2, described on the next page.

- Part 2 is to modify your program to use try/except statements to make sure the user enters integers for their guesses and let the user try again if they enter invalid data.

  Make a separate try/except while True loop for the row and for the column. This is because we don't want the user to have to re-enter a row just because they entered an invalid column.

  Test your try/except structures. What happens if you enter nothing instead of a number? You should be told "Invalid entry, please try again." or something like that.

  If you enter a valid row, but then a bad column, your program shouldn't require you to re-enter the valid row.

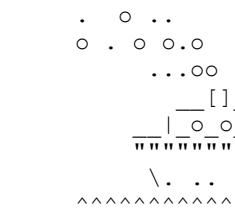
- After you get the basic try/except structures working, add if statements inside the try/except structures to verify that entered integers are greater than or equal to zero and less than boardSize. In a while True: loop you can use the "continue" command to make Python jump back up to the start of the current loop. There are lots of ways to do this.

  Here are some tests you need to do to make sure you are doing everything correctly:
  - If you enter nothing instead of a row or column, does it give you an error and let you try again?
  - If you enter a valid row, but then enter an invalid column, does the program make you re-enter the row? It should not.
  - If you enter a number larger than 7 (boardsize-1) or less than 0 does it give you an error and let you try again?

- That's it! If you made it this far you will get full credit and you have a robust Battleship game to play.

See sample output on the next page.

Here is sample output:



```
[For testing: Ship is at 1,1]

Welcome to Battleship!
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
Take a shot:
 Enter a row:    3
 Enter a column: 4
Miss!
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ X ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
Take a shot:
 Enter a row:    20
  Out of range, please try again.
 Enter a row:    2
 Enter a column: one
  Not a valid entry, please try again
 Enter a column: 1
Miss!
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ X ~ ~ ~ ~ ~ ~
~ ~ ~ ~ X ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~
Take a shot:
 Enter a row:    1
 Enter a column: 1
Hit! You sunk my battleship!
```