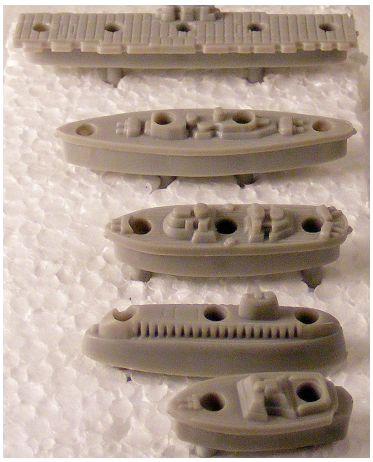
CS118 Program
Battleship Prep

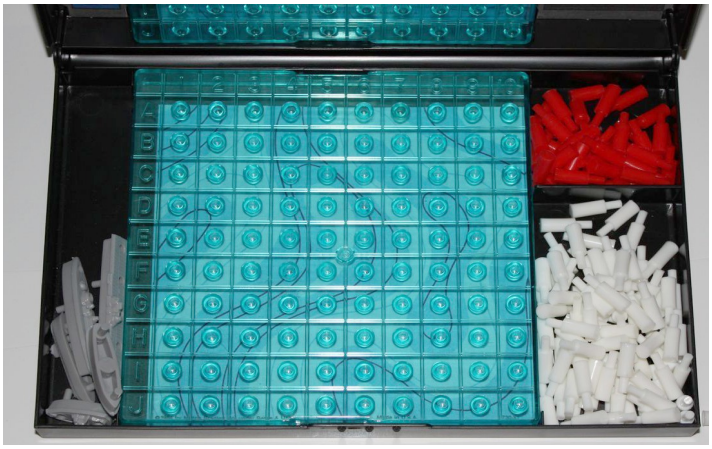


Prepare a Python program that will setup a game of Battleship.

As you can see from the image, there are five kinds of ships, each with 2 to 5 locations.

Aircraft Carrier	5 holes
Battleship	4 holes
Destroyer	3 holes
Submarine	3 holes
PT Boat	2 holes

Prepare a game board for each player. Python 2 doesn't support 2-dimensional arrays directly, so we'll implement these boards as lists of lists – the internal list is a horizontal row on the board. Each row consists of 10 spaces, and there are 10 rows on the board. On the physical board, row counting starts at the top and column counting starts at the left side.



Write code that will create a single board (called `BOARD`) with -1 in each space of each row (-1 will indicate that the position is empty). **Make this board using two loops** – the first loop will define a row list one element at a time (do NOT define this list all at once – build it by appending), and the second loop will make a copy of the row list and place it in the board 10 times (once for each row in the board). When you are done, you should have a list of lists with each internal list containing -1 in every position.

Now make a list called `GAME` – it will be a list of four `BOARD` lists. Do not make four separate `BOARD`s by copying the code. We don't need any more `BOARD`s – just make four copies of the `BOARD` list and store the copies in the `GAME` list. The first two boards are the ship location boards, the second two boards are the "guess" board which show where the player has guessed. **VERY IMPORTANT: After the `BOARD`s are copied to the `GAME` list, do not use the `BOARD` variable for anything else in the program.**

Now that our boards and ships are ready, ask Player 1 to choose a location for the aircraft carrier. Do this by asking for a row number (1 through 10) and a column number (1 through 10). Next, ask for a direction – `L` for Left, `R` for Right, `U` for Up, `D` for Down. You may assume the user will not provide an invalid location for the ship, but you must validate the direction as a legitimate letter – as long as the user provides an invalid input, have him/her re-enter. Be sure to accept either uppercase or lowercase inputs.

Once the location is collected, place the ship in the appropriate spots by changing the ship location board in `GAME` (Player 1 has the first board in the list, Player 2 has the second board). Store the aircraft carrier by changing the correct five -1 values to 0 (since it's the first of the ships).

10% EXTRA CREDIT

Prepare a function `print_board()` that will receive one of the boards contained within `GAME` as argument and display that board by iterating through the board's rows and columns, spacing the elements with tabs.

```
>>>
Please choose a row (1 - 10): 7
Please choose a column (1 - 10): 3
Direction? (U)p, (D)own, (L)eft, (R)ight: u
 -1   -1   -1   -1   -1   -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1   -1   -1   -1   -1   -1
 -1   -1    0   -1   -1   -1   -1   -1   -1   -1
 -1   -1    0   -1   -1   -1   -1   -1   -1   -1
 -1   -1    0   -1   -1   -1   -1   -1   -1   -1
 -1   -1    0   -1   -1   -1   -1   -1   -1   -1
 -1   -1    0   -1   -1   -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1   -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1   -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1   -1   -1   -1   -1   -1
>>>
```

Board output is extra credit

30% EXTRA CREDIT

Prepare a function `add_ships()` that will receive a player number (1 or 2) and `GAME` as arguments and then use the same technique as the base program to place all ships in the proper board of `GAME`: collect a row, column, and direction from the user and place a ship. The value to place for each ship is its index in the sequence of ships: Aircraft Carrier is 0, Battleship is 1, etc. Below is an example run of this function:

```
Placing Carrier...

Please choose a row (1 - 10): 1
Please choose a column (1 - 10): 1
Direction? (U)p, (D)own, (L)eft, (R)ight: r


Placing Battleship...

Please choose a row (1 - 10): 2
Please choose a column (1 - 10): 2
Direction? (U)p, (D)own, (L)eft, (R)ight: r


Placing Destroyer...

Please choose a row (1 - 10): 3
Please choose a column (1 - 10): 5
Direction? (U)p, (D)own, (L)eft, (R)ight: r


Placing Submarine...

Please choose a row (1 - 10): 4
Please choose a column (1 - 10): 4
Direction? (U)p, (D)own, (L)eft, (R)ight: l


Placing PT boat...

Please choose a row (1 - 10): 8
Please choose a column (1 - 10): 8
Direction? (U)p, (D)own, (L)eft, (R)ight: l


  0     0     0     0     0    -1    -1    -1    -1    -1
 -1     1     1     1     1    -1    -1    -1    -1    -1
 -1    -1    -1    -1     2     2     2    -1    -1    -1
 -1     3     3     3    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1     4     4    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

>>>
```