Academic Integrity and MOSS

presented by Ben Laube

What is MOSS?

"Moss (for a Measure Of Software Similarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes."

How does MOSS work?

- Black Box: we can view the inputs and outputs, but not the internal processes
- Instructors submit files to compare
- System returns HTML pages that detail similarities between files
- Cannot be used to determine if plagiarism or cheating actually occurred
- Highlights similarities, up to instructors to conclude

What can MOSS do?

- Whitespace insensitivity
  - Differences in whitespace (indentation/between blocks/etc.) and identifier names are ignored.
- Noise suppression
  - Potential matches must be long enough to be considered significant.
- Position independence
  - Blocks can match regardless of position in the source files.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Talking with peers about what class concepts you should use for the assignment.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Talking with peers about what class concepts you should use for the assignment.

  Acting honestly! :)

  Hashing things out at a high level is often encouraged in comp sci courses.

  Note that "high level" just means "keep it general".

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Posting your solutions online or showing them to peers before submissions close

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Posting your solutions online or showing them to peers before submissions close

Acting dishonestly... >:^(

There's a big difference between helping someone debug and helping them rewrite a whole block.

When in doubt, have them reach out to a TA to get help.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Viewing code from a video/forum and writing the same block in your own solution

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Viewing code from a video/forum and writing the same block in your own solution

Acting dishonestly... >:^(

Manually typing the copied code, is the same as using copy/paste shortcuts.

Yes, even if you change the variable names. MOSS still detects this.

Avoiding cmd + c ≠ avoiding copying.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Meeting an instructor or TA in office hours to talk about out how to approach a problem.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Meeting an instructor or TA in office hours to talk about out how to approach a problem.

  Acting honestly! :)

  This is usually your safest bet when it comes to getting help! We know exactly what kind of solution you should be working toward and can provide tips that won't get you in trouble.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Taking code and renaming identifiers or shuffling the order of blocks.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Taking code and renaming variables or shuffling the order of blocks.

Acting dishonestly... >:^(

This is just copying with slightly more effort put into it.

Might be easy for a human grader to miss, but MOSS detects this.

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Working with someone on the same source file for a partner assignment

What Counts As Cheating?

In the School of Computing at UNL, academic dishonesty can take many forms. A list of examples can be found on the policy webpage.

Acting honestly or dishonestly?

- Taking code and renaming variables or shuffling the order of blocks.

  Acting honestly! :)

  When instructors explicitly allow working together for an assignment, they'll make it clear.

  By default, you can safely assume no direct collaboration is allowed on coding assignments.

Grading Overview

For CSCE 155A, assignment grading is done by Learning Assistants (LA's) and managed by Course Leaders (CL's). A general layout of the grading process can be found below:

- Grading is assigned to LA's by a CL after the assignment deadline has passed.
- Once grading is finished and quality checked by a CL, student scores are posted.
- Any student with questions about their grade has up to 10 calendar days after grades are posted to discuss with their respective grader.
  - The grader of an assignment can be found in the submission comments on Canvas.
  - After this period, scores are treated as finalized and future requests by students to revisit their score may be denied.

How Cheating is Caught

1. During grading, LA's will report any suspicions to the CL's.
2. Anything missed by the graders will often be picked up by MOSS.
3. The instructors and CL review all potential matches
   a. Regardless of how a potential match was identified
4. Decide if the files show enough evidence to flag the student for violating the academic integrity policy.

Getting Flagged

If it is decided that a potential match is an instance of cheating, every student involved will be flagged as being in violation of the academic integrity policy.

Each student will receive a sanction and a submission comment on Canvas explaining the sanction with deadlines for next steps.

Discussion With Instructors

- Any student flagged for cheating should meet with instructors to discuss the situation.
- If you know you did not cheat, this is your time to clear things up and remove the sanction.
- If a student does not reach out by the deadline given, then the sanction stands.

Note: being flagged for a violation should not be taken lightly. Possible academic dishonesty is reviewed by multiple members of the instructor team and discussed at length before being flagged.

Consequences

A complete explanation of possible consequences for policy violations can be found on the policy webpage. In general, the SoC uses a three-strike system

1. For a first offense, the student will receive no credit for the assignment.
    a. The sanction can increase straight to a failing grade in the class if the violation was bad enough.
2. For a second offense, the student will receive a failing grade in the class and will not be permitted to drop the class.
    a. F stays on transcript.
3. For a third offense, the student will be expelled from their computing major or minor and will be barred from enrolling in future SoC courses.

All offenses will be reported to the Office of Student Conduct, regardless of the sanction imposed.

Appeals Process

- You are allowed to challenge academic dishonesty sanctions
  - For example, if you believe you are not guilty or feel the sanction was worse than called for.
- Contact the Chair of the School of Computing
- Chair assigns the case to an appropriate committee
- Committee meets with the student and the instructors to review the evidence, and make a recommendation to the instructor regarding the incident
- The instructor will review the recommendation, and may or may not amend the original decision.

If a student is still not satisfied with the instructors decision, they have the right to appeal at the university level.

Further Reading

- SoC academic integrity policy
- University of Nebraska Student Code of Conduct
- Moss documentation

Sample MOSS Results

Student 1

```python
# student01 - CSCE155A

from random import randint

# define a list for storing the board configuration
board = []


# function to create board
def create_board(size):
    for x in range(size):
        board.append(["~"] * size)


def random_row(size):
    return randint(0, size - 1)


def random_col(size):
    return randint(0, size - 1)


# function that places ship # in the random x,y coordinates
def place_ship():
    x = random_row(size)
    y = random_col(size)
    board[x][y] = "X"


# function that displays board configuration
def display_board(brd):
    for r in brd:
        print(" ".join(r))


# function that places X if there is ship in specified loc.
# Otherwise, it places O
def launch_missile(board, row, col):
    if board[row][col] == "X":
        print("Boom! You hit a ship!")
        board[row][col] = "!"
    else:
        print("You missed! Try again.")
        board[row][col] = "O"
    return board


print("Let's play Battleship!")
# prompt and accept board size
size = int(input("Enter board size: "))
create_board(size)
total_tiles = size * size
# number of ships must be less than one-fifth of number of titles
numShips = int((1 / 5) * total_tiles)
for i in range(numShips + 1):
    place_ship()
display_board(board)

# prompt and accept location of missile
for turn in range(size * size):
    coord_x, coord_y = [int(i) for i in input("Pick a coordinate to launch missile: ").split()]

    col = coord_x
    row = coord_y

    if (row < 0) or (row > (size - 1)) or (col < 0) or (col > size - 1):
        print("oops! That's not part of board")
        continue
    else:
        # launch missile in the specified loc
        board = launch_missile(board, row, col)
        display_board(board)
```

Student 2

```python
# Battleship game

from random import randint
# define a list for storing the board configuration
board = []
# function to create board
def create_board(size):
    for x in range(size):
        board.append(["."] * size)
# function that places ship # in the random x,y coords
def place_ship():
    x = random_row(size)
    y = random_col(size)
    board[x][y]="#"
# function that displays board configuration
def display_board(brd):
    for r in brd:
        print(" ".join(r))
# function to generate random x-coord
def random_row(size):
    return randint(0, size - 1)
# function to generate random y-coord
def random_col(size):
    return randint(0, size - 1)
# function that places x if there is ship in specified loc.
#otherwise, it places 0
def launch_missile(board,row,col):
    if(board[row][col]=="#"):
        print("You hit a ship!")
        board[row][col]="X"
    else:
        print("You missed!")
        board[row][col]="0"
    return board
print("Let's play Battleship!")
# prompt and accept board size
size = int(input("Enter board size: "))
create_board(size)
total_tiles = size*size
# number of ships must be less than one-fifth of number of titles
numShips = int((1/5)*total_tiles)
for i in range(numShips+1):
    place_ship()
    display_board(board)

# prompt and accept location of missle
for turn in range(size):
    coord_x, coord_y = [int(i) for i in input("Pick a coordinate to launch missile: ").split()]

    col = coord_x
    row = coord_y
    if((row< 0) or (row>(size-1)) or (col<0) or (col>size-1)):
        print("Sike! That's not part of the board")
    else:
        # launch missile in the specified loc
        board = launch_missile(board,row,col)
        display_board(board)
```

Student 1

Student 2

./moss_samples/sample01/student01_battleship.py (70%)		./moss_samples/sample01/student02_battleship.py (71%)	
38-60		27-47	
60-66		47-52	
24-37		11-19	

Student 1

```python
def place_ship():
    x = random_row(size)
    y = random_col(size)
    board[x][y] = "X"


# function that displays board configuration
def display_board(brd):
    for r in brd:
        print(" ".join(r))


# function that places X if there is ship in specified loc,
# otherwise, it places 0


def launch_missile(board, row, col):
    if board[row][col] == "X":
        print("Boom! You hit a ship!")
        board[row][col] = "!"
    else:
        print("You missed! Try again.")
        board[row][col] = "O"
    return board


print("Let's play Battleship!")
# prompt and accept board size
size = int(input("Enter board size: "))
create_board(size)
total_tiles = size * size
# number of ships must be less than one-fifth of number of titles
numShips = int((1 / 5) * total_tiles)
for i in range(numShips + 1):
    place_ship()
display_board(board)

# prompt and accept location of missile
for turn in range(size * size):


    coord_x, coord_y = [int(i) for i in input("Pick a coordinate to launch missile: ").split()]

    col = coord_x
    row = coord_y

    if (row < 0) or (row > (size - 1)) or (col < 0) or (col > size - 1):
```

Student 2

```python
def place_ship():
    x = random_row(size)
    y = random_col(size)
    board[x][y]="#"
# function that displays board configuration
def display_board(brd):
    for r in brd:
        print(" ".join(r))
# function to generate random x-coord


def launch_missile(board,row,col):
    if(board[row][col]=="#"):
        print("You hit a ship!")
        board[row][col]="X"
    else:
        print("You missed!")
        board[row][col]="0"
    return board
print("Let's play Battleship!")
# prompt and accept board size
size = int(input("Enter board size: "))
create_board(size)
total_tiles = size*size
# number of ships must be less than one-fifth of number of titles
numShips = int((1/5)*total_tiles)
for i in range(numShips+1):
    place_ship()
    display_board(board)

# prompt and accept location of missle
for turn in range(size):


    coord_x, coord_y = [int(i) for i in input("Pick a coordinate to launch missile: ").split()]

    col = coord_x
    row = coord_y
    if((row< 0) or (row>(size-1)) or (col<0) or (col>size-1)):
```

Student 3

```python
# Battleship game

import random
import numpy as np
import os
import time

# Variable Definition
num_ships = 0
size = int(input("Enter a board size: "))
ocean = np.full((size,size),' ')

# Make the map
def makeMap_function(size, num_ships, ocean):
    for x in range(0,size):
        for y in range(0,size):
            dice = random.randint(1,5)
            if (dice == 1):
                if num_ships < (1/5 * size * size):
                    ocean[x][y] = "🚢"
                    num_ships += 1
                else:
                    ocean[x][y] = '🌊'
            else:
                ocean[x][y] = '🌊'
    print("There are", num_ships, "ships in the ocean.")
    return num_ships

# Print the map
def print_array(arr):
    os.system('clear')
    for x in range(0,size):
        for y in range(0,size):
            print("\033[0;40;33m" + str(arr[y][x]),end=' ')
        print()

# Shooting function
def shooting_function(num_ships):
    while True:
        coord_x = int(input("\033[0;34;37mPick an x coordinate: "))
        coord_y = int(input("\033[0;34;37mPick a y coordinate: "))
        if coord_x < 0 or coord_x > size - 1 or coord_y < 0 or coord_y > size - 1:
            print("\033[0;34;31mNot a valid coordinate.")
            continue
        if ocean [coord_x][coord_y] == "🚢":
            ocean [coord_x][coord_y] = "X"
            print_array(ocean)
            print("\033[0;34;32mConfirmed hit!")
            num_ships -= 1
            if num_ships == 1:
                print(num_ships, "ship remaining")
            else:
                print(num_ships, "ships remaining")
        elif ocean [coord_x][coord_y] == "🌊":
            ocean [coord_x][coord_y] = "O"
            print_array(ocean)
            print("\033[0;34;32mHow could you miss? The ships are right in front of you!")
            if num_ships == 1:
                print(num_ships, "ship remaining")
            else:
                print(num_ships, "ships remaining")
        if num_ships == 0:
            print("Congrats! You destroyed all enemy ships. You are our savior!")
            exit()

# Main Code
num_ships = makeMap_function(size, num_ships, ocean)
print_array(ocean)
shooting_function(num_ships)
```

Student 4

```python
import numpy as np
import random
import math

board_size = int(input('What would you like the size of your board to be?'))
board = np.full((board_size,board_size),'~')
total_slots = board_size * board_size
ship_slots = int(total_slots/5)
ships_per_row = ship_slots/board_size
ships_remaining = int(total_slots/5)
end_flag = 0
def ships(ship_slots,board_size,ships_per_row):
    global board
    for i in range(ship_slots):
            x = random.randint(0,board_size - 1)
            y = random.randint(0, board_size - 1)
            board[x][y] = '#'

def missle_launch(ship_slots,total_slots):
    global end_flag
    global ships_remaining
    global board
    while (ships_remaining > 0):
        x = int(input('What is the Y coordinate of your missile strike')) - 1
        y = int(input('What is the X coordinate of your missile strike')) - 1
        if board[x][y] == '#':
            board[x][y] = 'X'
            print('Direct Hit!!')
            ships_remaining -= 1
        else:
            board[x][y] = 'O'
            print('You missed')
        for n in range(board_size):
            print()
            for m in range(board_size):
                print(board[n][m],end='')
        print()
        print('Ships remaining', ships_remaining)


while True:
    if ships_remaining > 0:
        ships(ship_slots,board_size,ships_per_row)
        end_flag = missle_launch (ship_slots,total_slots)
    else:
        break

print('Nice work sailor, you defeated all the enemy ships')
```

Student 5

```python
import random

#Function that will create a board in a list manner
def new_board(board_size):
    return[['.' for i in range(board_size)] for i in range(board_size)]

#Function that will print the board without brackets
def display_board(board):
    for i in board:
        print(*i)

#Function that will create battleship and keep them hidden
def create_battleship(board_size):

    #If board is 2 or smaller lognth of ship is 1
    if board_size < 2:
        ship_len = 1
    #If board is 4 or smaller lognth of ship is 2
    elif board_size < 4:
        ship_len = 2
    #If board is 6 or smaller, lognth of ship is either 2 or 3
    elif board_size < 6:
        ship_len = random.randint(2,3)
    #If board is 8 or smaller, lognth of ship is either 2,3, or 4
    elif board_size < 8:
        ship_len = random.randint(2,4)
    #If board greater than 9, lognth of ship is either 2,3,4 or 5
    else:
        ship_len = random.randint(2,5)

    ship_pattern = random.randint(0, 1)

    #If orientation is 0, make the battleship horizontal, else make it verticle
    if ship_pattern == 0:
        ship_row = [random.randint(0, board_size - 1)] * ship_len
        cols = random.randint(0, board_size - ship_len)
        ship_col = list(range(cols, cols + ship_len))
        coordinates = tuple(zip(ship_row, ship_col))
    else:
        ship_col = [random.randint(0, board_size - 1)] * ship_len
        row = random.randint(0, board_size - ship_len)
        ship_row = list(range(row, row + ship_len))
        coordinates = tuple(zip(ship_row, ship_col))

    return list(coordinates)

def verify_coords(board_size):
    temp = False
    col, row = [int(i) for i in input("Pick a coordinate (x, y) to launch a missile (input space input) : \n>>").split()]

    #Loops all variable until the coordinates (x,y) are valid and can be on the board
    while temp == False:

        if row < 0 or row > board_size - 1:
            row = int(input("Please (re)enter your row (y) value that is not greater or less than the board:\n>>"))
            row = row - 1
            temp = False
        elif col < 0 or col > board_size - 1:
            col = int(input("Please (re)enter your column (x) value that is not greater or less than the board:\n>>"))
            col = col - 1
            temp = False
        else:
            temp = True

    return(row, col)

def update_board(given_coords, board, ship, coords):

    if given_coords in coords:
        print("You already made that given_coords! Try again...")
        return board
    coords.append(given_coords)
    board[given_coords[0]] [given_coords[1]] = 'O'
    if given_coords in ship:
        print("You hit a battleship!")
        board[given_coords[0]] [given_coords[1]] = 'X'
        ship.remove(given_coords)
        return board
    print("You miss! Try again...")
    return board

#Print Victory and exit Program
def victory_point():
    print("Congrats! You have sunk my battleship!")
    exit()


#Main function (For full points, less than 10 pieces of code in main, [can be done in 6])
if __name__ == '__main__':
    board_size = int(input("Enter a board size greater than 1: "))
    board = new_board(board_size)
    ships = create_battleship(board_size)
    coords = []
    display_board(board)
    while(len(ships) > 0):
        valid_coords = verify_coords(board_size)
        board = update_board(valid_coords, board, ships, coords)
        display_board(board)
    victory_point()
```

Student 6

```python
import random

#Function that will create a board in a list manner
def create_board(grid_size):
    return[['.' for count in range(grid_size)] for count in range(grid_size)]

#Function that will print the board without brackets
def print_board(board):
    for b in board:
        print(*b)

#Function that will create battleship and keep them hidden
def create_battleship(grid_size):

    #If board is 2 or smaller lognth of ship is 1
    if grid_size < 2:
        battleship_length = 1
    #If board is 4 or smaller lognth of ship is 2
    elif grid_size < 4:
        battleship_length = 2
    #If board is 6 or smaller, lognth of ship is either 2 or 3
    elif grid_size < 6:
        battleship_length = random.randint(2,3)
    #If board is 8 or smaller, lognth of ship is either 2,3, or 4
    elif grid_size < 8:
        battleship_length = random.randint(2,4)
    #If board greater than 9, lognth of ship is either 2,3,4 or 5
    else:
        battleship_length = random.randint(2,5)

    battleship_orientation = random.randint(0, 1)

    #If orientation is 0, make the battleship horizontal, else make it verticle
    if battleship_orientation == 0:
        battleship_row = [random.randint(0, grid_size - 1)] * battleship_length
        cols = random.randint(0, grid_size - battleship_length)
        battleship_col = list(range(cols, cols + battleship_length))
        coordinates = tuple(zip(battleship_row, battleship_col))
    else:
        battleship_col = [random.randint(0, grid_size - 1)] * battleship_length
        row = random.randint(0, grid_size - battleship_length)
        battleship_row = list(range(row, row + battleship_length))
        coordinates = tuple(zip(battleship_row, battleship_col))

    return list(coordinates)

def user_guess(grid_size):
    valid = 'False'
    print("***Keep in mind that there is no 0 index in this program***")

    col, row = [int(i) for i in input("Pick a coordinate (x, y) to launch a missile (input *space* input) : \n>>").split()]

    #Removes 0 index
    row = row - 1
    col = col - 1

    #Loops all variable until the coordinates (x,y) are valid and can be on the board
    while valid == 'False':

        if row < 0 or row > grid_size - 1:
            row = int(input("Please (re)enter your row (y) value that is not greater or less than the board:\n>>"))
            row = row - 1
            valid = 'False'
        elif col < 0 or col > grid_size - 1:
            col = int(input("Please (re)enter your column (x) value that is not greater or less than the board:\n>>"))
            col = col - 1
            valid = 'False'
        else:
            valid = "True"

    return(row, col)

def board_update(guess, board, ship, guesses):

    if guess in guesses:
        print("You already made that guess! Try again...")
        return board
    guesses.append(guess)
    board[guess[0]] [guess[1]] = 'O'
    if guess in ship:
        print("You hit a battleship!")
        board[guess[0]] [guess[1]] = 'X'
        ship.remove(guess)
        return board
    print("You miss! Try again...")
    return board

#Print Victory and exit Program
def win():
    print("Congrats! You have sunk my battleship!")
    exit()


#Main function (For full points, less than 10 pieces of code in main, [can be done in 6])
if __name__ == '__main__':
    board_area = int(input("Enter a board size greater than 1: "))
    board = create_board(board_area)
    ship = create_battleship(board_area)
    guesses = []
    print_board(board)
    while(len(ship) > 0):
        attempt_guess = user_guess(board_area)
        board = board_update(attempt_guess, board, ship, guesses)
        print_board(board)
    win()
```

Student 5

Student 6

./moss_samples/sample03/student05_battleship.py (78%)		./moss_samples/sample03/student06_battleship.py (76%)	
1-48		2-49	
65-97		72-104	

Student 5

```python
import random

#Function that will create a board in a list manner
def new_board(board_size):
    return[['.' for i in range(board_size)] for i in range(board_size)]

#Function that will print the board without brackets
def display_board(board):
    for i in board:
        print(*i)

#Function that will create battleship and keep them hidden
def create_battleship(board_size):

    #If board is 2 or smaller legnth of ship is 1
    if board_size < 2:
        ship_len = 1
    #If board is 4 or smaller legnth of ship is 2
    elif board_size < 4:
        ship_len = 2
    #If board is 6 or smaller, legnth of ship is either 2 or 3
    elif board_size < 6:
        ship_len = random.randint(2,3)
    #If board is 8 or smaller, legnth of ship is either 2,3, or 4
    elif board_size < 8:
        ship_len = random.randint(2,4)
    #If board greater than 9, legnth of ship is either 2,3,4 or 5
    else:
        ship_len = random.randint(2,5)

    ship_pattern = random.randint(0, 1)

    #If orientation is 0, make the battleship horizontal, else make it verticle
    if ship_pattern == 0:
        ship_row = [random.randint(0, board_size - 1)] * ship_len
        cols = random.randint(0, board_size - ship_len)
        ship_col = list(range(cols, cols + ship_len))
        coordinates = tuple(zip(ship_row, ship_col))
    else:
        ship_col = [random.randint(0, board_size - 1)] * ship_len
        row = random.randint(0, board_size - ship_len)
        ship_row = list(range(row, row + ship_len))
        coordinates = tuple(zip(ship_row, ship_col))

    return list(coordinates)

def verify_coords(board_size):
    temp = False
```

Student 6

```python
import random

#Function that will create a board in a list manner
def create_board(grid_size):
    return[['.' for count in range(grid_size)] for count in range(grid_size)]

#Function that will print the board without brackets
def print_board(board):
    for b in board:
        print(*b)

#Function that will create battleship and keep them hidden
def create_battleship(grid_size):

    #If board is 2 or smaller legnth of ship is 1
    if grid_size < 2:
        battleship_length = 1
    #If board is 4 or smaller legnth of ship is 2
    elif grid_size < 4:
        battleship_length = 2
    #If board is 6 or smaller, legnth of ship is either 2 or 3
    elif grid_size < 6:
        battleship_length = random.randint(2,3)
    #If board is 8 or smaller, legnth of ship is either 2,3, or 4
    elif grid_size < 8:
        battleship_length = random.randint(2,4)
    #If board greater than 9, legnth of ship is either 2,3,4 or 5
    else:
        battleship_length = random.randint(2,5)

    battleship_orientation = random.randint(0, 1)

    #If orientation is 0, make the battleship horizontal, else make it verticle
    if battleship_orientation == 0:
        battleship_row = [random.randint(0, grid_size - 1)] * battleship_length
        cols = random.randint(0, grid_size - battleship_length)
        battleship_col = list(range(cols, cols + battleship_length))
        coordinates = tuple(zip(battleship_row, battleship_col))
    else:
        battleship_col = [random.randint(0, grid_size - 1)] * battleship_length
        row = random.randint(0, grid_size - battleship_length)
        battleship_row = list(range(row, row + battleship_length))
        coordinates = tuple(zip(battleship_row, battleship_col))

    return list(coordinates)

def user_guess(grid_size):
    valid = 'False'
```

Student 5

```
    return(row, col)

def update_board(given_coords, board, ship, coords):

    if given_coords in coords:
        print("You already made that given_coords! Try again...")
        return board
    coords.append(given_coords)
    board[given_coords[0]] [given_coords[1]] = 'O'
    if given_coords in ship:
        print("You hit a battleship!")
        board[given_coords[0]] [given_coords[1]] = 'X'
        ship.remove(given_coords)
        return board
    print("You miss! Try again...")
    return board

#Print Victory and exit Program
def victory_point():
    print("Congrats! You have sunk my battleship!")
    exit()


#Main function (For full points, less than 10 pieces of code in main, [can be done in 6])
if __name__ == '__main__':
    board_size = int(input("Enter a board size greater than 1: "))
    board = new_board(board_size)
    ships = create_battleship(board_size)
    coords = []
    display_board(board)
    while(len(ships) > 0):
        valid_coords = verify_coords(board_size)
        board = update_board(valid_coords, board, ships, coords)
```

Student 6

```
    return(row, col)

def board_update(guess, board, ship, guesses):

    if guess in guesses:
        print("You already made that guess! Try again...")
        return board
    guesses.append(guess)
    board[guess[0]] [guess[1]] = 'O'
    if guess in ship:
        print("You hit a battleship!")
        board[guess[0]] [guess[1]] = 'X'
        ship.remove(guess)
        return board
    print("You miss! Try again...")
    return board

#Print Victory and exit Program
def win():
    print("Congrats! You have sunk my battleship!")
    exit()


#Main function (For full points, less than 10 pieces of code in main, [can be done in 6])
if __name__ == '__main__':
    board_area = int(input("Enter a board size greater than 1: "))
    board = create_board(board_area)
    ship = create_battleship(board_area)
    guesses = []
    print_board(board)
    while(len(ship) > 0):
        attempt_guess = user_guess(board_area)
        board = board_update(attempt_guess, board, ship, guesses)
```

Thank you for listening

Questions?