Week 4 Assignment

Deadline: 7 September 2021, 9:00 A.M.

Submission:

1. Create StudentID_Firstname_Wk4 folder, where StudentID is your KU ID and Firstname is your given name.
2. Place files to submits: board.py and rock.py in the folder.
3. Compress the folder and submit the compressed file in Google Classroom by the submission deadline (Your compressed file should be named under StudentID_Firstname_Wk4)
4. Submit a summary text file (filename: summary.txt). In this summary, tell us what you have completed and what you have not. Submit in Google Classroom as well.

Grading Criteria:

1. Correctness (95%): Your program must run and give the expected result. Make sure your function names are defined correctly.
2. Cleanliness (5%): Your program should be clean; variable names are meaningful; unused or redundant code are removed; comments are added for others to understand your code easily.
   In the following assignments, PEP8 convention will be applied with cleanliness. You can check out about PEP 8 convention here: ***https://www.python.org/dev/peps/pep-0008/***

Part 1 : Boardgame Moving

Filename: board.py

Write a program as if a list of n players take turn to move on board game. At each turn, the players will toss a dice in a round robin style. The number obtained from dice will determine how far each player can move along the board. After playing for m turns, report which player walks the farthest distance as the winner.

Sample Outputs:

```
Enter number of players: 3
Enter number of rounds: 8
Initializing...
Player 1 : Distance = 0
Player 2 : Distance = 0
Player 3 : Distance = 0
Playing...
Round 1:
Player 1 : Distance = 3
Player 2 : Distance = 6
Player 3 : Distance = 2
Round 2:
```

```
Enter number of players: 5
Enter number of rounds: 12
Initializing...
Player 1 : Distance = 0
Player 2 : Distance = 0
Player 3 : Distance = 0
Player 4 : Distance = 0
Player 5 : Distance = 0
Playing...
Round 1:
Player 1 : Distance = 5
Player 2 : Distance = 5
```

Player 1 : Distance = 7 Player 2 : Distance = 8 Player 3 : Distance = 5 Round 3: Player 1 : Distance = 9 Player 2 : Distance = 13 Player 3 : Distance = 9 Round 4: Player 1 : Distance = 14 Player 2 : Distance = 16 Player 3 : Distance = 15 Round 5: Player 1 : Distance = 15 Player 2 : Distance = 21 Player 3 : Distance = 17 Round 6: Player 1 : Distance = 21 Player 2 : Distance = 27 Player 3 : Distance = 23 Round 7: Player 1 : Distance = 27 Player 2 : Distance = 33 Player 3 : Distance = 28 Round 8: Player 1 : Distance = 28 Player 2 : Distance = 36 Player 3 : Distance = 29 Game over... Player 2 wins with maximum distance = 36	Player 3 : Distance = 3 Player 4 : Distance = 5 Player 5 : Distance = 2 Round 2: Player 1 : Distance = 7 Player 2 : Distance = 9 Player 3 : Distance = 8 Player 4 : Distance = 8 Player 5 : Distance = 6 Round 3: Player 1 : Distance = 9 Player 2 : Distance = 10 Player 3 : Distance = 10 Player 4 : Distance = 12 Player 5 : Distance = 12 Round 4: Player 1 : Distance = 13 Player 2 : Distance = 11 Player 3 : Distance = 13 Player 4 : Distance = 15 Player 5 : Distance = 13 Round 5: Player 1 : Distance = 18 Player 2 : Distance = 17 Player 3 : Distance = 19 Player 4 : Distance = 17 Player 5 : Distance = 19 Round 6: Player 1 : Distance = 22 Player 2 : Distance = 22 Player 3 : Distance = 21 Player 4 : Distance = 18 Player 5 : Distance = 20 Round 7: Player 1 : Distance = 23 Player 2 : Distance = 27 Player 3 : Distance = 27 Player 4 : Distance = 22 Player 5 : Distance = 23 Round 8: Player 1 : Distance = 26 Player 2 : Distance = 29 Player 3 : Distance = 33 Player 4 : Distance = 25 Player 5 : Distance = 27 Round 9: Player 1 : Distance = 30 Player 2 : Distance = 35

	Player 3 : Distance = 39 Player 4 : Distance = 31 Player 5 : Distance = 31 Round 10: Player 1 : Distance = 32 Player 2 : Distance = 38 Player 3 : Distance = 44 Player 4 : Distance = 37 Player 5 : Distance = 33 Round 11: Player 1 : Distance = 33 Player 2 : Distance = 42 Player 3 : Distance = 46 Player 4 : Distance = 38 Player 5 : Distance = 39 Round 12: Player 1 : Distance = 38 Player 2 : Distance = 44 Player 3 : Distance = 48 Player 4 : Distance = 41 Player 5 : Distance = 45 Game over... Player 3 wins with maximum distance = 48

List of functions

Function Name	Function Inputs	Outputs	Functionality
initialize_list	Number of players (or n)	A list of n zeros.	Create and return a list of n zeros
print_distance_list	A list of distance	Nothing	Print distances of all players
toss_dice	Nothing	One number	Randomize integer between 1 and 6 (including 1 and 6). Return the random integer as if it is outcome of dice tossing. See example of randomizing integer below.
play_one_round	A list of distance	Nothing	For each round, allow n players to take turn tossing a dice. Then, update and report the new distances.
play_all_rounds	Number of rounds (m), and a list of distance	Nothing	Repetitively play for m rounds.
find_winner	A list of distance	Two numbers	From a list of distance, return 2 numbers. The first number is the winning player. The second number is maximum distance of the winning player.

Example of randomizing an integer. **Do not forget to import random library.**

Sample code	Corresponding Output

`import random`
`print(random.randint(1, 3))`
`print(random.randint(1, 3))`
`print(random.randint(1, 3))`
`n = 10`
`x = random.randint(1, n)`
`print(x)`
`x = random.randint(1, n)`
`print(x)`
	
	2
	1
	2
	
	8
	
	3

Explanation: Note that random.randint(1, 3) randomizes between 1 and 3 (including 1 and 3). After each randint() is run, the random integer between 1 and 3 (inclusive) is generated. If we call randint() again, we may or may not get the same random integer as before. random.randint(1, n) randomizes between 1 and n (including 1 and n).	

Part 2 : Rock, Paper, Scissors

Filename: rock.py

Write a program for two teams to paly rock, paper and scissors. At the end, the program reports which team has the most number of wins.

There are two teams of n players. The teams will play m rounds of rock, paper, and scissors. Each team stores "number of wins (or statistics)" of each player in a list.

In each round, a representative from each team is randomly selected to play against the other team. Then, the program finds the round winner, and reports the updated number of wins (or statistics) of each team.

At the end, the program reports the winning team (with more number of wins) and the final scores of both teams.

Sample Outputs:

Enter number of players: 3 Enter number of rounds: 5 Initial Team Stats: Team 1: [0, 0, 0] Team 2: [0, 0, 0] Playing... Round 1: Team1: Player 1 plays with Hand: Paper Team2: Player 2 plays with Hand: Scissors Team 2 wins. Update Team Stats:	Enter number of players: 6 Enter number of rounds: 12 Initial Team Stats: Team 1: [0, 0, 0, 0, 0, 0] Team 2: [0, 0, 0, 0, 0, 0] Playing... Round 1: Team1: Player 5 plays with Hand: Scissors Team2: Player 4 plays with Hand: Paper Team 1 wins. Update Team Stats:

Team 1: [0, 0, 0] Team 2: [0, 1, 0] Round 2: Team1: Player 2 plays with Hand: Paper Team2: Player 1 plays with Hand: Rock Team 1 wins. Update Team Stats: Team 1: [0, 1, 0] Team 2: [0, 1, 0] Round 3: Team1: Player 1 plays with Hand: Scissors Team2: Player 2 plays with Hand: Scissors Both teams tie. Update Team Stats: Team 1: [0, 1, 0] Team 2: [0, 1, 0] Round 4: Team1: Player 2 plays with Hand: Rock Team2: Player 3 plays with Hand: Paper Team 2 wins. Update Team Stats: Team 1: [0, 1, 0] Team 2: [0, 1, 1] Round 5: Team1: Player 1 plays with Hand: Scissors Team2: Player 3 plays with Hand: Scissors Both teams tie. Update Team Stats: Team 1: [0, 1, 0] Team 2: [0, 1, 1] Game over... Team 2 wins. Final scores: 2 vs. 1	Team 1: [0, 0, 0, 0, 1, 0] Team 2: [0, 0, 0, 0, 0, 0] Round 2: Team1: Player 1 plays with Hand: Rock Team2: Player 5 plays with Hand: Scissors Team 1 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 1, 0] Team 2: [0, 0, 0, 0, 0, 0] Round 3: Team1: Player 6 plays with Hand: Rock Team2: Player 5 plays with Hand: Paper Team 2 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 1, 0] Team 2: [0, 0, 0, 0, 1, 0] Round 4: Team1: Player 2 plays with Hand: Scissors Team2: Player 2 plays with Hand: Scissors Both teams tie. Update Team Stats: Team 1: [1, 0, 0, 0, 1, 0] Team 2: [0, 0, 0, 0, 1, 0] Round 5: Team1: Player 5 plays with Hand: Rock Team2: Player 5 plays with Hand: Scissors Team 1 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 0, 0, 0, 1, 0] Round 6: Team1: Player 6 plays with Hand: Scissors Team2: Player 5 plays with Hand: Rock Team 2 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 0, 0, 0, 2, 0] Round 7: Team1: Player 1 plays with Hand: Scissors Team2: Player 2 plays with Hand: Scissors Both teams tie. Update Team Stats:

	Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 0, 0, 0, 2, 0] Round 8: Team1: Player 4 plays with Hand: Paper Team2: Player 2 plays with Hand: Paper Both teams tie. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 0, 0, 0, 2, 0] Round 9: Team1: Player 4 plays with Hand: Scissors Team2: Player 2 plays with Hand: Rock Team 2 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 1, 0, 0, 2, 0] Round 10: Team1: Player 1 plays with Hand: Scissors Team2: Player 6 plays with Hand: Rock Team 2 wins. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 1, 0, 0, 2, 1] Round 11: Team1: Player 2 plays with Hand: Paper Team2: Player 6 plays with Hand: Paper Both teams tie. Update Team Stats: Team 1: [1, 0, 0, 0, 2, 0] Team 2: [0, 1, 0, 0, 2, 1] Round 12: Team1: Player 3 plays with Hand: Paper Team2: Player 1 plays with Hand: Rock Team 1 wins. Update Team Stats: Team 1: [1, 0, 1, 0, 2, 0] Team 2: [0, 1, 0, 0, 2, 1] Game over... Both teams tie. Final scores: 4 vs. 4

Part 2 continues next page.

List of functions

Function Name	Function Inputs	Outputs	Functionality
initialize_list	Number of players (or n)	A list of n zeros.	Create and return a list of n zeros
print_two_team_stat	Two lists of statistics (or number of wins)	Nothing	Print statistics of two teams. Note that to convert a list L to string, you can use str(L).
randomize_player	Number of players (or n)	One integer	Randomize an integer between 1 and n (including 1 and n).
randomize_hand	Nothing	One string	Randomize integer between 1 and 3 (including 1 and 3). If random number is 1, return "Rock" If random number is 2, return "Paper" If random number is 3, return "Scissors"
play_one_round	Two lists of statistics	Nothing	Randomly choose one player from each team to play. Then, find out and report the round winner. After that, update and report the statistics.
find_round_winner	Two strings (or hands of two players)	One integer	First string is a hand of team1 player. Second string is a hand of team2 player. From hands of two players, find out who is the round winner. If team1 player wins, return 1. If team2 player wins, return 2. If the players tie, return 0.
report_winner	One integer (Indicating the winning team)	Nothing	Display which team wins, or both teams tie.
play_all_rounds	Number of rounds (m), and two lists of statistics	Nothing	Repetitively play rock, paper and scissors for m rounds.
find_final_winner	Two lists of statistics	Three numbers	From two lists of statistics, return 3 numbers. The first number is the winning team (1 or 2. In case of tie, return 0.) The second number is total number of wins **from the winning team** The third number is total number of wins **from the losing team** If both teams tie, the second and the third numbers are the same. For example, `find_final_winner([0,2,1],[1,3,1])` will return 2, 5, 3 since the winning team is

			team 2. Number of wins from the winning team is 5. Number of wins from the losing team is 3.